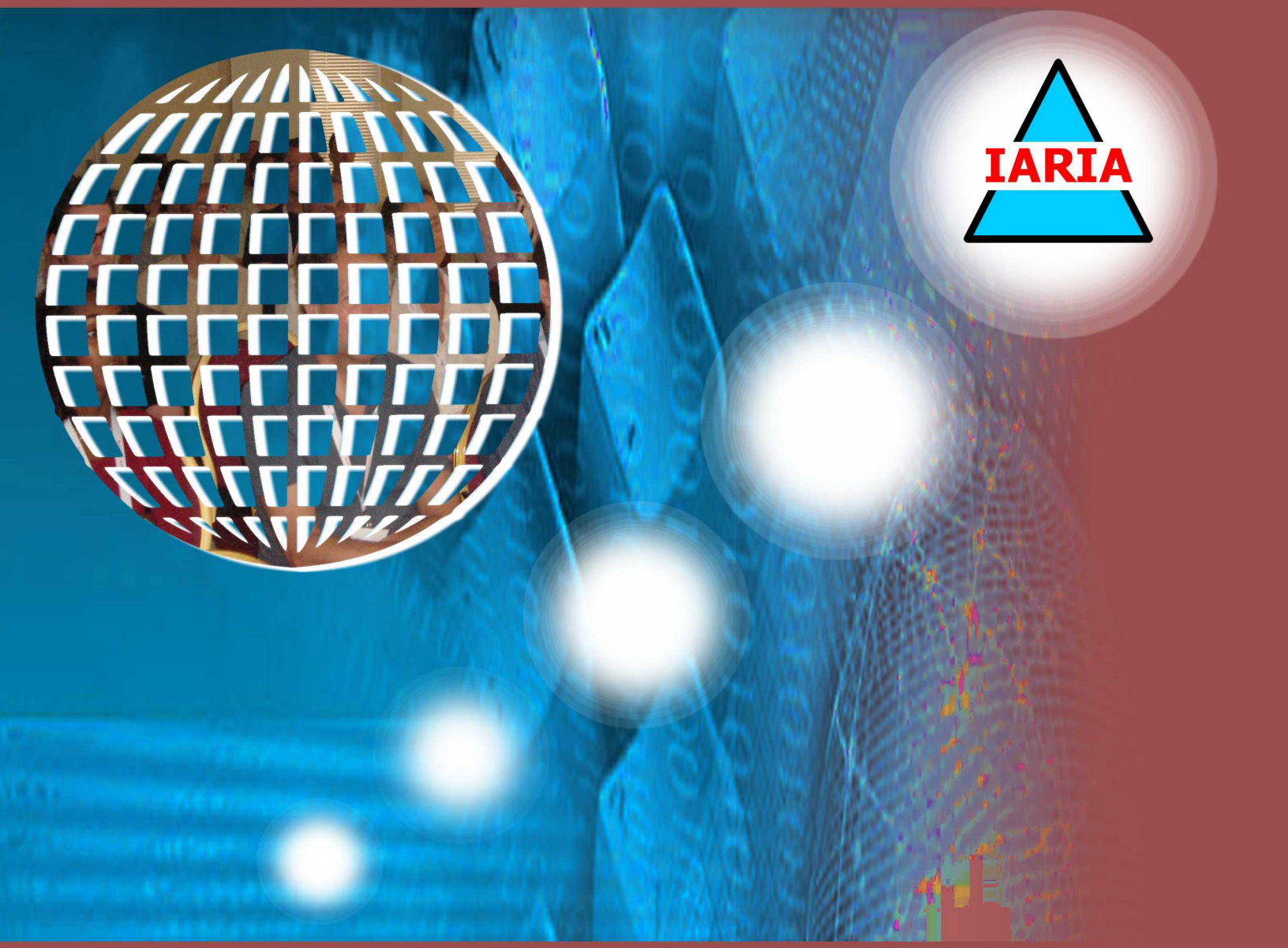
International Journal on Advances in Security



2023 vol. 16 nr. 1&2

The *International Journal on Advances in Security* is published by IARIA.
ISSN: 1942-2636
journals site: http://www.iariajournals.org
contact: petre@iaria.org

Sponsored by IARIA
www.iaria.org

International Journal on Advances in Security
Volume 16, Number 1 & 2, 2023

Sylvia Kierkegaard, IAITL-International Association of IT Lawyers, Denmark
Hyunsung Kim, Kyungil University, Korea
Geir M. Køien, University of Agder, Norway
Ah-Lian Kor, Leeds Metropolitan University, UK
Evangelos Kranakis, Carleton University - Ottawa, Canada
Lam-for Kwok, City University of Hong Kong, Hong Kong
Jean-Francois Lalande, ENSI de Bourges, France
Gyungho Lee, Korea University, South Korea
Clement Leung, Hong Kong Baptist University, Kowloon, Hong Kong
Diego Liberati, Italian National Research Council, Italy
Giovanni Livraga, Università degli Studi di Milano, Italy
Gui Lu Long, Tsinghua University, China
Jia-Ning Luo, Ming Chuan University, Taiwan
Thomas Margoni, University of Western Ontario, Canada
Rivalino Matias Jr ., Federal University of Uberlandia, Brazil
Manuel Mazzara, UNU-IIST, Macau / Newcastle University, UK
Catherine Meadows, Naval Research Laboratory - Washington DC, USA
Carla Merkle Westphall, Federal University of Santa Catarina (UFSC), Brazil
Ajaz H. Mir, National Institute of Technology, Srinagar, India
Jose Manuel Moya, Technical University of Madrid, Spain
Leonardo Mostarda, Middlesex University, UK
Jogesh K. Muppala, The Hong Kong University of Science and Technology, Hong Kong
Syed Naqvi, CETIC (Centre d'Excellence en Technologies de l'Information et de la Communication),Belgium
Sarmistha Neogy, Jadavpur University, India
Mats Neovius, Åbo Akademi University, Finland
Jason R.C. Nurse, University of Oxford, UK
Peter Parycek, Donau-Universität Krems, Austria
Konstantinos Patsakis, Rovira i Virgili University, Spain
João Paulo Barraca, University of Aveiro, Portugal
Sergio Pozo Hidalgo, University of Seville, Spain
Yong Man Ro, KAIST (Korea advanced Institute of Science and Technology), Korea
Rodrigo Roman Castro, University of Malaga, Spain
Heiko Roßnagel, Fraunhofer Institute for Industrial Engineering IAO, Germany
Claus-Peter Rückemann, Leibniz Universität Hannover / Westfälische Wilhelms-Universität Münster / North-German Supercomputing Alliance, Germany
Antonio Ruiz Martinez, University of Murcia, Spain
Paul Sant, University of Bedfordshire, UK
Peter Schartner, University of Klagenfurt, Austria
Alireza Shameli Sendi, Ecole Polytechnique de Montreal, Canada
Dimitrios Serpanos, Univ. of Patras and ISI/RC ATHENA, Greece
Pedro Sousa, University of Minho, Portugal
George Spanoudakis, City University London, UK
Vladimir Stantchev, Institute of Information Systems, SRH University Berlin, Germany
Lars Strand, Nofas, Norway
Young-Joo Suh, Pohang University of Science and Technology (POSTECH), Korea
Jani Suomalainen, VTT Technical Research Centre of Finland, Finland

International Journal on Advances in Security
Volume 16, Numbers 1 & 2, 2023

CONTENTS

Rainer Falk, Siemens AG, Germany
Steffen Fries, Siemens AG, Germany

pages: 96 - 105
Protected Establishment of a Secondary Network Access Channel
Steffen Fries, Siemens, Germany
Rainer Falk, Siemens, Germany

Enhanced Attack Resilience within Cyber Physical Systems

Rainer Falk, Steffen Fries
Siemens AG
Technology
Munich, Germany
e-mail: {rainer.falk|steffen.fries}@siemens.com

***Abstract*—Cyber physical systems control, monitor, and supervise physical, technical systems using information and communication technology, also called operation technology. The focus of cyber security is protection against cyber attacks, their detection, and recovery from successful cyber attacks. Cyber resilience aims at delivering an intended outcome of the cyber physical system despite attacks and adverse cyber events and even due to failures not directly related to attacks. Industrial security standards define how cyber physical systems and the used devices can be protected against attacks (prevent). Despite all efforts to protect from attacks, it should always be assumed that attacks may happen. Security monitoring allows to detect successful attacks (detect), so that corresponding measures can be performed (react). This paper describes an additional, complementary approach for protecting cyber physical systems. The devices are designed in a way that makes it harder to use them for launching attacks on other devices or on their physical environment. A device-internal hardware-based or isolated firewall limits the network traffic that the device software executed on the device can send or receive. Even if the device software contains a vulnerability allowing an attacker to compromise the device, the technically possible negative impact on other connected devices is limited, thereby enhancing the resilience of the cyber physical system in the presence of manipulated devices.**

Keywords–cyber security; cyber resilience; system integrity; cyber physical systems; industrial automation and control system; Internet of Things.

I. INTRODUCTION

Traditionally, IT security has been focusing on protection of confidentiality, integrity, and availability of data at rest and data in transit, and sometimes also protecting data in use by confidential computing or by applying homomorphic encryption. In cyber physical systems (CPS), major protection goals are availability, meaning that the CPS, e.g., an industrial automation system, stays productive, and system integrity, ensuring that the CPS is in fact operating as intended and designed. Typical industrial application domains are factory automation, process automation, building automation, railway signaling systems, intelligent traffic management, and power system management. Such CPSs are also called *industrial* Internet of Things (IIoT), distinguishing them from other Internet of Things (IoT) domains as, e.g., consumer IoT. Cyber security is covering different phases during operation as there are protect, detect, and react: Protecting against threats, detecting when an attack has occurred, and recovering from successful attacks. With the approach of "resilience under attack", it shall be ensured that the CPS can stay operational even during an ongoing attack, maybe with limited performance or functionality [1]. This property reduces the impact of a successful attack, as the CPS can be continued to be used even if parts of the CPS should have been attacked successfully. The availability of the overall CPS is thereby improved, as the CPS can stay operational even under an ongoing attack.

When designing a security solution for a CPS or for a device used within the CPS, the focus is on protecting the assets of the CPS or device, by preventing attacks against the relevant assets. However, this approach is not complete from a more holistic perspective: It is also important to detect ongoing attacks, and to recover efficiently from an attack. Also, the environment of a device or a CPS has to be protected from attacks originating from a manipulated CPS or one of its devices. In particular, IoT devices have been attacked with the objective to use them for launching attacks against *other* systems. Dao, Phan et al. described distributed denial of service (DDoS) attacks originating from manipulated IoT devices [2]. As (consumer) IoT devices have often also a weak security management, vulnerabilities are often not patched in time, making them an easy attack target. Vulnerable IoT devices connected to the Internet can easily be found by using Internet-based device search engines like Shodan [3] or Morpheus [4].

This paper presents an approach for protecting the network environment, i.e., other devices of a CPS and further connect devices, from attacks originating from a manipulated component of the CPS. It is an extended version of the conference paper [1], extending in particular the possible reduction in risk exposure, see section VI. The objective is to limit the impact of a manipulated CPS device on other devices of the CPS, thereby enhancing resilience of the overall CPS. The intention is to keep the CPS in a trustworthy operational state even if some devices of the CPS should have been successfully attacked and manipulated. The high-level objective addressed by this paper is to present a design for resilient CPS devices that limits the possibility that an attacked device can be used by an attacker for further attacks

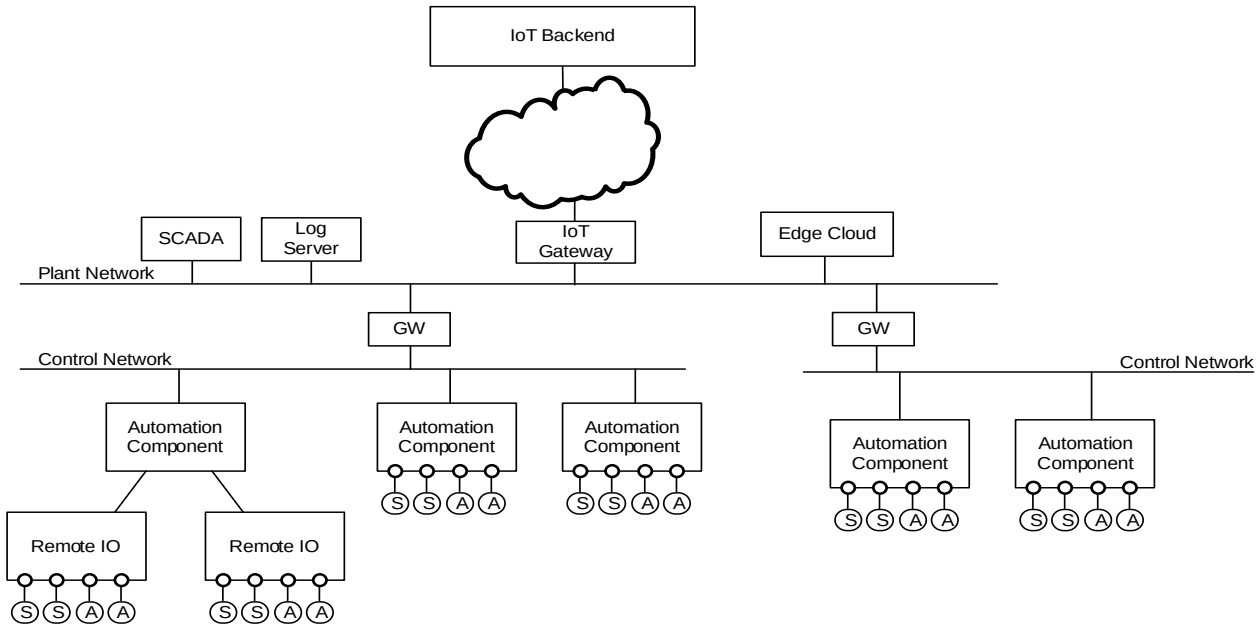


Figure 1. Example – Industrial Automation and Control System

on other devices of the CPS. A manipulated device could otherwise provide manipulated sensor or control data to other CPS devices or manipulate their firmware or configuration data. Such manipulations of CPS devices may finally lead to manipulated control operations impacting the real, physical world, i.e., on the controlled physical process, in a critical way. The main idea is to extend CPS devices with a resilience function that allows to isolate a manipulated operational CPS device from other devices of the CPS. This isolation from other CPS devices extends the isolation from accessing the physical world by a manipulated CPS device by a physical-world firewall [5].

After giving an overview on cyber physical systems and on industrial cyber security and IoT security in Sections II and III, a new approach on protecting the network environment from manipulated devices of a CPS is described in Section V. It is a concept to increase the resilience of a CPS when being under attack. Aspects to evaluate the new approach are discussed in Section VI, investigating the possible reduction of overall risk exposure of the CPS that can be achieved by the presented approach. Section VII concludes the paper.

II. Cyber Physical Systems

A CPS, e.g., an Industrial Automation and Control System (IACS), monitors and controls a technical system. Examples are process automation, machine control, energy automation, and cloud robotics. Automation control equipment with sensors (S) and actuators (A) is connected directly with automation components, or via remote input/output modules. The technical process is controlled by measuring its current state using the sensors, and by determining the corresponding actuator signals. Also, IoT in general can be seen as a cyber physical system, as IoT devices usually interact via the physical world using sensors and actuators [5].

Figure 1 shows an example of an industrial automation and control system, i.e., of an industrial CPS, comprising different control networks connected to a plant network and a cloud backend system. Separation of the network is typically used to realize distinct control networks with strict real-time requirements for the interaction between sensors and actuators of a production cell, or to enforce a specific security policy within a production cell. Such an industrial automation and control system is an example of a CPS and is utilized in various automation domains, including discrete automation (factory automation), process automation, railway automation, energy automation, and building automation.

Figure 2 shows the typical simplified structure of IoT devices, e.g., automation components. The functionality realized by an automation component is largely defined by the firmware/software and the configuration data stored in its flash memory (FW Flash). Such an automation component can also be considered as a remote input/output (IO) device, as in many cases, its operation is controlled by a different control device connected via the network interface (NW IF).

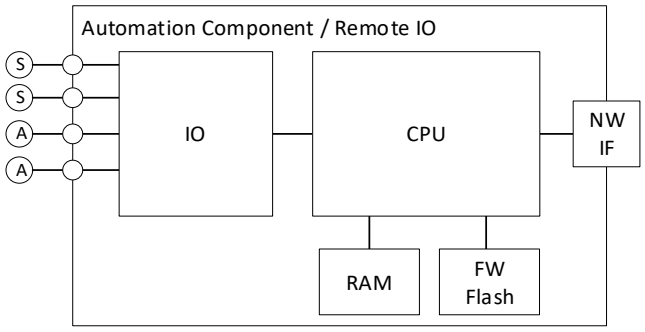


Figure 2. Automation Component

In practice, it has to be assumed that each software component may comprise vulnerabilities, independent of the effort spend to ensure high software quality. This is one reason why automation systems are usually organized in separate security zones. Network traffic can be filtered using network firewalls between different zones, limiting the impact of an impact in one security zone on other connected security zones. In addition, it is often not possible to fix known vulnerabilities immediately by installing a software update, as updates have to be tested thoroughly in a test system before being installed in an operational system, and as an installation is often possible only during a scheduled maintenance window. Also, the priorities of security objectives in different security zones are often different, too. In CPSs, the impact of a vulnerability in an OT system may not only affect data and data processing as in classical IT, but it may have an effect also on the physical world. For example, production equipment could be damaged, or the physical process may operate outside the designed physical boundaries, so that the produced goods may not have the expected quality or even that human health or life is endangered.

III. CYBER SECURITY OF CPS

Protecting IACSs against intentional attacks is increasingly demanded by operators to ensure a reliable operation, and also by regulation. This section gives an overview on industrial security, and on the main relevant industrial security standard IEC 62443 [15]. It summarizes also selected security standards addressing consumer IoT devices.

A. Industrial CPS Security Requirements

Industrial security is called also Operation Technology security (OT security), to distinguish it from general Information Technology (IT) security. Industrial systems have not only different security requirements compared to general IT systems but come also with specific side conditions preventing the direct application of security concepts established in the IT domain in an OT environment. For example, availability and integrity of an automation system often have a higher priority than confidentiality. As an example, high availability requirements, different organization processes (e.g., yearly maintenance windows), and required component or system certifications may prevent the immediate installations of updates.

The three basic security requirements in IT environments are confidentiality, integrity, and availability ("CIA" requirements). This CIA order corresponds to the priority of the basic security requirements typically relevant in common IT systems, However, in automation systems or industrial IT, the priorities are commonly just the other way around: Availability of the IACS has typically the highest priority, followed by integrity. Confidentiality is often no strong requirement for control communications, but may be needed to protect critical business know-how.

Figure 3 shows that in common IT systems, the priority is "CIA". As shown graphically, the CIA pyramid is inverted (turned upside down) in many automation systems.

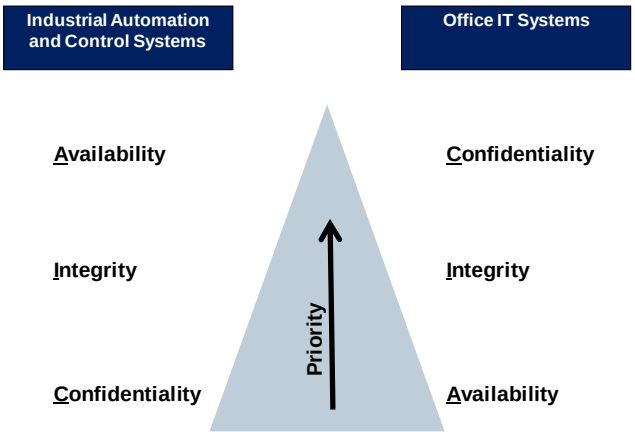


Figure 3. The CIA Pyramid [13]

Specific requirements and side conditions of an IACS like high availability, planned configuration (engineering info), long life cycles, unattended operation, real-time operation, and communication, as well as safety requirements have to be considered when designing a cyber security solution. Often, an important aspect is that the applied security measures do not put availability and integrity of the automation system at risk. Depending on the considered industry (vertical), they may also be part of the critical infrastructure domain, for which security requirements are also imposed for instance by the European Network and Information Systems (NIS) directive [14] or country specific realizations of the directive. Further security requirements are provided by applying standards defining functional requirements, for instance defined in IEC 62443. The defined security requirements can be mapped to different automation domains, including energy automation, railway automation, building automation, process automation.

Security measures to address these requirements range from security processes, personal and physical security, device security, network security, and application security. No single security technology alone is adequate, but a combination of security measures addressing prevention, detection, and reaction to incidents is required ("defense in depth").

B. Overview IEC 62443 Industrial Security Standard

The international industrial security framework IEC 62443 [15] is a security requirements framework defined by the International Electrotechnical Commission (IEC). It addresses the need to design cybersecurity robustness and resilience into industrial automation and control systems, covering both organizational and technical aspects of security over the life cycle. Specific parts of this framework are applied successfully in different automation domains, including factory and process automation, railway automation, energy automation, and building automation. The standard specifies security for Industrial Automation and Control Systems (IACS) along the lifecycle of industrial systems. Specifically addressed for the industrial domain is the setup of a security organization and the definition of security processes as part of an Information Security

Management System (ISMS) based on already existing standards like ISO 27001 [16] or the NIST cyber security framework. Furthermore, technical security requirements are specified distinguishing different security levels for industrial automation and control systems, and also for the used components. The standard has been created to address the specific requirements of industrial automation and control systems.

Different parts of the IEC62443 standard are grouped into four clusters, covering:

- common definitions and metrics;
- requirements on setup of a security organization (ISMS related, similar to ISO 27001 [16]), as well as solution supplier and service provider processes;
- technical requirements and methodology for security on system-wide level, and
- requirements on the secure development lifecycle of system components, and security requirements to such components at a technical level.

The framework parts address different roles over different phases of the system lifecycle: The operator of an IACS operates the IACS that has been integrated by the system integrator, using components of product suppliers. In the set of corresponding documents, security requirements are defined, which target the solution operator and the integrator but also the product manufacturer.

According to the methodology described in IEC 62443 part 3-2, a complex automation system is structured into zones that are connected by and communicate through so-called "conduits" that connect different zones. A conduit maps, e.g., to the logical network protocol communication between two zones over a firewall. Moreover, this document defines Security Levels (SL) that correspond with the strength of a potential adversary. To achieve a dedicated SL, the defined requirements have to be fulfilled.

Part 3-3 of IEC 62443 [18], addressing an overall automation system, is in particular relevant for the system integrator. It defines seven foundational requirements that group specific requirements of a certain category:

- FR 1 Identification and authentication control
- FR 2 Use control
- FR 3 System integrity
- FR 4 Data confidentiality
- FR 5 Restricted data flow
- FR 6 Timely response to events
- FR 7 Resource availability

For each of the foundational requirements, several concrete technical security requirements (SR) and requirement enhancements (RE) are defined. Related security requirements are defined for the components of an industrial automation and control system in IEC 62443 part 4-2 [19], addressing in particular component manufacturers.

C. Consumer IoT Security

While industrial CPS or industrial automation and control systems are called also industrial IoT, IoT in general includes also consumer IoT. Example consumer IoT devices are connected smoke sensors, connected home automation devices, connected washing machines, or smart cameras.

The standard EN 303 645 [20] defined by the European Telecommunications Standards Institute (ETSI) defines baseline security requirements for consumer IoT devices, addressing cyber security and data protection (user privacy). It is complemented by the assessment specification TS 103 701 [21] and a (draft) implementation guide TR 103 621 [22]. The cyber security provisions defined by EN 303 645 address the following areas:

- No universal default passwords
- Implement a means to manage reports of vulnerabilities
- Keep software updated
- Securely store sensitive security parameters
- Communicate securely
- Minimize exposed attack surfaces
- Ensure software integrity
- Ensure that personal data is secure
- Make systems resilient to outages
- Examine system telemetry data
- Make it easy for users to delete user data
- Make installation and maintenance of devices easy
- Validate input data

The defined security requirements or examples also consider resilience of the overall IoT system comprising a huge set of IoT devices, e.g., by randomizing when an update check is performed. Several provisions are defined addressing resilience of the system with respect to outages of data network connectivity or power supply. The main functionality of an IoT device shall be maintained locally when network connectivity is lost, ensuring that the main functionality is available to users independently of network availability. IoT devices may reconnect after power outage with a randomized delay to avoid an overload on the network infrastructure by a huge set of reconnecting IoT devices. This maintains the availability of the network infrastructure after a loss of power in a larger geographic area.

The National Institute of Standards and Technology (NIST) developed a standard for IoT security [23], with an associated catalogue of security requirements [24]. The catalogue defines requirements for the following main cyber security capabilities:

- Device identification
- Device configuration
- Data protection
- Logical access to interfaces
- Software update

– Cybersecurity state awareness
– Device security

In addition, also non-technical, supporting capabilities have been defined:

– Documentation
– Information and query reception
– Information dissemination
– Education and awareness

While resilience in general is not within the addressed scope, the requirements catalogue also includes examples related to resilience, e.g., that an IoT device continues operation when associated networks are unavailable. Furthermore, a draft defining baseline security criteria for consumer IoT devices has been published for comments [25]. A program on trustworthy network of things has been setup, where both IoT devices are protected from the Internet and where the Internet is protected from IoT devices, improving security and robustness of large-scale deployments of IoT devices. One aspect are manufacturer usage descriptions (MUD), as specified in [27], that define the intended communication behavior of an IoT device. It enables a network to block other types of network communication, reducing the attack surface associated to a specific IoT device. It also provides the possibility to automatically configure rules for communication traffic, based on the intended communication behavior. Such rules can be either directly applied, or after inspection and approval by a network administrator.

IV. RESILIENCE UNDER ATTACK

Being resilient means to be able to withstand or recover quickly from difficult conditions [30]. It shifts the focus of "classical" IT/OT cyber security, which puts the focus on preventing, detecting, and reacting to cyber-security attacks, to the aspect to continue to deliver an intended outcome even when an adverse cyber attack is taking place, and to recover quickly back to regular operation [31]. More specifically, resilience of a system is the property to be resistant to a range of threats and withstand the effects of a partial loss of capability, and to recover and resume its provision of service with the minimum reasonable loss of service quality (e.g., performance). It has been addressed in telecommunications, ensuring that subscribers can continue to be served even when one line is out of service [11]. Bodeau and Graubart [32] define resilience guidelines for providers of critical national telecommuni-cations infrastructure in the UK. Kott and Linkov [33] have compiled a book of different contributions addressing various aspects of cyber resilience in networks and systems. Besides an overview on cyber security, metrics to quantify cyber resilience, approaches to assess, analyze and to enhance cyber resilience are described. The notion of resilience is related to risk management, and also to robustness. Risk management, the "classical" approach to cyber security, identifies threats and determines the risk depending on probability and impact of a potential attack. The objective is to put the focus of defined security measures on the most relevant risks. Resilience, however, puts the focus on a reduction of the impact, so that the system stays operational with a degraded performance or functionality even when it has been attacked successfully, and to recover quickly from a successful attack. Robustness is a further related approach that tries to keep the system operational *without* a reduction of the system performance [12], i.e., to withstand attacks.

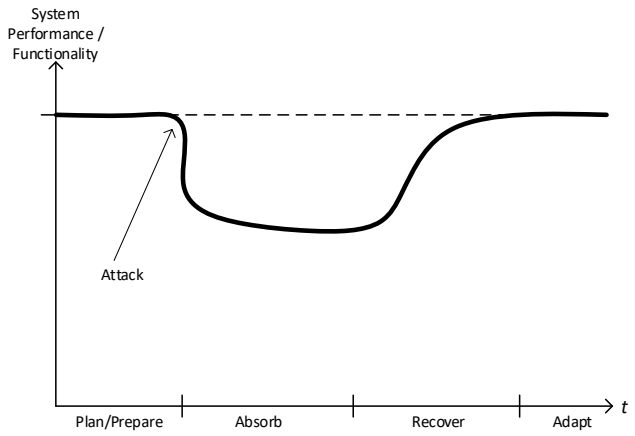


Figure 4. Concept of Cyber Resilience

Figure 4 illustrates the concept of cyber resilience: Even if an attack is carried out, the impact on the system operation, i.e., the performance or functionality of the system, is limited. The effects of an attack are "absorbed", so that the system stays operational, but with limited performance or functionality. A recovery takes place to bring the system up to the regular operation. In adaptation of resilience, the system might be enhanced to better prepare for future attacks. As stated above, a main objective in a CPS is that the CPS stays operational and that its integrity is ensured. In the context of an industrial automation and control system, that means that (only) intended actions of the system in the physical world continue to take place even when the automation and control system of the CPS should be attacked.

The standard NIST SP 800-193 [28] describes technology-independent guidelines for resilience of platform firmware. Resilience-specific roots of trust are defined for update of platform firmware, for detection a corrupted firmware, and for recovery from a compromised platform state. A working group on "cyber resilient technologies" of the Trusted Computing Group (TCG) is working on technologies to enhance cyber resilience of connected systems. Different building blocks for cyber resilient platforms have been described that allow to recover a malfunction device reliably back into a well-defined operational state [29]. Such building blocks enhance resilience as they allow to recover quickly and with reasonable effort from a manipulation. Basic building blocks are a secure execution environment for the resilience engine on a device, protection latches to protect access to persistent storage of the resilience engine even of a compromised device, and watchdog timers to ensure that the resilience engine can in fact perform a recovery. These building blocks are complementary to the extension described in section VI, and they may be used in combination.

V. Protecting Network Environment from Manipulated IoT Devices

The security objective "resilience under attack" means that a CPS, e.g., an IACS or an industrial IoT environment, should stay operational even when some of its components would be manipulated. Considering the manifold of devices used in real-world CPS and practical limitations to timely install patches, it has to be assumed that some of them will have vulnerabilities that can be exploited and be used to also infect further CPS devices. Hence, it shall be avoided that a successfully hacked device can be used to launch attacks against other devices. This is a specific security objective: When designing the security architecture for a device, usually attacks *against* the device are considered. Here, it shall be avoided that a successfully attacked device can be misused by an attacker to launch attacks on other devices within the CPS. So, the possibility to misuse a vulnerable CPS device to launch attacks on other devices of the CPS is reduced.

The software execution environment executes the software (firmware) of the device that might have a vulnerability. A separated, e.g., a separate hardware based, on-device firewall limits the network communication that the executed software can perform. This enforcement is realized independently from the executed device software, so that it is still working even if the device software has been manipulated by an attacker. This independence is a necessary pre-requisite. In the described design, this independence is achieved by separate hardware-based component. However, the independence from the executed device software could be achieved also by using an isolated software execution environment, e.g., a separate processor or a separate trusted execution environment. Using a hardware-based realization has the advantage of limiting the impact on real-time communication properties as delay and jitter, and also on the energy consumption. It can be easily implemented if a dedicated hardware-based network interface is in use anyhow to support real-time communication protocols.

Possible filter criteria are source and destination network addresses, protocols (e.g., TCP, UDP), port numbers, transmit rate (frames/packets per second), or data volume. In an advanced form, the firewall may even verify on application level whether certain control flows are aligned with either the typical (historical) behavior of the device or with the engineered CPS configuration information. The policy might be fixed, e.g., for embedded control devices with a fixed functionality, or it may be configurable. Important is that the device software cannot modify the filter policy on its own.

The filter policy might be adapted automatically depending on the patch status of the device software, or depending on the result of a device integrity self check of the IoT device, or based on a cryptographically protected health check confirmation received from a device integrity monitoring service. This would allow to keep the system operational, although with potentially limited capabilities, thus keeping it resilient. Also, limiting specific functionalities as result of missing device integrity may stipulate the timely application of patches, to get the system back to normal operation with full functionality and performance.

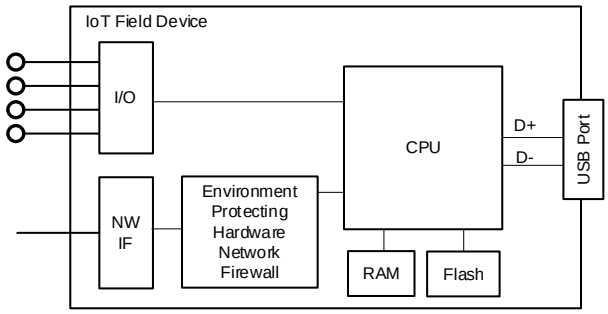


Figure 5. Attack-preventing IoT Device Architecture

Figure 5 shows an IoT Field Device with a central processing unit CPU executing device firmware/software stored in a flash of RAM memory. The software can communicate over the network interface (NW IF) with other devices, e.g., using HTTPS or OPC UA over TCP/IP. Also, sensors and actuators can be connected via an input-output (I/O) interface. An USB interface allows to configure the device or to install a firmware update.

To enhance resilience, the device includes a hardware-based network firewall to protect the network environment from attacks originating from the IoT field device. It limits the type of network communication that can be performed by the device software executed on the CPU. This function is fixed, so that the device software cannot modify it, so that the filtering is performed with high level of trustworthiness. It would be possible as well to use a secure execution environment that is isolated from the main device software executed on its CPU. In both cases, the network firewall is effective even if the main device software executed on the CPU should be manipulated.

The hardware-based firewall can be realized by an integrated circuit, e.g., an application-specific integrated circuit (ASIC), a field programmable gate array (FPGA), or a separate microcontroller or security controller, or it can be integrated with a hardware-based network interface. The filter policy might be adapted, depending on whether a cryptographically protected network access token (NAcT) is provided to the hardware firewall. The NAcT can be provided by a backend device integrity check service. The device software may provide a received NAcT token to the device hardware firewall, but cannot manipulate it. This allows the backend device integrity check service to temporarily activate a less restrictive policy if the device integrity has been verified successfully. A NAcT token can be protected by a cryptographic checksum, e.g., a digital signature (e.g., RSA, DSA, ECDSA) or a symmetric message authentication code (e.g., HMAC, AES-CBC-MAC). The NAcT token realizes an authenticated watchdog, as described by England, Aigner, Marochko, Mattoon, Spiger, and Thom [7]. However, here it is used for selecting a firewall policy, not for initiation a device recovery procedure. If an integrity monitoring system monitoring the integrity of control devices or a network-based intrusion detection system, realizing the device integrity check service, detects an ongoing attack in the IACS, it can limit reliably the network communications of devices, allowing to confine the attack.

A different approach compared to attack monitoring is to monitor write access to the flash memory, i.e., to check whether the device software (firmware) stored in the flash memory is updated regularly. The less restrictive, open filter policy stays activated only if the device firmware is updated regularly.

This section described a hardware network firewall of an IoT device to prevent attacks of a manipulated IoT device via network communications. A related, complementary approach, limiting access to the physical world via the sensors and actuators connected to the input/output interface by manipulated software running on the CPU has been described in previous work [6].

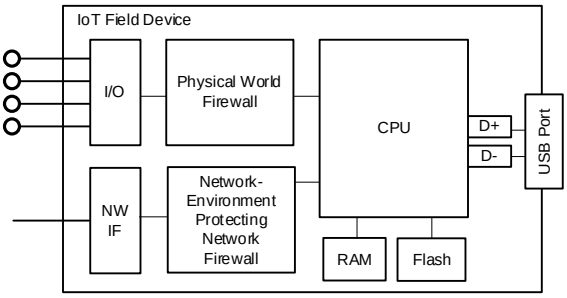


Figure 6. IoT Device Architecture protecting against attacks on both the physical and the cyber environment

Both approaches can be combined in a single IoT device, see Figure 6. The impact of successful attacks on an IoT device on both the network environment as well as on physical world is limited. The possibility that a manipulated IoT device can be used for launching attacks on other connected systems is reduced. While shown as separate entities, both firewall functionalities can also be combined into a single on-device firewall functionality, protecting both access to physical world via I/O interfaces as well as access to other devices by data communication. After having detected a device manipulation by a device integrity check, restrictive filtering policies would be activated in a reliable way by the device so that a manipulated software executed on the CPU has only limited possibilities to perform unwanted sensitive operations impacting the physical world or other IoT devices of the CPS.

VI. EVALUATION

While the original motivation for ”plug and produce”, as defined for Industry 4.0, is to increase flexibility in production and to reduce the time needed to reconfigure an automation environment for different manufacturing tasks or batches, this flexibility is also advantageous for increasing resilience under attack: Even if some of the devices are manipulated (attacked) and cannot be used for production until they are patched, the flexibility of the overall production system allows to reconfigure the IACS components, avoiding or at least limiting the interaction with affected devices. Therefore, production can continue, maybe with limitations, even when some devices should have been manipulated. This means that functionality coming with Industry 4.0 can already be used also to increase resilience under attack, i.e., to improve availability of automation and production systems being under attack. When using the enhancement described in section V, it depends on the specific IACS and on the specific attack scenario to what degree the IACS can stay operational under which specific attack scenario. For the evaluation, it has to be determined to what degree relevant risks of the IACS are reduced by introducing such protection measures.

The security of a CPS is evaluated in practice in various approaches and stages of the system’s lifecycle:

– A Threat and Risk Analysis (TRA, also abbreviated as TARA) is typically conducted at the beginning of the product design or system development, and updated after major design changes, or to address a changed threat landscape. In a TRA, possible attacks (threats) on the system are identified. The impact that would be caused by a successful attack and the probability that the attack happens are evaluated to determine the risk of the identified threats. The risk evaluation allows to prioritize the threats, focusing on the most relevant risks and to define corresponding security measures. Security measures can target to reduce the probability of an attack by preventing it, or by reducing the impact.

– Security checks can be performed during operation or during maintenance windows to determine key performance indicators (e.g., check compliance of device configurations) and to verified that the defined security measures are in fact in place.

– Security testing (penetration testing, also called pentesting for short) can be performed for a system that has been built, but that is currently not in operation. A pentest can usually not be performed on an operational automation and control system, as the pentest could endanger the reliable operation auf the system. Pentesting can be performed during a maintenance window when the physical system is in a safe state, or using a separate test system. Security testing can be performed also on a digital representation of a target system, e.g., a simulation in the easiest case. This digital representation is also called “digital twin”. This allows to perform security checks and pentesting for systems that are not existing yet physically (design phase), or to perform pentesting of operational systems in the digital world without the risk of disturbing the regular operation of the real-world system. Using a dedicated test system or a digital twin simulation environment allows also to install patches and to test their effectiveness and possible influence on the CPS operation without interfering with the operational system.

As long as the technology proposed in the paper has not been proven in a real-world operational setting, it can be evaluated conceptually by analyzing the impact that the additional security measure would have on the identified residual risks as determined by a TRA. The general effect of the presented resilience-under attack security measure is that the worst-case impact of a threat, i.e., a successful attack, on the physical world controlled by the CPS is reduced. Whatever attack is ongoing on the IT-based automation and

control system, still the possible impact on the real, physical world is limited. While security measures often target the prevention of attacks, the proposed resilience measure reduces the impact and thereby the risk. The impact of a threat is reduced if the IACS in fact can stay operational, at least with limited functionality, in relevant attack scenarios. A further, indirect effect of the improved resilience would be that relevant key performance indicators of the CPS, e.g., its uptime/availability, the output of produced goods, the required production time, quality-relevant metrics of produced goods, or the number of deficient goods are maintained even when the CPS is being attacked.

However, TRAs for real-world CPS are not available publicly. Nevertheless, an illustrative example may be given by a chemical production plant performing a specific process like refinery, or a factory producing glue or cement. If the plant is attacked, the attack may target to destroy the production equipment by immediately stopping the process leading to physical hardening of the chemicals / consumables and thus to a permanent unavailability of the production equipment. In this case, trusted sensors could be used to detect a falsified sensor signal, and the physical-world firewall can be used to limit actions in the physical world. Both, the trusted sensors and the physical world firewall build a security overlay network, independent from the actual operational control network. Thereby, a physical damage of the production equipment can be avoided. If needed, a controlled shutdown of the production site can be performed.

Threat	Likelihood	Impact	Risk
Device communication intercepted	unlikely	moderate	minor
Device communication manipulated	unlikely	critical	moderate
Vulnerability in unpatched device exploited	likely	critical	major
Device replaced by fake device	possible	moderate	moderate
⋮	⋮	⋮	⋮

Figure 7. Example Threats of a Threat and Risk Analysis

Figure 7 shows a simplified table as used typically in a threat and risk analysis to collect and evaluate relevant treats to a technical system or component. Some selected threats are shown as examples. Realistic TRAs for real-world systems and components include usually a much longer list of threats. The likelihood and the impact of the threat is determined by judgement of competent personal, usually in a team including technical experts and people responsible for the product or system. The corresponding risk is determined based on likelihood and impact. It has shown to be useful to define and document explicitly the criteria leading to the categorization of likelihood and impact, including also the made assumptions on the operational environment. The TRA with prioritized risks is the basis for security design decisions, focusing on the most critical risks. It is the basis to define a security concept that includes suitable protection measures. Protection measures may not be technical measures only, but include as well organizational and personal security measures (e.g., performing regularly security audits and security trainings).

Figure 8 shows how the likelihood and the impact are mapped to the corresponding risk value. In the example, the three categories unlikely, possible, and likely are used to describe the likelihood. For the impact, the three categories negligible, moderate, and critical are used. In practice, also more fine-granular rankings can be used, distinguishing, e.g., four or five different categories. Also, the risk evaluation can in general include further categories, e.g., disastrous.

For the example threats shown in Figure 7, the risk that the device communication is intercepted is evaluated as minor, as the assumption in the example is that the device communication is protected cryptographically (e.g., by the Transport Layer Security protocol TLS [42]), and that the data would not reveal highly sensitive information.

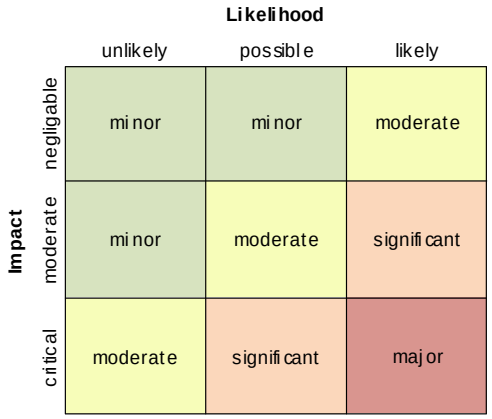


Figure 8. Risk Mapping

The risk that the communication is manipulated, leading to a manipulated device operation, is unlikely as well as the communication is assumed to be protected in transit.

However, the impact is evaluated as critical, as, without any further protection, this threat could lead to arbitrary effects on the device operation and therefore also on the CPS. The risk that a vulnerability of the software running on the device is exploited is ranked here as major, as the assumption is that the device is not regularly patched while being connected to the public Internet. Therefore, it is likely that the vulnerability will be exploited. The functionality of the manipulated device could be changed in arbitrary ways, so that the impact can be critical, leading to a major risk. The threat that the device is replaced by a fake device is evaluated as moderate.

An overview on the determined risks can be shown in a graphical risk reporting as shown in Figure 9. It gives an easily understandable representation on the distribution of identified risks. This representation can be useful to depict the overall risk exposure of a CPS if many risks have been identified. In particular, the example shows one major threat (red field) as well as a moderate threat (yellow field) having a critical impact. Butting resilience measures as the one described in section V into place can reduce the impact of threats, thereby improving the overall risk exposure.

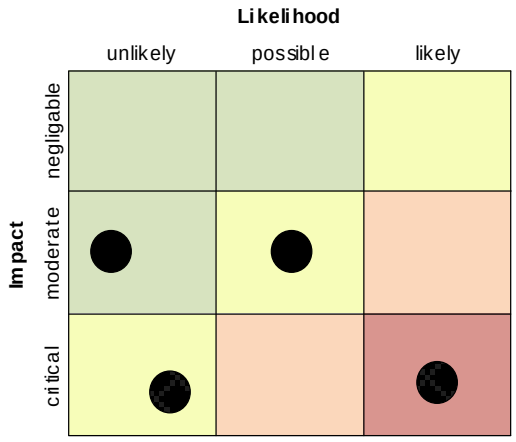


Figure 9. Risk Reporting for the Example Threats without Resilience-Under-Attack Protection

If the resilience under attack protection as described in section IV is put into place, the possible impact of the threats is reduced. This effect is illustrated in Figure 10. As, in the example shown, the impact of the two risks with critical impact reduces from critical to moderate, the risk is reduced correspondingly. Thereby, also the overall risk situation of the overall CPS in which the considered device is used, is improved.

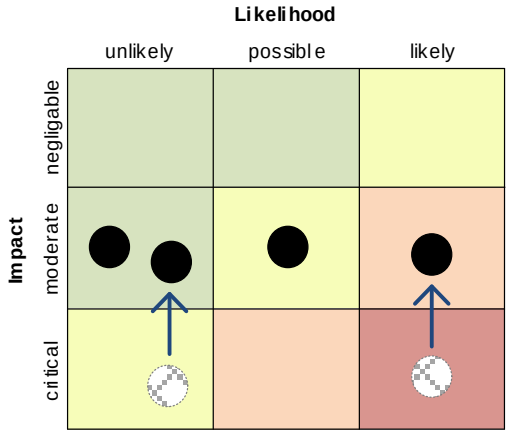


Figure 10. Risk Reporting for the Example Threats with Resilience-Under-Attack Protection

As the evaluation in a real-world CPS requires significant effort, and as attack scenarios cannot be tested that could really have a (severe) impact on the physical world, a simulation-based approach or using specific testbeds are possible approaches, allowing to simulate the effect on the physical world of certain attack scenarios with compromised components in a simulation model of the CPS, or to evaluate it in a protected testbed, e.g., a CPS test system. The simulation would have to include not only the IT-based control function, but also the physical world impact of an attack. Using physical-world simulation and test beds to evaluate the impact of attacks have been described by Urbina, Giraldo et al. [43]. They allow to analyze the impact of successful attacks on the physical world in a safe evaluation environment.

VII. CONCLUSION

A CPS comprises the operational cyber-technology and the physical world with which the system interacts. Both parts have to be covered by a security concept and solution. Traditional cyber security puts the focus on the cyber-part, i.e., automation and control systems. The security of the physical part, like machinery, is protected often by physical and organizational security measures, only. This paper presented a concept for a new approach that enhances the resilience of a CPS in the presence of attacked devices, by making it harder that a compromised CPS device is used for attacking other devices of the CPS. This can be a useful element to ensure the availability of the automation system. Even under attack, the automation system has not to be shut down completely. It can stay operational, possibly with reduced performance or functionality. It is complementary to other approaches for enhancing CPS resilience by protecting the physical-world interface [5] as well as to platform resilience measures as known from [28] [29] that allow to recover a manipulated device quickly and reliably back into a well-defined state.

Possible future work is to not only analyze the effect on reducing the risk exposure of a complex real-world CPS, but to determine the effect of successful attacks on relevant operational key performance indicators of the CPS, as, e.g., uptime and output of a production system. A CPS simulation environment, i.e., a digital twin of the CPS, or a non-operational CPS testbed can be used to analyze the impact of different attack scenarios on CPS key performance indicators that are relevant to the operation of the CPS in a safe way. Such analysis is considered to be relevant in particular for CPS, as attacks may impact not only IT-related assets, but may have also an impact on both the real, physical world of the controlled technical system and on the business-relevant operation of the CPS.

REFERENCES

[1] R. Falk and S. Fries, “Enhancing Attack Resilience in the Presence of Manipulated IoT Devices within a Cyber Physical System”, The Sixth International Conference on Cyber-Technologies and Cyber-Systems CYBER 2021, pp. 1-6, October 03, 2021 to October 07, 2021 - Barcelona, Spain, [Online]. Available from https://www.thinkmind.org/index.php?view=article&articleid=cyber_2021_1_20_80046 [retrieved March, 2023]

[2] N. N. Dao, T. V. Phan, U. Sa’ad, J. Kim, T. Bauschert, and S. Cho, “Securing Heterogeneous IoT with Intelligent DDoS Attack Behavior Learning”, arXiv: 1711.06041v3 [cs.NI] 7 Aug 2019, [Online]. Available from: https://arxiv.org/pdf/1711.06041.pdf [retrieved March, 2023]

[3] Shodan - Search engine for the Internet of everything, [Online], https://www.shodan.io/, [retrieved March, 2023]

[4] Morpheus – Network Security Scanner, [Online], https://www.morpheus.com.na/, [retrieved March, 2023]

[5] R. Falk and S. Fries, “Enhancing Resilience by Protecting the Physical-World Interface of Cyber-Physical Systems”, The Fourth International Conference on Cyber-Technologies and Cyber-Systems CYBER 2019, pp. 6–11, September 22, 2019 to September 26, 2019 - Porto, Portugal, [Online]. Available from: https://www.thinkmind.org/index.php?view=article&articleid=cyber_2019_1_20_80033 [retrieved March, 2023]

[6] R. Falk and S. Fries, “Enhancing the Resilience of Cyber-Physical Systems by Protecting the Physical-World Interface”, International Journal On Advances in Security, volume 13, numbers 1 and 2, pp. 54-65, 2020, [Online]. Available from: http://www.thinkmind.org/index.php?view=article&articleid=sec_v13_n12_2020_5 [retrieved March, 2023]

[7] P. England, R. Aigner, A. Marochko, D. Mattoon, R. Spiger, and S. Thom, “Cyber resilient platforms”, Microsoft Technical Report MSR-TR-2017-40, Sep. 2017, [Online]. Available from: https://www.microsoft.com/en-us/research/publication/cyber-resilient-platforms-overview/ [retrieved March, 2023]

[8] Electronic Communications Resilience&Response Group, “EC-RRG resilience guidelines for providers of critical national telecommunications infrastructure”, version 0.7, March 2008, available from: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/62281/telecoms-ecrrg-resilience-guidelines.pdf [retrieved March, 2023]

[9] D. Urbina, J. Giraldo, N. O. Tippenhauer, and A. Cardenas, “Attacking fieldbus communications in ICS: applications to the SWaT testbed”, Singapore Cyber-Security Conference (SG-CRC), IOS press, pp. 75–89, 2016, [Online]. Available from: http://ebooks.iospress.nl/volumearticle/42054 [retrieved March, 2023]

[10] C. C. Davidson, T. R. Andel, M. Yampolskiy, J. T. McDonald, W. B. Glisson, and T. Thomas, “On SCADA PLC and fieldbus cyber security”, 13th International Conference on Cyber Warfare and Security, National Defense University, Washington, DC, pp. 140–148, 2018

[11] D. Bodeau and R. Graubart, “Cyber resiliency design principles”, MITRE Technical Report, January 2017, [Online]. Available from: https://www.mitre.org/sites/default/files/publications/PR%20170103%20Cyber%20Resiliency%20Design%20Principles%20MTR17001.pdf [retrieved March, 2023]

[12] A. Kott and I. Linkov (Eds.), “Cyber Resilience of Systems and Networks”, Springer, 2019

[13] R. Falk and S. Fries, “Enhancing integrity protection for industrial cyber physical systems”, The Second International Conference on Cyber-Technologies and Cyber-Systems, CYBER 2017, pp. 35–40, November 12 - 16, 2017, Barcelona, Spain, [Online]. Available from: http://www.thinkmind.org/index.php?view=article&articleid=cyber_2017_3_30_80031 [retrieved March, 2023]

[14] European Commission, “The directive on security of network and information systems (NIS Directive)”, [Online]. Available from: https://ec.europa.eu/digital-single-market/en/network-and-information-security-nis-directive [retrieved March, 2023]

[15] IEC 62443, “Industrial automation and control system security” (formerly ISA99), [Online]. Available from: https://webstore.iec.ch/searchform&q=62443 [retrieved March, 2023]

[16] ISO/IEC 27001, “Information technology – security techniques – Information security management systems – requirements”, October 2013, [Online]. Available from: https://www.iso.org/standard/54534.html [retrieved March, 2023]

[17] NIST, “Framework for Improving Critical Infrastructure Cybersecurity”, Version 1.1, April 16, 2018, [Online]. Available from: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf [retrieved March, 2023]

[18] IEC 62443-3-3:2013, “Industrial communication networks – network and system security – Part 3-3: System security requirements and security levels”, Edition 1.0, August 2013

[19] IEC 62443-4.2, “Industrial communication networks - security for industrial automation and control systems - Part 4-2: technical security requirements for IACS components”, February 2019

[20] EN 303 645, “Cyber Security for Consumer Internet of Things: Baseline Requirements”, ETSI, V2.1.1 (2020-06), June 2020, [Online]. Available from: https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.01_60/en_303645v020101p.pdf [retrieved March, 2023]

[21] TS 103 701, “Cyber Security for Consumer Internet of Things: Conformance Assessment of Baseline Requirements”, ETSI, V1.1.1 (2021-08), August 2021, [Online]. Available from: https://www.etsi.org/deliver/etsi_ts/103700_103799/103701/01.01.01_60/ts_103701v010101p.pdf [retrieved March, 2023]

[22] TR 103 621, ” Guide to Cyber Security for Consumer Internet of Things”, ETSI, Draft 0.0.6 (2021-06), June 2021.

[23] M. Fagan, J. Marron, K. Brady, B. Cuthill, K. Megas, R. Herold, D. Lemire, B. Hoehn, “IoT Device Cybersecurity Guidance for the Federal Government: Establishing IoT Device Cybersecurity Requirements”, NIST SP 800-213, November 2021, [Online]. Available from: https://csrc.nist.gov/publications/detail/sp/800-213/final [retrieved March, 2023]

[24] M. Fagan, K. Megas, J. Marron, K. Brady, B. Cuthill, R. Herold, D. Lemire, B. Hoehn, “IoT Device Cybersecurity Guidance for the Federal Government: IoT Device Cybersecurity Requirement Catalog”, NIST SP 800-213A, November 2021, [Online]. Available from: https://csrc.nist.gov/publications/detail/sp/800-213a/final [retrieved March, 2023]

[25] NIST, “DRAFT Baseline Security Criteria for Consumer IoT Devices”, NIST, August 31, 2021, [Online]. Available from: https://www.nist.gov/system/files/documents/2021/08/31/IoT%20White%20Paper%20-%20Final%202021-08-31.pdf [retrieved March, 2023]

[26] NIST, “Trustworthy Network of Things”, December 15 , 2020, [Online]. Available from: https://www.nist.gov/programs-projects/trustworthy-networks-things [retrieved March, 2023]

[27] E. Lear, R. Droms, D. Romascanu, “Manufacturer Usage Description Specification”, Internet Request for Comments, RFC8520, March, 2019, [Online]. Available from: https://www.rfc-editor.org/rfc/rfc8520.html [retrieved March, 2023]

[28] A. Regenscheid, “Platform Firmware Resiliency Guidelines”, NIST SP 800-193, May, 2018, [Online]. Available from: https://csrc.nist.gov/publications/detail/sp/800-193/final [retrieved March, 2023]

[29] TCG, “Cyber Resilient Module and Building Block Requirements”, V1.0, October 19, 2021, [Online]. Available from: https://trustedcomputinggroup.org/wp-content/uploads/TCG_CyRes_CRMBBReqs_v1_r08_13jan2021.pdf [retrieved March, 2023]

[30] P. England, R. Aigner, A. Marochko, D. Mattoon, R. Spiger, and S. Thom, “Cyber resilient platforms”, Microsoft Technical Report MSR-TR-2017-40, Sep. 2017, [Online]. Available from: https://www.microsoft.com/en-us/research/publication/cyber-resilient-platforms-overview/ [retrieved March, 2023]

[31] Electronic Communications Resilience&Response Group, “EC-RRG resilience guidelines for providers of critical national telecommunications infrastructure”, version 0.7, March 2008, available from: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/62281/telecoms-ecrrg-resilience-guidelines.pdf [retrieved March, 2023]

[32] D. Bodeau and R. Graubart, “Cyber resiliency design principles”, MITRE Technical Report, January 2017, [Online]. Available from: https://www.mitre.org/sites/default/files/publications/PR%2017-

0103%20Cyber%20Resiliency%20Design%20Principles%20MTR17001.pdf [retrieved March, 2023]

[33] A. Kott and I. Linkov (Eds.), "Cyber Resilience of Systems and Networks", Springer, 2019

[34] P. Bock, J. P. Hauet, R. Françoise, and R. Foley, "Ukrainian power grids cyberattack - A forensic analysis based on ISA/IEC 62443", ISA InTech magazine, 2017, [Online]. Available from: https://www.isa.org/intech-home/2017/march-april/features/ukrainian-power-grids-cyberattack [retrieved March, 2023]

[35] ZVEI, "Orientation guideline for manufacturers on IEC 62443", "Orientierungsleitfaden für Hersteller zur IEC 62443" [German], ZVEI Whitepaper, 2017, [Online]. Available from: https://www.zvei.org/presse-medien/publikationen/orientierungsleitfaden-fuer-hersteller-zur-iec-62443/ [retrieved March, 2023]

[36] H. R. Ghaeini, M. Chan, R. Bahmani, F. Brasser, L. Garcia, J. Zhou, A. R. Sadeghi, N. O. Tippenhauer, and S. Zonouz, "PAtt: Physics-based Attestation of Control Systems", 22nd International Symposium on Research in Attacks, Intrusions and Defenses, USENIX, pp. 165–180, September 23-25, 2019, [Online]. Available from: https://www.usenix.org/system/files/raid2019-ghaeini.pdf [retrieved March, 2023]

[37] Plattform Industrie 4.0, "Industrie 4.0 Plug-and-produce for adaptable factories: example use case definition, models, and implementation", Plattform Industrie 4.0 working paper, June 2017, [Online]. Available from: https://www.zvei.org/fileadmin/user_upload/Presse_und_Medien/Publikationen/2017/Juni/Industrie_4.0_Plug_and_produce/Industrie-4.0-_Plug-and-Produce-zvei.pdf [retrieved March, 2023]

[38] T. Hupperich, H. Hosseini, and T. Holz, "Leveraging sensor fingerprinting for mobile device authentication", International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, LNCS 9721, Springer, pp. 377–396, 2016, [Online]. Available from: https://www.syssec.ruhr-uni-bochum.de/media/emma/veroeffentlichungen/2016/09/28/paper.pdf [retrieved March, 2023]

[39] H. Bojinov, D. Boneh, Y. Michalevsky, and G. Nakibly, "Mobile device identification via sensor fingerprinting", arXiv:1408.1416, 2016, [Online]. Available from: https://arxiv.org/abs/1408.1416 [retrieved March, 2023]

[40] P. Hao, "Wireless device authentication techniques using physical-layer device fingerprint", PhD thesis, University of Western Ontario, Electronic Thesis and Dissertation Repository, 3440, 2015, [Online]. Available from: https://ir.lib.uwo.ca/etd/3440 [retrieved March, 2023]

[41] R. Falk and M. Trommer, "Integrated Management of Network and Host Based Security Mechanisms", 3rd Australasian Conference on Information Security and Privacy, ACISP98, pp. 36-47, July 13-15, 1998, LNCS 1438, Springer, 1998

[42] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", Internet RFC8446, August 2018, [Online]. Available from: https://datatracker.ietf.org/doc/html/rfc8446 [retrieved March, 2023]

[43] D. Urbina, J. Giraldo, A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg, "Limiting The Impact of Stealthy Attacks on Industrial Control Systems", ACM Conference on Computer and Communications Security (CCS), pp. 1092–1105, Vienna, Austria, 2016

Adapting to Change: A Study of the Software Architecture Evolution of a Physical Security Information Management System

Oğuzhan Özçelik
ASELSAN A.Ş.
Ankara, Turkey
e-mail: oozcelik@aselsan.com.tr

Halit Oğuztüzün
Department of Computer Engineering
Middle East Technical University
Ankara, Turkey
e-mail: oguztuzn@ceng.metu.edu.tr

***Abstract*—Physical Security Information Management (PSIM) system customizations tend to be similar to each other with core requirements being more or less the same in different projects. One of the most common differences in these projects is the sensors being used. Some sensors could be integrated into the PSIM system easily if they are compatible with a standard communication interface such as Open Network Video Interface Forum (ONVIF) protocols. But sensors that use a special communication interface need to be integrated one by one. A PSIM system is always expected to integrate additional sensors to its inventory. In order to do this easily, the modules that need to be developed to integrate a sensor must be segregated and developed individually for each sensor. These modules can be seen as features to be used in a software product line architecture. The planned reuse mentality of software product line engineering makes it possible to deliver similar products within a short amount of time. In this work, we aim to segregate the sensor integration of a PSIM system and compare the old and new generations of the architecture both qualitatively, based on their architecture models, and quantitatively, based on test results. Several tests and surveys have conducted in order to inspect the new architecture's performance.**

Keywords-Physical Security Information Management Systems; Physical Protection Systems; Software Product Line Engineering.

I. INTRODUCTION

A Physical Security Information Management (PSIM) system integrates diverse independent physical security applications and devices. Applications such as building management or network video recorder systems, and devices such as security cameras, access control systems, radars and plate recognition systems are used interconnectedly through a centralized platform. It is designed to ensure the physical security of a facility, city or an open field, while providing a complete user interface to the security operators to monitor and control them. With the help of PSIM systems, security personnel can make prompt decisions about a security situation by investigating the comprehensive picture the PSIM system generated with the data that it gathered, associated and analyzed.

This work is a continuation of our previous work [1]. We have conducted a survey to see the problem, and the gains we achieve with this architectural change better. And we have tested the old and new architecture in order to see whether the new architecture comes with a performance loss.

Physical Protection System (PPS) is also a common term to refer to such a system. Mary Lynn Garcia described the PPS functions, which can be seen in Figure 1, in three main categories: detection, delay, response [2]. Detection is the discovery of a malevolent incident. Measuring the threat level of an action would also be beneficial while deciding the following functions' extents. This measurement must provide information about the importance of detection and if it is important, every detail about the cause of the alarm. The level of detail is primarily based on the type of sensor that detected the alarm. Next function of a PPS is delay. After the adversary action got detected, the first thing to do is delaying its operations. This can be accomplished by locks, barriers or security personnel in the perimeter. The reason for this function is basically stalling the adversary in order to gain time for the next function, response. Response is the cumulative actions taken by the security personnel or system, in order to prevent adversary action.

The subject PSIM system of this work is called SecureX, which is not the name of the actual system, but a placeholder used for confidentiality reasons. SecureX is a PSIM system that aims to satisfy the needs mentioned above and to provide an easy integration environment for new sensors and applications. The ever-increasing number of such new systems and particular security needs of different customers drove SecureX team to embrace a software product line engineering approach in order to reduce the response time to reply to the customers' demands. These demands vary from

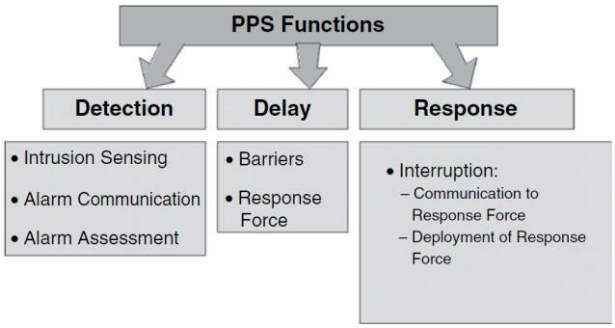


Figure 1. Functions of a Physical Protection System [2].

practical improvements to integrating a new sensor or security application as a feature to the system. SecureX is deployed with the full feature set and only at runtime these features are reduced to the ones required by a given customer, using different configuration files. Any new integration required by a customer needs to be developed as a feature in SecureX. Afterwards, a new SecureX build must be generated. Following every new integration, a new testing process takes place and because the previously integrated system might not always be available for testing, it must be guaranteed that the new integration will not affect the previously completed integrations. In this work, a new method for integrating such new systems while reducing the number of required tests is proposed. The new method is also going to be an evolutionary step toward a product line architecture.

The rest of the paper is structured as follows. In Section II, several PSIM products and their specializations are mentioned. Also, we briefly explain how they approach the sensor integration problem and why that is not enough in the case of SecureX. In Section III, the general architecture of SecureX is described and the point where sensor integration takes place is shown. Also, the technology that will be used is described. In Section IV, this sensor integration point is described in more detail. In Section V, the problems with the current architecture are explained and in Section VI, a new architecture that solves those problems is described. In Section VII, results of a survey and performance tests are detailed. In Section VIII, the benefits of the new architecture are shown by further explaining how it solves each problem of the current design.

II. Related Works

There are several companies offering PSIM products. Although they provide every essential feature of a PSIM system, they may have different specializations. Some companies are more promident in video management systems and some in geographic information systems. Plate recognition and access control systems are also fields in which a PSIM system can be used. In the SecureX's case, all four types of systems mentioned now can be used together.

Genetec [3] provides a video analytics tool to detect intrusions. They also develop access control systems and use plate recognition systems to monitor vehicles. Milestone [4] uses its own Network Video Recorder (NVR) system and provides an easy-to-use video management system. They work with numerous different companies and provide an easy integration framework to work with them. Nedap [5] is specialized in access control systems and they work with other companies like Genetec and Milestone to get integrated in their PSIM systems as well. However, not many details exist on how they work internally. These products integrate some general communication standards like ONVIF [6] protocols and also release Software Development Kits (SDK) and expect sensor manufacturers or customers to integrate their custom subsystems into the PSIM system as well. This way, they accelerate sensor integration by including numerous 3rd parties. While developing an SDK to use in integrations is a feasible solution, in the SecureX's case, the main objective is developing an architecture that can simplify not only the sensor integrations, but also the component selection to deploy because different customers have different requirements. Another requirement is that the new architecture will be able to remove the update and test overhead. A software product line architecture would be suitable to accomplish this goal.

Recently, Tekinerdogan et al. [7] described how a PSIM system should be designed with software product line engineering methodologies to reduce the cost of development by improving reuse. The present work describes a step in architectural evolution toward a product line architecture.

In different programming languages, there are many frameworks in which a software product line could be implemented. One specific technology that has the software product line implementation capability is called Open Services Gateway Technology (OSGi) framework for Java [8]. Its details will be explained in the coming chapters, but its abilities are shown by Almeida, E. et al [9]. In their work, they tried to provide a method that can be used in the domain implementation phase of software product lines. They conducted experiments using a pilot project in order to investigate the feasibility of their method. Seven M.Sc. students with industrial software development experience are selected and after a short training, the participants were expected to complete the tasks assigned to them. After the project had been completed, the quantitative analyses showed that the method is beneficial in developing software components with high maintainability while lowering the overall complexity. The participants also got surveyed and their answers indicated that the method provides useful guidance, thanks to the OSGi, a technology which is very suitable to be used in software product line development. But subjects without experience with this technology noted that they had challenges using it. However, these challenges are nothing that training cannot be overcome. Overall, their experimental study showed that using a software product line architecture with OSGi helps developers to build products with better quality.

III. Architecture of SecureX

SecureX is a PSIM system that is used in a wide variety of fields from border or airport security to protecting various critical facilities and oil or gas pipelines with special sensors. In some projects, the system is used in low performance computers and tablets while some projects use high performance servers. Some projects requires a dozen sensors to protect a small remote location and some uses thousands of sensors in a highly concentrated manner inside a city. Some projects are a combinations of those. Houndreds of small, secure facilities with dozens of sensors each, connected hierarchically to each other and at the top, controlled by high authority security officers. These different projects comes with different requirements from both public and private instutions. SecureX has to be able to adapt the different needs of each customer. This need

caused SecureX to be a highly configurable system that is tailored for every new customer and project.

SecureX has a distributed architecture which can be seen in Figure 2. Graphical User Interface (GUI) Clients of SecureX are installed on the computers of security officers, enabling them to monitor the entire security infrastructure of the area under surveillance. These clients are connected to the SecureX Server application which handles the communication between SecureX components. The server is also responsible for recording events, including detections and errors sent from adapter components to the central database. SecureX could also be installed in a hierarchical fashion in which higher servers could also control and monitor the security components that are connected to the servers under them. Under the SecureX Server, there are adapter applications for each sensor group such as camera, radar, plate recognition systems, access control systems, etc. These adapters are the points where the SecureX environment makes its connections to the outer world.

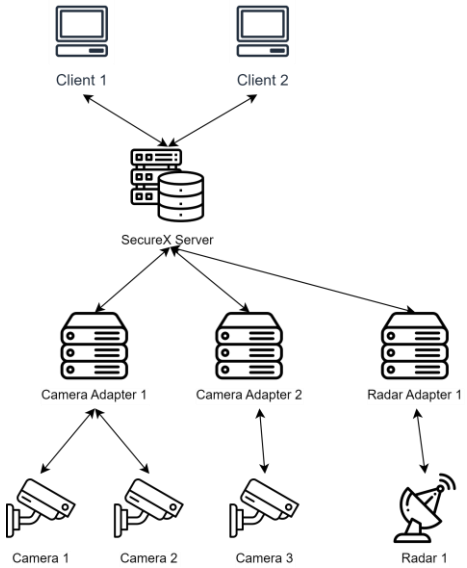


Figure 2. Deployment model of SecureX.

When a user wants to perform some action with a sensor, after pressing a button in the SecureX GUI Client, a message will be sent to the SecureX Server. Then, the server delegates this message to the adapters and other servers that are hierarchically under that server. The message arrives at the sensor's adapter and, according to the Interface Control Document (ICD) used in its integration, a message would be sent to the sensor to perform the desired action. Events and detections caught by the sensors would follow the reverse route and find their way to the SecureX GUI Clients.

SecureX is developed using the OSGi framework, which is a Java framework to develop modular software [10]. It is a platform in which manufacturers and developers can use as a software component framework. It is a versatile deployment API that can manage the life cycle of applications.

A. OSGi

The OSGi framework, based on its specifications, is a framework that can be used for creating highly modular Java systems. With its component model, it is a very reasonable candidate to be used in software product line development. It provides a simple way to change software components not only without a need to rebuild the entire system, but also dynamically changing them at the runtime. This shows the main capability of OSGi that simplifies the development of variation points, which is a crucial aspect of software product line architecture. The components are called "bundles" in the OSGi world, and the framework provides methods for installing, uninstalling and updating those bundles [11]. The life cycle that each bundle undergo in the OSGi framework can be seen in the Figure 3.

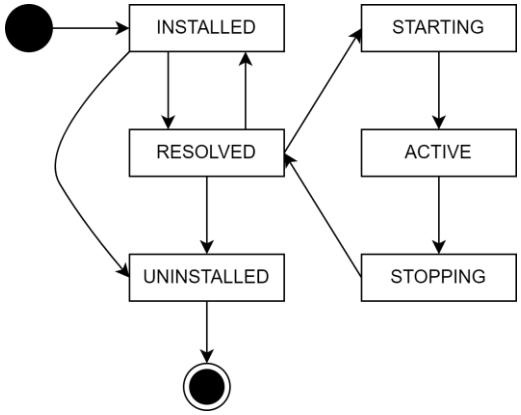


Figure 3. OSGi bundle life cycle. [11]

Every application that runs on the OSGi framework is expected to be able to immediately respond to the component changes at runtime. Any component might get an update or gets installed at runtime and the application that uses the component must properly react to this change and migrate to the new component. OSGi is a dynamic environment that expects applications to catch up to its changes.

Every bundle in an OSGi application has a start level that is defined in the bundle configuration files. When an application that runs on the OSGi framework starts, its bundles get initialized in the order of their start levels. SecureX uses this ordered initialization procedure and runtime bundle installation capabilities to optimize its initialization time by only installing the key bundles at first and installing the remaining bundles at the runtime.

IV. Existing Adapter Architecture

There are several adapter applications developed for different types of sensors such as Camera Adapter, Radar Adapter or Seismic Adapter etc. Their working principles are quite similar. The SecureX Server connects to the adapters and the adapter connects to the sensors. To segregate the sensor integration, we must first analyze the existing adapter architecture.

A few of the bundles in the Camera Adapter program can be seen in Figure 4. SecureX uses this framework to take advantage of its service architecture. We use the Camera Adapter application to describe the adapter architecture, but all adapter applications of SecureX are quite similar.

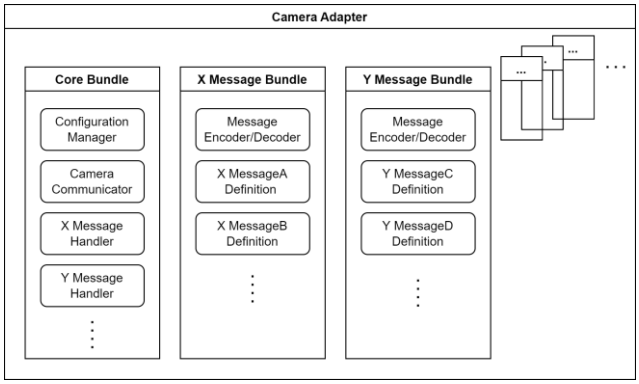


Figure 4. Simplified Camera Adapter model in the existing architecture.

The Camera Adapter application consists of many OSGi bundles whose purposes vary from providing network connection interfaces or utility tools, to message definition of sensors. These message definition bundles contain the methods for encoding and decoding messages to and from the sensor. Generally, the message formats for each sensor are different. They have different data types, header types, checksum calculation methods, big or little endian formats. Some sensors accept JSON formatted string messages, and some require encoding messages in a certain length byte array and sending them. Information about how to communicate with a sensor is given in its ICD. A message bundle is basically an implementation of the related ICD.

The *Configuration Manager* class in the *Core* bundle is mainly responsible for opening a Transmission Control Protocol (TCP) port to accept incoming server connections and initializing the *Message Handlers*. Each sensor's type, model, unique identifier key and required information about establishing a connection to it is written in a configuration XML file. The *Configuration Manager* constantly iterates over these files, creating a *Camera Communicator* and a specific *Message Handler* for every new or updated file. Messages are received by the TCP server and forwarded from there to the *Camera Communicator* and lastly to the sensor's *Message Handler*.

A *Camera Communicator*, which extends from the *Sensor Communicator* class as in every other sensor family, is the class where the processing of messages that came from the server starts. It handles generic messages or preprocesses them before the messages arrive at the *Message Handler*. When a message is received from the server, it is added to the message buffer of every active *Camera Communicator* in that adapter. *Camera Communicators* takes this message and decide if this message is meant for their sensor. To do this, they use the sensor identifiers in the messages. If the identifier is the same as the *Message Handler* they have, the message gets processed as will be explained in the subsequent paragraph, otherwise it is discarded.

The processing of the messages starts at the *Camera Communicator* level. Some messages are not specific to different sensor integrations and can be handled at the *Camera Communicator* level. Alternatively, some messages require a preprocessing step such as transforming some variables before they get forwarded to the *Message Handler*. After the initial processing is done, the *Camera Communicator* sends the message to the *Message Handler*.

The *Message Handler* is where the connection to the sensor is established using the protocol the sensor uses, which could be TCP, User Datagram Protocol (UDP), WebSocket, serial port, (Representational State Transfer) REST or any other network connection method that is stated in its ICD. The *Message Handler* knows how the connection should be established and how the incoming and outgoing messages should be processed. It receives the incoming message from the communicator and sends necessary commands to the sensor. The *Message Handler* needs a utility bundle to do the message conversions. When it needs to encode/decode messages to/from the sensor, it uses the Message bundle of that sensor that contains the message types, formats, checksum methods and the information of exactly how a message should be generated. After a message is generated, the *Message Handler* sends it to the sensor using the connection interface.

V. The Integration Problem

To keep up with the new and updated sensors to be integrated, and changing customer needs regarding sensor types and capabilities, sensor integration must be segregated and can be developed and updated independently. After analyzing the adapter architecture in the previous chapter, we can focus on what makes it difficult to integrate sensors in the current architecture.

When the adapter starts, the *StartLevelEventDispatcher* thread in the OSGi framework initializes all bundles that are marked for auto-start in the bundle configuration file. In Figure 5, initialization of the *Core* bundle is shown. The *Core* bundle is the one that starts the main Camera Adapter process with its thread "*ConfigurationMonitor*". In the initialization of the *Core* bundle, a single *Configuration Manager* instance gets created. The *Configuration Manager* then opens a port to listen to incoming SecureX Server connections. After that, it starts a thread that periodically checks sensor configuration files to find new or updated configurations. If there is such a file, then the *Configuration Manager* creates a *Camera Communicator* and the *Message Handler* for that sensor. In the existing architecture, in order to create a *Message Handler* instance, the *Configuration Manager* has to know which *Message Handler* needs to be used for which sensor configuration. In the configuration file, the identifier of the correct *Message Handler* is given, and the *Configuration Manager* uses that identifier to construct the *Message Handler*. But these *Message Handler* classes are inside the *Core* bundle and the *Configuration Manager* has a class dependency for them. This is the root problem in the current architecture.

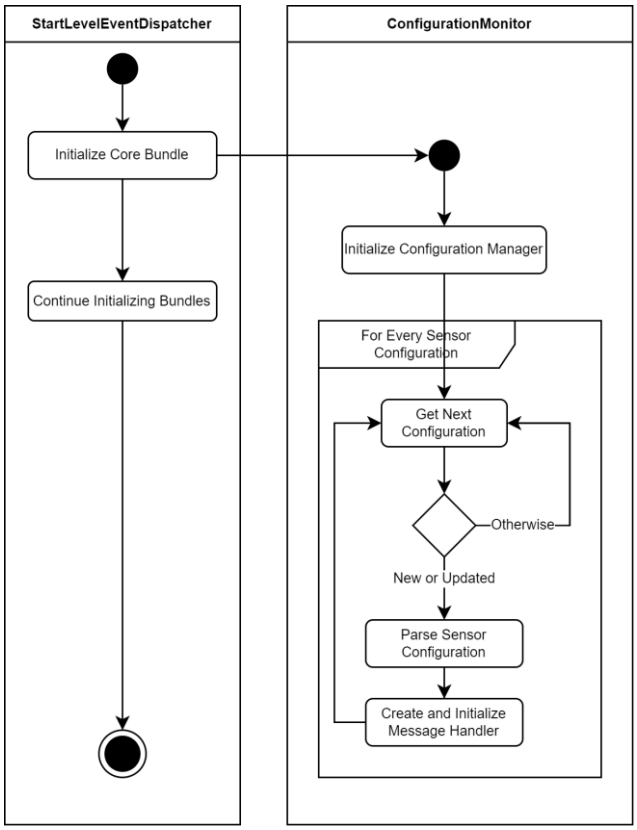


Figure 5. Message Handler initialization in the existing architecture.

A. *Difficulties with the Existing Architecture*

In order to carry out a new sensor integration, the message definition bundle has to be added in the Camera Adapter product file and its *Message Handler* has to be included in the *Core* bundle. The *Configuration Manager* class needs to know with which configuration identifier the new *Message Handler* should be constructed beforehand, hence the dependency. Because of this design, integrating or updating the integration of a sensor requires updating the *Core* bundle in the adapter. The components in the *Core* bundle, such as *Configuration Manager* and *Camera Communicator*, are used in every *Message Handler* and need to be compatible with all of them too. Therefore, any change in those components in the integration of a sensor could affect the already integrated sensors and cause them not to function as intended. Alarms detected by the sensor might start not to be forwarded to the server or changing the orientation of the sensor becomes difficult because of a change in some movement speed calculations.

In the current design, to update an already deployed system, a complete new build needs to be generated and tested. But the regression testing of the previous sensor integrations is not always easy or even possible. These sensors could be produced in very limited numbers, and they can only be found in the customer's facilities, working with the previous SecureX version. The location of these facilities might be difficult to access too and trips to these locations are not only costly, but sometimes, also dangerous. Because these sensors are almost always used in closed networks, the only way to test them is by going to these facilities, increasing the cost of testing. Also, customers would not want testers to separate these sensors from the PSIM system to test with the new version, creating a window of vulnerability.

Even if the tests are somehow completed, the update procedure has its own problems. To quickly update systems used in remote locations with little to no network access, or used in thousands of mobile locations without stable internet access, the update size must be minimal. But, with the current architecture, the whole adapter build needs to be updated, rather than just a couple of bundles.

Also, to catch up with new and updated sensors or security systems, 3rd party companies are employed for integrations. But this process is done through signing a Non-Disclosure Agreement (NDA) and sharing substantial parts of the adapter code with them to be used to integrate the sensors. Any one of them could expose the code at any point and this indeed is a security vulnerability.

Because of these reasons, there is a need for an architecture that ensures that the new integrations will not affect the existing ones. The main problem with the current design is, for every new integration, it has a need to update the *Core* bundle. The reason for that is the *Configuration Manager* class needs to know all available *Message Handlers* and for what kind of sensor they need to be used beforehand via class dependencies. In the new architecture, this problem is targeted with the aim to reduce testing overhead, reducing the amount of code that is shared with 3rd parties and also enables updating the deployed systems with small amount of data.

VI. New Adapter Architecture

To solve the problems with the existing architecture, a new adapter architecture shown in Figure 6 is developed. With this new architecture, all *Message Handler* classes moved to their message definition bundles and an OSGi service called *IMessage Handler Provider Service* that

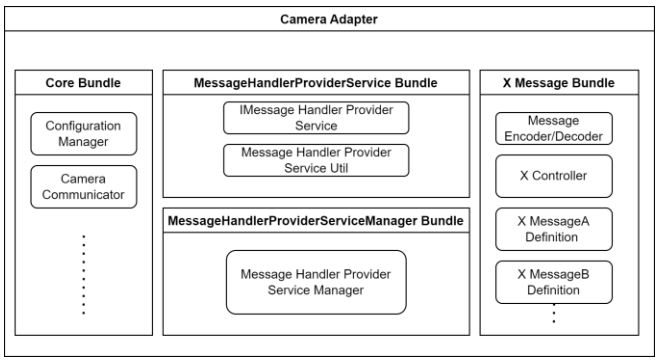


Figure 6. Simplified Camera Adapter model in the new architecture.

provides a *Message Handler* constructor for a given configuration identifier is developed. With that change, now the *Core* bundle does not depend on the *Message Handlers* or message bundles, but it depends on the *Message Handler Provider Service* bundle. Message bundles also depend on this service bundle too. This fixes the problem of the *Core* bundle depending on *Message Handlers* and its need to be updated to include a dependency with every new sensor integration. These message bundles, similar with every other OSGi bundle, can be extracted as a compiler .jar file and be installed externally.

Figure 7 shows the new classes and their hierarchies while Figure 8 shows the new message handler initialization procedure. The *Message Handler Provider Service Manager* implements the *IMessage Handler Provider Service* interface and when it is initialized by the *StartLevelEventDispatcher*, it reads a directory in which the new sensor integration bundles are placed as .jar files. The manager installs those new integrations and after the initialization of every new bundle, it registers itself as an instance that implements the *IMessage Handler Provider Service* interface to the OSGi context.

While those bundles are initialized, they register themselves with the *IMessage Handler Provider Service* in the OSGi context using the configuration identifier to indicate the sensor they should be used for. Accessing the registered *IMessage Handler Provider Service* is made possible through the *Message Handler Provider Service Util* class. This access technique blocks the requester thread until a service instance registers. The *Message Handler Provider Service Manager* registers itself after it initializes every integration file. Because *Message Handlers* access this manager using the same blocking technique, they can only register themselves after the service manager finishes its job. This causes all *Message Handlers* to register almost simultaneously.

While this process continues, the *Core* bundle also starts by the *StartLevelEventDispatcher* thread and continues its regular processes. But this time, the *Configuration Manager* class does not know any *Message Handler* itself. The dependencies for *Message Handler* classes are removed. When the *Configuration Manager* reads a sensor configuration, it uses its configuration identifier and asks a *Message Handler* constructor from the registered *IMessage Handler Provider Service*. It uses the *Message Handler Provider Service Util* class to access the service, so it also waits until an *IMessage Handler Provider Service* finishes its initializations and registers itself. After that, if a *Message Handler* for a given configuration identifier exists in the application, the *Configuration Manager* uses its constructor to create an instance and initialize it. The initialized *Message Handler* connects to the sensor and starts its regular processes. If a *Message Handler* does not exist for that identifier, the *Configuration Manager* skips that configuration for this iteration.

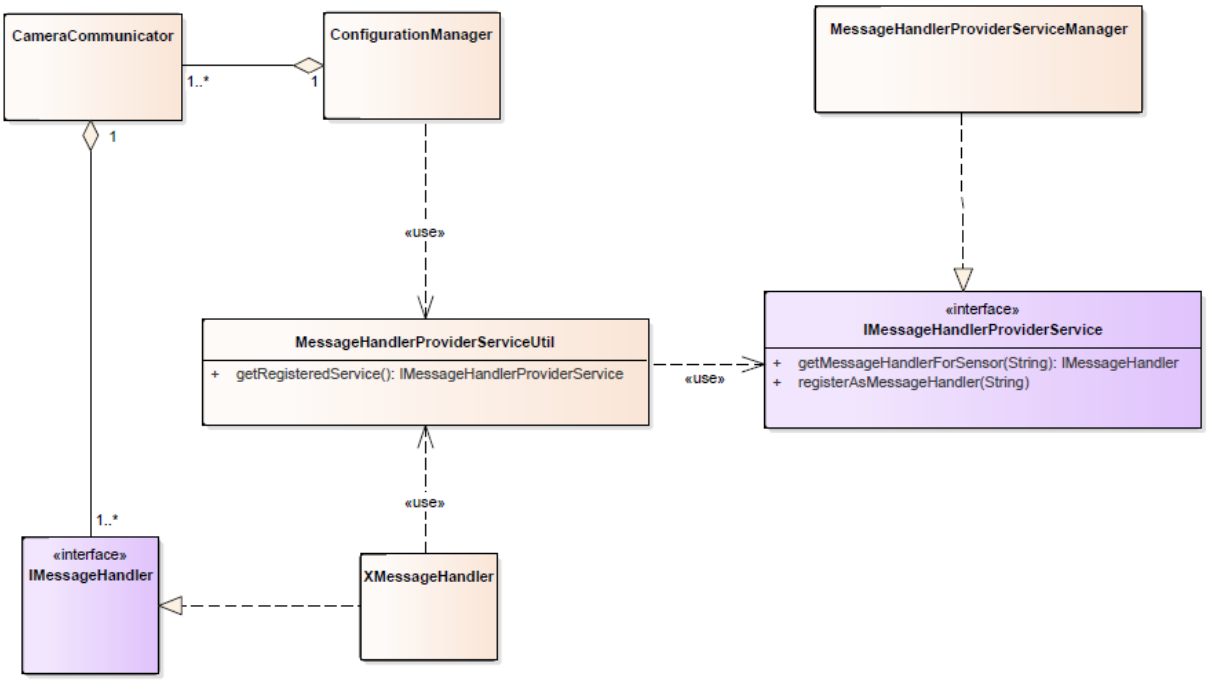


Figure 7. Camera Adapter Class Diagram (Simplified).

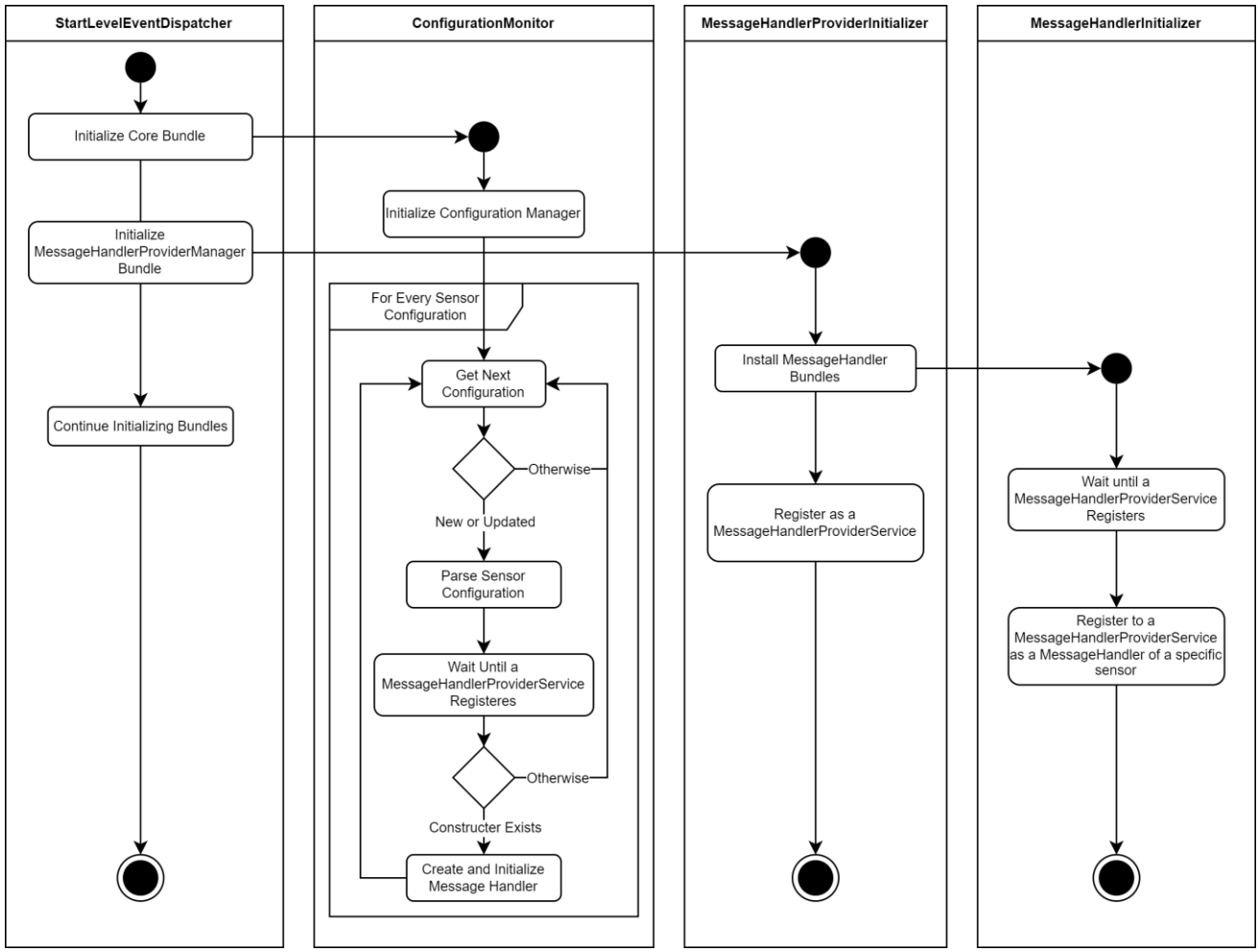


Figure 8. Message Handler initialization in the new architecture.

VII. Experiments And Survey

Similar to any other PSIM system, SecureX does not tolerate slow performance. It must provide a quick response capability for its users. Therefore, the architecture change must not cause a performance drawback. Also, to justify this architecture change, the new system must lower the test costs, as this was one of the promises of the new architecture.

A. Performance Tests

The main difference between the old and new architectures is the initialization of the adapter. As shown in Figure 7, the new initialization procedure is more complicated than the old one, which is shown in the Figure 4. Comparing those two diagrams, the difference mainly resides in how the *Configuration Manager* gets access to the *Message Handler* constructers. In the old architecture, *Configuration Manager* and all *Message Handlers* are in the Core bundle. Therefore, when *StartLevelEventDispatcher* initializes the Core bundle, every *Message Handler* class gets initialized along with the *Configuration Manager*. This enables *Configuration Manager* to access *Message Handlers* instantly without any additional dependency.

In the new architecture, *Message Handlers* are initialized in their separate bundles, and they register themselves to the *Message Handler Provider Manager*. *Configuration Manager* uses *Message Handler Provider Manager* to access the *Message Handler* constructers. This additional step causes a delay in the initialization phase of the adapter. But after the initialization is completed, any extra delay in other parts of the adapter is not expected. The tests confirm this hypothesis.

With the old architecture, time it takes to start the *ConfigurationManager* thread and for it to generate a *Message Handler* instance is on average 30 milliseconds, ranging between 29 and 31 milliseconds. In the new architecture, the average time for the same part of the initialization phase takes about 96 milliseconds, ranging from 88 to 104 milliseconds. This increase in time is the obvious result of not accessing the *Message Handler* constructers from within the same bundle and using an OSGi

service to do so. Both the *Message Handler*'s registration to the *Message Handles Provider Service*, and the service's own registration to the OSGi context takes time. But *Message Handler Provider Service Manager* keeps the *Message Handler* constructers in a map to easily access them if the *ConfigurationManager* needs it again. Therefore, this increased initialization time only happens on the first access of the constructer of a *Message Handler*. If there is another sensor of the same type in the system, the previous constructer gets used for the initializing of its *Message Handler*. But for a different type of sensor, another constructer must be generated.

This one time per sensor type increase in *Message Handler* initialization is trivial and it has little to no effect on SecureX's effectiveness.

After the *Message Handler* initializes, its operations such as processing, sending and receiving messages do not change between old and new architectures. On both architectures, typical message processing took about 3 milliseconds. This duration is the time between receiving a message from the SecureX server and after processing it, sending a notification to the server. So, the runtime performance of the adapter seems to be unaffected by this architecture change.

These tests show that the new design does not come with a significantly low performance. Increase in the initialization time is insignificant and hard to notice in the everyday use.

B. Survey for Cost of Testing

Another claim of the new design is that the test costs for a new sensor integration is high and the reason for this is the new bugs of the previously tested systems. We surveyed the testers who participated SecureX sensor integration tests to find out if that claim is true.

We have surveyed testers using in-depth interviews to understand the challenges in the SecureX tests. All nine testers who took the survey have at least one year, four of them has over three years of experience testing the SecureX sensor integrations. All participants had tested different cameras, radars, acoustic and seismic detectors. The used question set can be seen in Table 1. Based on their responses, on average, testing a camera or acoustic sensors takes about two hours, while a radar or seismic detector takes four hours. These test durations are not to be expected to be reduced by the proposed architecture change. New design does not provide a way to test one sensor faster, but it reduces the number of sensors to be tested after each new integration.

Table 1. Survey Questions

ID	Question
1	How long have you been testing software?
2	How long have you been testing SecureX?
3	What type of sensors did you test?
4	How many different cameras did you test?
5	How many different radars did you test?
6	How many different seismic detectors did you test?
7	How many different acoustic detectors did you test?
8	How long does it take to test a camera?
9	How long does it take to test a radar?
10	How long does it take to test a seismic detector?
11	How long does it take to test an acoustic detector?
12	In the last year, how many times was it necessary to go to the test site or the location where the system is installed to perform the test?
13	In the last year, how many times an intercity travel was necessary to reach to the test site or the location where the system is installed to perform the test?
14	How long does it take to go to the test site or the location and where the system is installed to perform the test?
15	In the last year, when a new sensor integration is tested, how often was it necessary to test other sensors of the same type as well? Ex: After testing a newly integrated camera, testing the other cameras in the system.
16	In the last year, when a new sensor integration is tested, how often a new bug from other sensors of the same type is detected?
17	How long does it take to fix and re-test the new bugs of the previously integrated sensors?
18	How much the development and test cost increase if new bugs of the previously integrated sensors were to be detected?
19	What was the worst-case scenario you experienced about bugs in previous integrations or increased test iterations like?

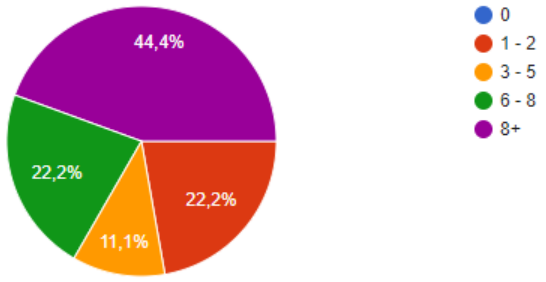


Figure 9. Distribution of the answers to Question 12.

Because the participants had worked on different projects that SecureX is used, their answers to the questions shown in Figure 9 and Figure 10 depend on those projects' test locations and configurations. If tests can be performed in house, and travel is unnecessary, the only test cost is the time spent testing the integrated sensors. When intercity travel is needed, the transportation and sometimes accommodation costs are added to the overall test cost.

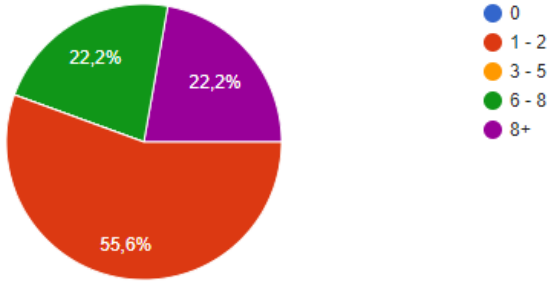


Figure 10. Distribution of the answers to Question 13.

But after testing the integrated sensor, tests for previously integrated sensors are needed; test durations for each of these sensors are also be considered when calculating the test cost.

Figure 11 shows that every participant stated that it was always necessary to test the previously tested sensors of the same type after a new sensor integration is completed. For example, after testing a newly integrated camera, testing other cameras in the system. Six of the testers said at least three times this was necessary and two of them said they had to test other sensors on more than eight occasions. Figure 12 shows the reasoning behind these additional tests. Often after

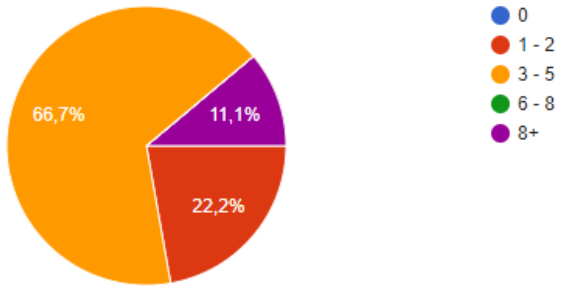


Figure 11. Distribution of the answers to Question 16.

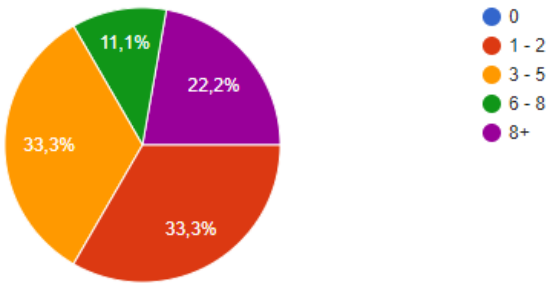


Figure 12. Distribution of the answers to Question 15.

a new sensor integration is completed, these additional tests reveal new bugs of the previously integrated sensors. Sensor integration in the old architecture does not segregate these integrations enough and provides an environment that is error prune.

These bugs extend the test and development duration as they need to be fixed and tested again. Also, the possibility of a bug occurring in the previous integrations cause testers to request testing those integrations whenever a new integration gets completed. As shown in Figure 13, this extra test and development process comes with an average cost increase of 30% to 50%, depending on the project configuration and location.

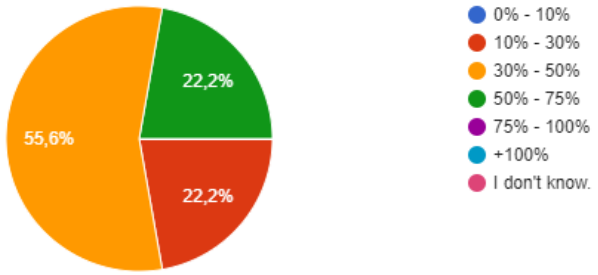


Figure 13. Distribution of the answers to Question 18.

The participants also asked what was the worst-case scenario that they experienced about sensor integration tests. Testers also point out that after the development and bug fixing processes for the bugs in the integration of sensors of same type, it was observed that the previously acquired and tested capabilities from other sensors were lost. This situation creates the need to review the integrations repeatedly and retest them after each bug fix. On one occasion, a SecureX system was installed at a remote location and is used by the operators when the customer wanted a new camera to be added for their changed security needs. The camera integration completed and tested at the company. But because the SecureX configuration the customer uses contains sensors not available at that moment during the tests of the new sensor, testers had to go to the location that SecureX system is installed. They, along with a developer, tested other sensors that the customer uses in order to verify that they still function as before. Testers found a couple of bugs and the developer fixed them and

after the re-test of the system, it is left to the operators again. During these tests, controls of the tested sensors are taken from the operators, and this lowers the PSIM system's availability.

As these tests take place in the customer's deployment, they may give customers a bad reputation about SecureX. Because for any bug that is found in those tests, there is a chance that it can be seen by the customer, or the security personnel at the location. Because it is not always possible to complete the tests without involving anyone from the customer's company. Usually, tests take place in the same room that the security personnel use and anyone would be curious to see what the new sensor can do. Even if the found bugs are minor or hard to reproduce in typical usage of the system, it would not matter if the customer realizes those bugs as well. If this situation keeps happening, reputation of SecureX would start to decline. Also, these security personnel or the customers themselves would mention their own requests from the PSIM system. Without anyone from the project management, the testers and developers are not always expected, to discuss the details of those requests.

Participants noted that for regression tests, having to go to remote locations where the existing systems are installed is costly and a way to reduce those costs are needed.

VIII. CONCLUSION

The proposed adapter architecture allows us to integrate additional sensors into the already deployed PSIM systems, without requiring to generate another complete build of an adapter software. Because previous integrations are not touched, integration tests of only the newly integrated sensors would be sufficient. When the sensor is integrated, it will most probably be available for testing as well and going to the field for using the sensor of a customer in order to conduct the tests will no longer be needed.

The survey with the testers has showed the extra work that needs to be done because of the new or reoccurring bugs in already tested sensor integrations. With the new architecture, these additional tests are no longer a regular requirement that takes place every time a new sensor gets integrated to the SecureX. Testing the sensor integrations only at their own integration times and keeping them bug free, even if new sensors or systems added to the project is crucial. An additional capability to a system should not take away or break the already existing and used capabilities. The cost of re-testing previously tested system after every integration is not something to be ignored. In addition to the amount of man hour being wasted for these tests, the financial cost also includes the logistic costs, which is depended on the test location. Therefore, the training that seemed to be required to work with this architectureas Almedia E. et al. found in their work [9], require a cost that is not worth mentioning of, when those additional test costs are considered.

The .jar files of the integration bundles are smaller than one MB. Thanks to these low sized components, system updates can be completed even with unstable or slow networks. Even if new sensor integrations have a problem working with previously integrated sensors, simply removing the .jar file would be enough to revert back to the previous deployment.

Segregating sensor integration also enables easily selecting and combining different integration bundles according to the project's requirement, as one could expect from a system developed with software product line principles. When starting a new project, depending on the sensors that are going to be used, only their integration files can be used. There is no need for adding every sensor integration to the project. The new design also enables employing 3rd party companies for integrations without sharing the bulk of the adapter code. Now, any integrator can develop an integration bundle only with the *Message Handler*, *IMessage Handler Provider Service* and the *Message Handler Provider Service Util* classes.

The proposed architecture is also shown to have similar performance with its predecessor with only a minimal delay at startup. Even if this startup duration increase was much more, if it's not extreme, it still might not be a problem. Generally, PSIM systems are not expected to shutdown and startup frequently. Due to high availability requirements, they tend to be designed as if they were expected to run continuously. So this minor increase in the initialization time can easily be ignored. Also, with the new architecture, stopping the system for adding a new senosr integration or changing an integration file is not required. New integrations can be added or updated while the system is running. This new capability also lowers the amount of times that the system had to be restarted. After the system is restarted and initialization is completed, the performance of the system was the same as it was with the old architecture. Only thing that the new architecture changes is the way sensor integrations are initialized.

The new architecture provides a helpful pattern towards transforming SecureX into a Software Product Line (SPL). An external .jar installer service could be used not only for sensor integrations, but also for features such as additional GUI views or in the server, new alarm evaluation algorithms. Because every feature is developed as an OSGi bundle, they all could be externalized. The sensor integration problem could be solved by developing an SDK, similar to the products given in Section II, but our design also eliminates the need of deploying the SecureX with a full feature set and stripping it off with configuration files at runtime. As this design gets implemented in other parts of SecureX, they could all be removed from the base build and can be added per customer demand.

The new design opens an evolutionary path for segregating such different aspects in SecureX architecture and is expected to be even more beneficial in the future. As such, the architectural change is not only applicable to PSIM systems like SecureX, but also any system that is developed with OSGi. Because at its core, our work can be described as a case study in how a software product line architecture can be implemented in an OSGi based product. What we achieve is within reach of any similar product.

We altered the existing architecture and took advantage of the OSGi framework to improve the modularity of our system. The modularity we achieve is a crucial requirement

for an SPL architecture, as SPL products are actually a combination of features selected from a feature set to satisfy particular requirements. These features can be developed in the same manner the sensor integration jar files are developed in the new architecture. And feature selection can be completed by using different feature jar files for different requirements. *Message Handler Provider Service Util* class and *IMessage Handler Provider Service* interface gives an example on how to select and use different features as well. Therefore, the architecture we proposed can be used in any software product line project.

REFERENCES

[1] O. Özçelik, M. H. S. Oğuztüzün, "Software Architecture Evolution of a Physical Security Information Management System". The Eighth International Conference on Advances and Trends in Software Engineering (SOFTENG), 2022, pp. 15-20.

[2] M. L. Garcia. The design and evaluation of physical protection systems. 2nd ed. Amsterdam: Elsevier, 2008.

[3] Genetec KiwiVision. [Online], retrieved May 2023 Available: https://www.genetec.com/products/

[4] Milestone XProtect. [Online], retrieved May 2023 Available: https://www.milestonesys.com/solutions/

[5] Nedap Aeos Access Control. [Online], retrieved May 2023 Available: https://www.nedapsecurity.com/solutions/

[6] Open Network Video Interface Forum (ONVIF). [Online], retrieved May 2023 Available: https://www.onvif.org/

[7] B. Tekinerdoğan, İ. Yakın, S. Yağız, and K. Özcan, "Product Line Architecture Design of Software-Intensive Physical Protection Systems". IEEE International Symposium on Systems Engineering (ISSE), 2020, pp. 1-8, doi: 10.1109/ISSE49799.2020.9272239.

[8] "The Java Language Specification, Java SE 8 Edition" J. Gosling, B. Joy, G. Steele, G. Bracha, and A. Buckley. Apr. 2015. [Online]. retrieved May 2023 Available: https://docs.oracle.com

[9] E. Almeida, et al. "Domain Implementation in Software Product Lines Using OSGi". Seventh International Conference on Composition-Based Software Systems, 2008, doi: 10.1109/ICCBSS.2008.19

[10] R. S. Hall, K. Pauls, S. McCulloch, and D. Savage. "OSGi in Action - Creating Modular Applications in Java". Manning Publications, 2011

[11] "OSGi Service Platform, Core Specification, Release 8," The OSGi Alliance, April. 2018. [Online]. retrieved May 2023 Available: http://docs.osgi.org/specification/

Managing Cyber Black Swans

Can potentially crippling cyber situations be foreseen, allayed, and turned into growth opportunities?

Anne Coull
Objective Insight
Sydney, Australia
anne.coull@proton.me

Elena Sitnikova
Flinders University
Adelaide, Australia
elena.sitnikova@flinders.edu.au

***Abstract*— Black Swan situations and their consequences are considered extremely unlikely before they happen and make perfect sense afterwards. Two malicious cyber attacks that triggered Black Swan situations, Emotet and WannaCry, are assessed, along with their attack sequences, and the vulnerabilities they exploited. The early warning signs and practical actions to prevent these types of Cyber Black Swan situations are presented. Prevention is based on protection through practical defence in depth controls along with effective ongoing maintenance. Added to this is the crucial element of situational awareness and a call to action for the cyber teams to focus their response efforts. This robust foundation of security and resilience, when combined with adaptability, are the attributes for antifragility. Enabling the organisation to thrive and grow in the midst of this volatility.**

Keywords- Black Swan; Emotet; WannaCry; Early Warning Indicator; Critical Vulnerability; Situational Awareness; Response; Antifragility; Adaptability.

I. INTRODUCTION

In his book: "Antifragility, things that gain from disorder," Nassim Taleb [51] uses the term *Black Swan* to describe unexpected situations with three attributes: Before the situation occurs, it is considered extremely unlikely, if not impossible; When it occurs its consequences are significant, either in changing belief, or in consequence; After it has occurred, it makes perfect sense as something that could happen [1][19][52]. As an Australian, the notion of a Black Swan as an unexpected event is counter intuitive. While the swans in Europe may be white, in Australia the native swans are black. In a country of jumping kangaroos and duck-billed platypus, the unexpected is modus operandi [1].

With Australian insight it becomes clear that unusual creatures and events do not just suddenly appear, they evolve over time. Similarly, Black Swan situations develop over time and show early warning indicators. Noticing these early signs and acting upon them, will make the difference between a dramatic event, a well-managed situation, or just another day doing business [1]. The proposed approach for responding to these Black Swans is based on situational awareness, basic, practical, and well-maintained cyber controls, and response to emergency situations.

Two black swan cyber situations, the Emotet Trojan, and the WannaCry Worm, are reviewed along with their attack vectors and the vulnerabilities they target. These two cyber attacks that triggered Black Swan situations were selected due to their scale and impact, which in turn can be attributed primarily to the lack of preparation and poor response of the target organisations. These attacks differed in their initial access approach, their style of attack, and the combinations of attack vectors they utilised [6][44]. For each of these black swan cyber situations, the potential for predictability and reduced impact through stringent maintenance and monitoring, situational awareness, and response to early warning indicators is assessed.

This approach to cyber security and resilience requires a combination of structured, pragmatic thinking and follow-through, along with action-based responses to emerging threats. If the organization is to grow and thrive, rather than merely survive from these events, adaptive thinking is also needed. Antifragility develops through environmental awareness, seeking out and being open to opportunities, and taking action to take advantage of the volatile environment and create new markets.

Section 2 outlines the Emotet and WannaCry exploits and their respective impacts. Section 3 analyses their attack sequences for access, escalation, persistence, scanning, spread, exfiltration, and assault. Section 4 looks at how these events could have been foreseen by reading the early warning indicators. Section 5 outlines how these attacks and others like them can be mitigated using practical cyber defence in depth with effective ongoing maintenance, situational awareness, and timely response. Section 6 addresses the actions needed to contain, manage, and recover from an infection. Section 7 explains how antifragile organisations, that are adaptable and agile, can gain advantage from these situations, and thrive in these volatile cyber environments.

II. EMOTET AND WANNACRY EXPLOITS AND THEIR IMPACTS

Over the last decade two of the most significant cyber attacks, in terms of scale and impact, have been Emotet and WannaCry. Both utilised a combination of exploits to target Microsoft vulnerabilities and gain access to organisations, establish persistence, escalate privileges, and exfiltrate data whilst concurrently spreading, infecting, establishing a foothold, and implementing assault strategies across the network [6][7][22][27][35][42][43][46][48]-[50].

A. *Emotet scale and impact*

Emotet is believed to be based out of Ukraine [17][44]. It started as a banking Trojan and has continued to evolve since it was first identified in 2014 [17][44] (see Figure 1).

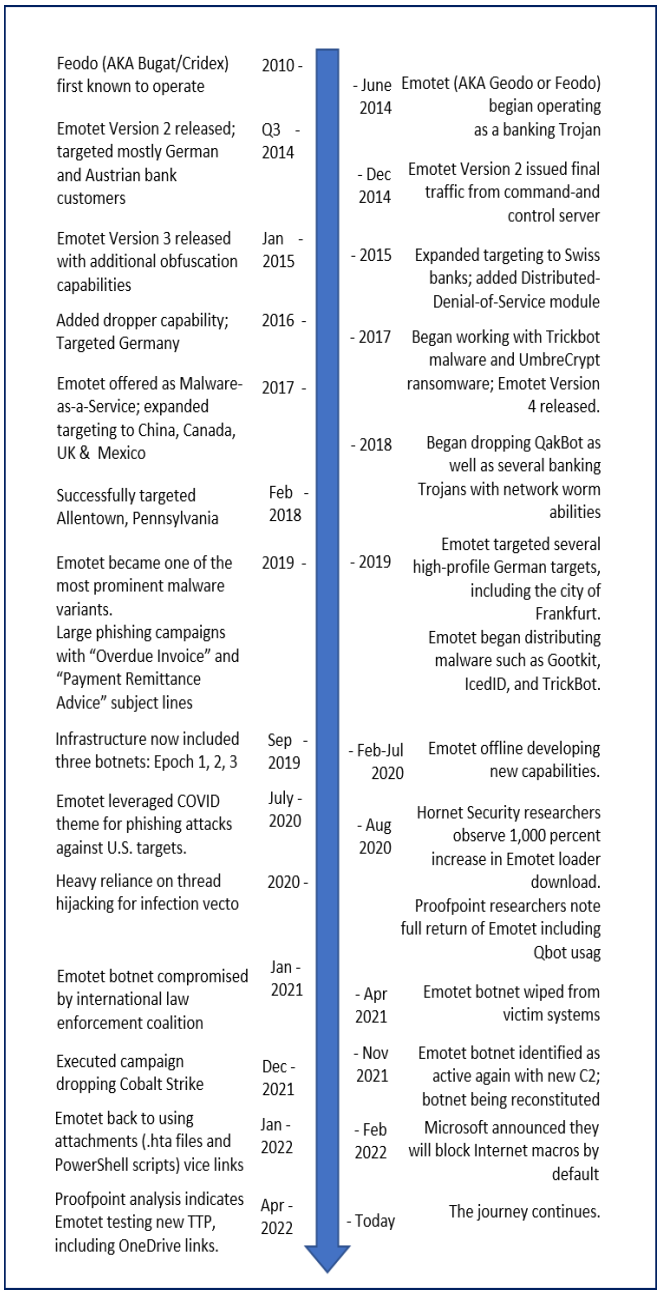


Figure 1. Emotet timeline, adapted from [17].

In 2019, Emotet was responsible for approximately 60% of malware email spam [44]. By 2020 it had morphed into Botnet as a Service, and Malware as a Service (MaaS) with global distribution [55]. Infected devices themselves become command and control (C2) bots. In May 2019, 310 unique infected IP addresses were identified, of which two thirds (208) were confirmed bots, and 8% (17) of these were also infected with Trickbot [43] (see Figure 2).

Figure 2. Geographic distribution of Emotet Botnet IP addresses, June 2019 [40].

In January 2021, the German Bundeskriminalamt (BKA) federal police agency coordinated a combined effort of law enforcement agencies to shut down the global botnet of hundreds of Emotet servers [43]. The Trojan malware, or a copycat, returned in November 2021 and infected an estimated 1.2 million systems in 2022 [17][43][55] (see Figure 1).

B. *WannaCry scale and impact*

In May 2017, after infecting more than 300,000 computers and crippling 150+ organisations worldwide. WannaCry was dubbed "the largest ransomware event in history" [43] (see Figure 3). The WannaCry ransomware attack was stopped by a MalwareTech cyber researcher, who identified a key design flaw and purchased the URL WannaCry referenced in its attack sequence [26] (see Figure 4).

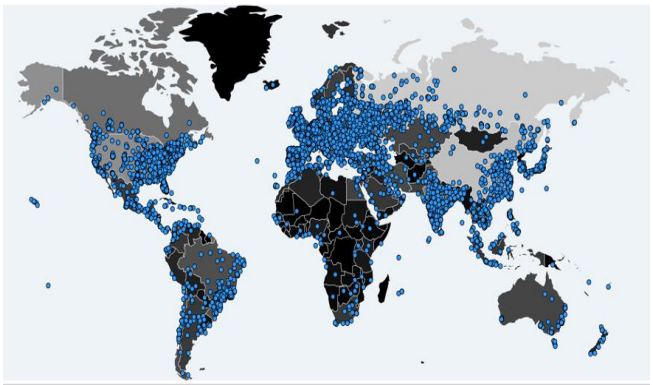


Figure 3. Distribution of WannaCry infections 14 May 2017, after 24 hours [6][23].

The situations triggered by WannaCry and Emotet could both be regarded as Black Swans. Each was considered extremely unlikely before they were experienced and identified. Each gained the attention of Europol and Eurojust due to their scale and the significant and costly consequences for those affected [43]. WannaCry brought

the British NHS to a standstill [15], including the closure of public hospitals. By June 2022, Emotet had spread from banks to auto & other manufacturers, health, government, education, transport, real estate, and retail [16][17] across Japan, Asia Pacific, Europe, Middle East, Africa, North and South America [17] (see Figure 2). It was estimated as costing in excess of $1 million for every organisation it infected [6][43].

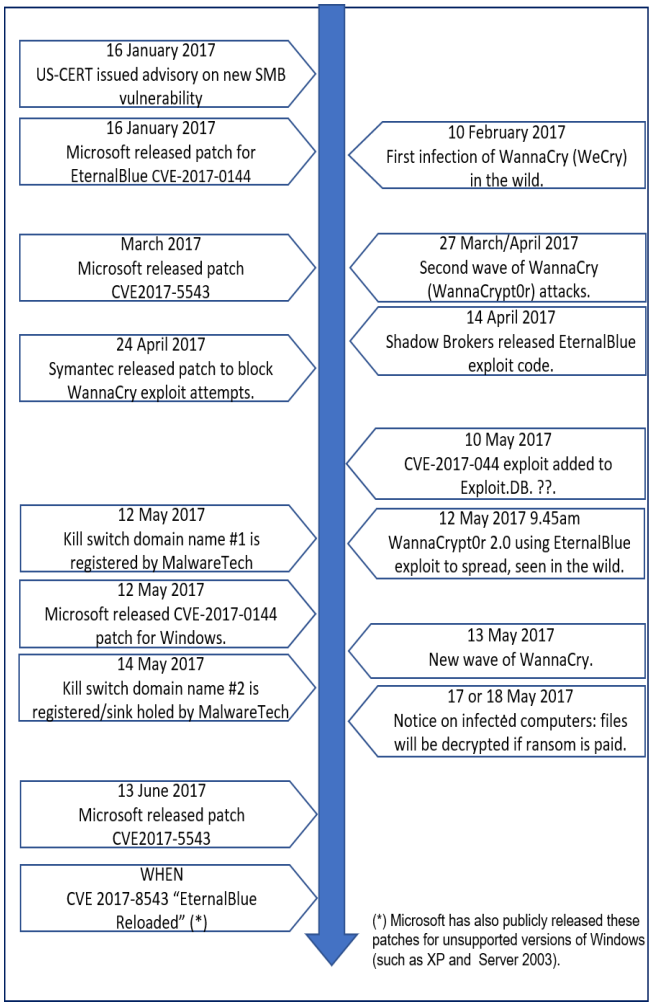


Figure 4. WannaCry timeline, derived from [18][25][26][49].

III. ATTACK SEQUENCE ANALYSIS

While there are some commonalities in the zero-day exploits targeting Microsoft SMB remote control vulnerabilities, Emotet and WannaCry utilise different attack vectors in the infection process.

A. *Emotet access, escalation, persistence, scanning, spread, exfiltration, and assault*

Emotet utilises social engineering phishing campaigns to entice recipients to click on a link that downloads a macro-infected Microsoft office file. These emails appear to come from a friend or colleague, or from a known organisation, and include PayPal receipts, shipping notifications, "past-due" invoices [6], or COVID information [41] (see Figure 5). The macro executes the payload malware for the next stage, where it establishes persistence using auto-start registry keys and services to embed a scheduled task at startup [6][42][43].

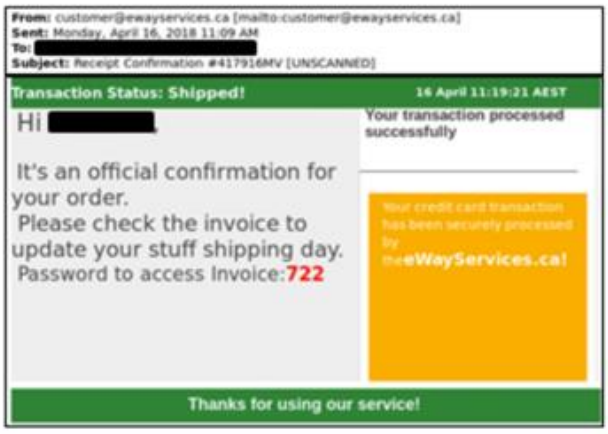


Figure 5. Emotet malicious email Emotet [6].

Emotet spreads by extracting contact lists from infected users' email accounts and using these to send phishing emails so they appear to come from a friend or colleague. Concurrently, Emotet spreads to systems across the network by enlisting a credential enumerator with service and bypass components. It utilises publicly available tools to recover passwords: (i) NetPass.exe from NirSoft extracts passwords stored on the user's system and external drives; (ii) WebBrowserPassView extracts passwords stored on web-browsers such as Google Chrome, Internet Explorer, Mozilla etc.; and (iii) MailPassView extracts passwords stored on email providers such as Gmail, Outlook, Hotmail etc.

Emotet concurrently utilises a malicious-actor-developed spreader module that applies brute force with enriched password lists to move through the Windows Admin Shares. It uses these credentials to access accounts and copy itself to the ADMIN$ of other network hosts, before using Server Message Block (SMB) to schedule execution on these hosts. It locates writable share drives and infects the entire disk by writing the Emotet service component onto the network [6][42][43] (see Figure 6).

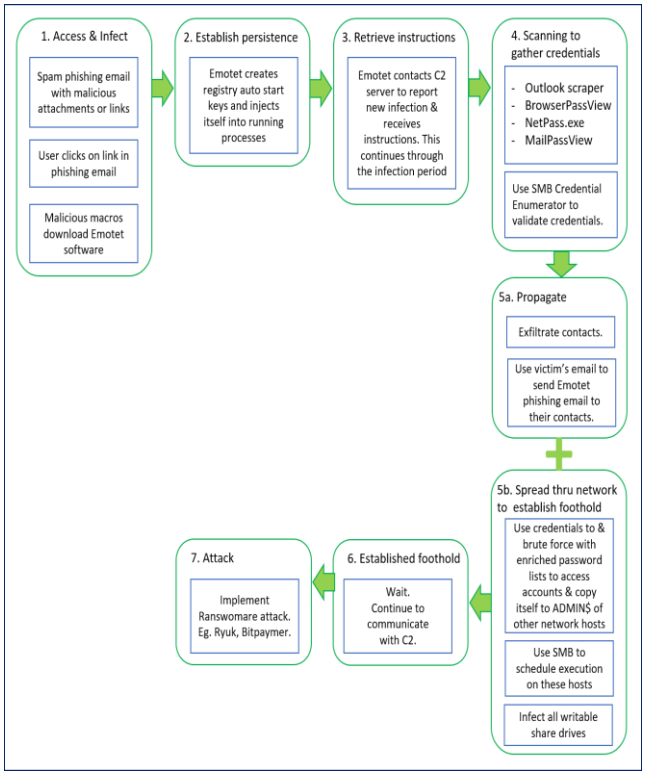


Figure 6. Emotet attack sequence, derived from [6][42][43].

From 2016 Emotet incorporated a Trickbot banking trojan which evolved to exploit the Microsoft Windows SMBv1 and NBT Remote Code Execution Vulnerabilities (CVE-2017-0144, CVE-2017-0147), and the Windows SMB Remote Code Execution Vulnerabilities (CVE-2019-0630, CVE-2019- 0633) [6][9][11]-[13][29][31]-[33][36][38]-[40][42]. The Trickbot is used to launch the malware payload, bypass Microsoft security measures, communicate with the command-and-control infrastructure, upload data and download DLL updates [42].

B. WannaCry access, escalation, persistence, scanning, spread, exfiltration, and assault

WannaCry identified its targets using EternalBlue to scan externally facing hosts across the internet where TCP ports 139 and 445 were open [54]. These ports are used to communicate using the SMB network protocol that enables remote code execution in MS Windows & sharing across networks [7][22][50].

When it identified the Microsoft SMB Windows Server Remote Code Execution Vulnerability (CVE-2017-0144) and the Microsoft SMB Windows Server Remote Code Execution Vulnerability (CVE-2017-0145) [9][10][29][30][36][37][49] which enabled remote code execution over SMB v1, EternalBlue then accessed the vulnerable target systems and installed the DoublePulsar exploit for persistence [7][27][35]. It continued to scan, access and replicate while it encrypted files and destroyed backups on every computer it infected: disrupting businesses by denying users access to their critical data [18][46][48][49] (see Figure 8).

It then displayed a ransomware image to the users of infected devices [7][24] (see Figure 7).

Figure 7. WannaCry image displayed on infected user's desktop [7][24].

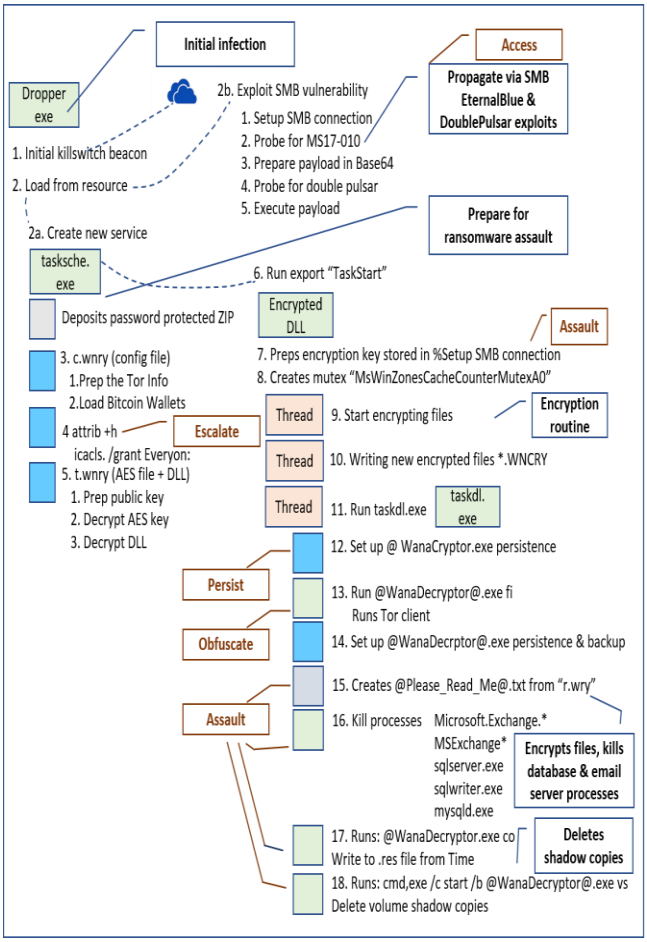


Figure 8. WannaCry infection process, adapted from [46].

IV. Proactively Looking for the Early Warning Indicators

Situational awareness is key to preventing malicious exploits developing into Black Swan situations. It enables organisations to notice the early warning signs and prepare for and respond to emerging situations. The early warning signs are there for Emotet and WannaCry, but they will only be noticed by those who actively seek them out. The early warning signs include:

1. The cybersecurity capability of the organisation:
    a. The cyber-risk awareness of personnel, based on their click-rate on targeted phishing campaigns.
    b. The level of compliance with standards and guidelines for basic defence maintenance practices, including compliance to the Australian Signal Directorate's Essential Eight cyber mitigations, in particular: the extent of unpatched Microsoft windows systems; privileged access management; ability to download macro-enabled email attachments; and availability of separately stored backup data [2][3][8].
2. Critical vulnerability reports:
    a. Microsoft CVE-2017-0144, CVE-2017-0145, & CVE-2017-0147 vulnerability reports published in the Microsoft Vulnerability Update Guide on 14 March 2017 [29]-[31] and corresponding CVE reports [9]-[11] and NIST reports published on 16 March 2017 [36]-[38].
    b. Microsoft CVE-2019-0630 & CVE-2019-0633 vulnerability reports published in the Microsoft Vulnerability Update Guide on 12 February 2019 [32][33] and corresponding CVE reports [12][13] and NIST reports published on 3 May 2019 [39][40].
3. Threat alerts and reports:
    a. Threat alerts and reports are readily available through research centres such as Fifth Quadrant [14], Malwarebytes [25], MalwareTech [26], Metasploit [28], Qualys [45], Proofpoint [44], Talos [53], Truesec [57][58], and Verizon [61].
    b. Cyber teams in peer organisations sharing information. Australian organisations, including the big-4 Banks, openly share information in a joint effort to fight cyber crime, directly and through the Joint Cyber Security Centre (JCSC) [20].
    c. Up-to-date information is available through the reports provided by the Australian Signals Directorate (ASD) and US Cert [4][6].

V. Preventing Black Swan Situations

Emotet, WannaCry and similar trojan and worm-based malware exploits can be prevented, and/or their effects limited by applying basic cyber defence maintenance practices, maintaining situational awareness, and monitoring and responding when a threat is identified, or suspected.

A. Applying basic cyber defence maintenance practices

1. Address the weakest link. Educate all people in the organisation on the risks and indicators of cyber exploits, such as emails with links and attachments. Educate people to *not* click on links or unpreviewed attachments and to run their mouse over to see where it links to, even if the email comes from a trusted colleague or friend. Educate them to *not* click on online advertisements, and to never share unencrypted sensitive information through external email or on the phone [6][8].

2. Incorporate desired cyber practices into policies. For example, implement a policy requiring users to forward suspicious emails to the security team [6].

3. Control and monitor who has access to what, when. Implement Privileged Access Management based on the principle of least privilege [8].

4. Keep all operating system and application patching up-to-date, by applying tested patches and updates as a priority. In particular, test and apply critical patches immediately. Five weeks prior to the main WannaCry attack, both the Australian Signals Directorate [4] and Microsoft had issued updated CVE reports. Microsoft released emergency patches to the Windows SMB vulnerabilities that enabled WannaCry's EternalBlue and DoublePulsar exploits recommending updates be applied immediately [7][8][34].

5. Regularly perform vulnerability scans to identify any unpatched devices [8][45].

6. Set a Firewall rule to restrict inbound SMB communication between client systems, using Windows Group Policy Object, or if using a non-windows host-based intrusion prevention system (HIPS), implement custom modifications for the control of client-to-client SMB communication [6].

7. Using antivirus programs on clients and servers, with automatic updates of signatures and software will mitigate against many other malware exploits that are signature based [6][8].

8. Whitelist IP addresses and block suspicious and known malicious IP addresses at the firewall. Filter out emails with known malspam indicators, such as known malicious subject lines, by implementing filters at the email gateway [6][8].

9. Block or scan file attachments commonly associated with malware, such as .dll and .exe and those that include macros, as well as attachments that cannot be scanned by antivirus software, such as .zip files [6][8].

10. Disable macros and PowerShell to prevent macro driven PowerShell commands, such as those utilised by Emotet [6][8].

11. Implement Domain-Based Message Authentication, Reporting & Conformance (DMARC), a validation system that minimises spam emails by detecting email spoofing using Domain Name System (DNS) records and digital signatures [6].

12. Be prepared for the worst. Take daily backups for timely recovery and restoration of service to the business and its customers. Ensure these are stored offline or on a separate network and restoration is tested regularly to prevent failure when restoration is really needed [2][3][6][8].

13. Limit exposure of critical systems to zero-day exploits. Take vulnerable, critical systems off-line and/or restrict their external accessibility when a zero-day exploit is underway.

14. Apply emergency zero-day patches immediately. During the WannaCry event, Microsoft released emergency patches for out-of-support versions of MS Windows such as XP and Server 2003 [18] [38][46][49].

B. Maintaining situational awareness

Security teams need to be tasked with staying abreast of the current and emerging threats, the vulnerability-state of the organisation's systems and people, the effectiveness of existing controls, and availability of updated controls to address emerging vulnerabilities. The Cyber Security Strategy, based on robust risk management, needs to be maintained in-line with the evolving environment. Ongoing communication to key stakeholders is critical to ensure organisational support and resourcing.

C. Responding proactively to emerging threats and exploits

Historically, Australia has faced the global-scale cyber attacks after Europe and North America. This has provided a short window in which to act, and mitigate known vulnerabilities. On Sunday 14th of May, Australian organisations had a few hours during which to ensure critical windows-based systems were patched or segregated to protect themselves from the ensuring WannaCry attack, prior to business opening on the Monday morning.

As cyber exploits are combined in new and crafty ways, the ingeniousness of the response needs to match that of the adversaries.

VI. MANAGING AN INFECTION

Priority response actions are needed to minimise the impact of these cyber Black Swan situations, for organisations that become infected.

A. Managing an Emotet infection.

Emotet progresses with credential extraction, captures the user's credentials on all accounts and systems, and spreads across the network prior to issuing a ransomware demand that makes the user eventually aware of the infection. By this time Emotet is embedded throughout the network. Immediate action is required to:

1. Contain the infection, to prevent it spreading across the network. Identify, shutdown, and take the infected machines off the network. Don't use domain or shared local administrator accounts to log into infected systems [5].
2. It may be necessary to temporarily take the network offline [5].
3. Remove the Emotet malware from infected devices by reimaging these devices, and ensure the windows patches provided by Microsoft have been applied [5][32][33][36][38].
4. Apply backups.
5. Prevent re-infection. Scan the reimaged and clean systems and move these to a segregated virtual local area, away from the infected network [5].
6. Issue password resets for all affected credential groups, including password vaults [5].
7. Identify the original source of the infection. ie. The device that was first infected [5].
8. Review the log files for this user's account to ensure any auto-forward rules for emails are turned off, and prevent further data breaches [5].
9. Use the clean systems to return to normal business operations.
10. Continue to monitor for any unidentified infections, affected systems, and users.
11. Document the incident report.
12. Communicate with senior stakeholders, employees and customers, throughout. Sensitive customer

information may have been exfiltrated during the infected period. Communicate to impacted entities and regulatory bodies in line with company policy and regulatory requirements.
13. To prevent re-infections: Review the operating system maintenance and patching processes; and Educate users to not click on phishing email links.
14. If not already in place, implement targeted educational phishing campaigns.
15. Post Incident Review to assess protection, response, and restoration effectiveness and identify improvements.
16. Communicate findings, and action improvements.
17. Post Implementation Review the system maintenance processes, and measure the click rate in the organisation 3 & 6 months later to ensure the required changes have been made and the desired click rated achieved.
18. Implement further improvements, if required.

B. *Managing a WannaCry infection.*

Those infected with WannaCry malware were immediately made aware by the ransomware messages, and the loss of data availability due to WannaCry encrypting all the word, excel, pdf etc. files [6][21]. Immediate action was required to:

1. Contain the infection, to prevent it spreading across the network and prevent re-infection, by identifying unpatched devices applying the Windows patches provided by Microsoft [32][33][39][40].
2. Implement regular vulnerability scanning to identify unpatched devices [45]. Targeted vulnerability scanners that focus on finding the specific Windows vulnerabilities that enabled WannaCry, are now also available [59].
3. Recover the encrypted files from backups. At the time of the original WannaCry attack, the only way to recover lost or encrypted files, was to restore from backup. Since then, WannaCry decryptors have been developed [60].
4. Return to normal business operations.
5. Continue to monitor for any unidentified infections and affected files.
6. Document the incident report.
7. Communicate with senior stakeholders, employees, and customers, throughout. Communicate to regulatory bodies in line with company policy and regulatory requirements.
8. Review the operating system maintenance and patching processes to prevent similar infections.
9. Post Incident Review to assess protection, response, and restoration effectiveness and identify improvements.
10. Communicate findings, and action improvements.
11. Post Implementation Review these processes 3 & 6 months later to ensure the required changes have been made.
12. Implement further improvements, if required.

VII. FROM SURVIVE TO THRIVE

The practices of situational awareness, proactive risk management, systems' maintenance, and priority response to threats and infections are the fundamentals of good cyber and systems' management practices. When performed consistently over time, these practices provide a solid foundation for robust security and business resilience. Significantly reducing the likelihood that the organization will be heavily impacted by a cyber attack. But routine procedures are designed to operate within known thresholds. When unexpected and unknown events exceed these thresholds, such as those experienced in cyber black swan situations, these alone cannot necessarily be relied upon to perform optimally. These situations require more flexible response [47].

Fragility relates to how systems are negatively impacted when the environmental volatility exceeds a threshold [52]. Organisational risk management is typically based on known historical events. The thresholds of the risk management are defined by these past events. By definition, black swan events are unexpected, and lie outside the threshold [51][52].

The process of evolution through natural selection produced the Australian platypus and kangaroos. Evolution is, at its core, an example of antifragility [56]. Only organisms that are adaptable to the changing environmental conditions and variables survive, thrive, and proliferate. Those that are only robust within stable environments do not survive when this environment changes rapidly or unexpectedly.

Paradoxically, cyber antifragility is built on a foundation of structure, standardisation, and good cyber and systems practices, combined with behavioural adaptability to volatile situations to take advantage of opportunities emerging from the disruption [56].

Going beyond security and resilience to antifragility moves the focus from surviving to thriving in the face of adversity. The critical attribute displayed by antifragile organisations is their ability to modify goals and behaviours when in crisis. This requires flexible thinking, problem solving, and in the case of cyber black swan situations, ability to utilise technology to obtain and interpret information in real time [56]. Leaders, and teams need to be able to adjust their ways of working and coordinate

differently; To adjust their performance strategies, and adapt to the new situation [47].

Antifragile organisations turn crises into opportunity by rapidly modifying their business model and organisational behaviours to take advantage of the situation. This requires foresight, a hunger for new opportunities, agility to rethink and change the way they work, ongoing research to determine how this can be achieved, and capital investment to enable it [56].

Cyber antifragility is more than just defence or resilience as it necessitates both stability and growth in the face of adversity, of any scale. Rather than just recovering to the pre-event state, after the incident, antifragile organisations improve as a result of being regularly challenged by new cyber events and situation [56].

VIII. Conclusion

Analysis of the Black Swan Situations generated by the Emotet and WannaCry malicious exploits highlight the ways in which these types of situations can be predicted and prevented.

Situational awareness enables organisations to notice and interpret the early warning signs; to stay abreast of and prepare for emerging vulnerabilities and threats. Preparation involves addressing the weakest link as well as implementing practical controls, such as privileged access management, maintaining defence in depth, and keeping reliable backups.

Awareness of its cyber capability and open vulnerabilities also ensures the organisation can respond to zero-day exploits by limiting exposure of critical systems and immediately applying emergency patches to vulnerable systems. Priority protection and response action when an infection is detected limits the extent of the infection, its effects, and the impact on the business and customers. These are the fundamentals of good systems and cyber management practice, providing security and resilience.

Antifragile organisations take this to the next level by developing the agility to adapt in the face of adversity, to take advantage of the situation. While others survive, these organisations thrive, benefiting from the constantly changing threat landscape and the cyber black swans.

References

[1] A. Coull, “Black Swan or Just an Ugly Duckling?, Can potentially crippling cyber situations be foreseen and mitigated? ” IARIA CYBER 2022 : The Seventh International Conference on Cyber-Technologies and Cyber-Systems. Available from: https://www.thinkmind.org/index.php?view=instance&instance=CYBER+2022

[2] ACSC, “Strategies to mitigate cyber security incidents, “Australian Government, Australian Signals Directorate, 2017, Available from: https://www.cyber.gov.au/acsc/view-all-content/publications/strategies-mitigate-cyber-security-incidents, accessed October 2022.

[3] ACSC, “Essential eight explained, Australian Government,” Australian Signals Directorate, 2019, Available from: https://www.cyber.gov.au/sites/default/files/2020-01/PROTECT%20-%20Essential%20Eight%20Explained%20%28April%202019%29.pdf, accessed October 2020.

[4] ACSC, Australian Signals Directorate - view all alerts, Available from: https://www.cyber.gov.au/acsc/view-all-content/alerts, accessed February 2023.

[5] 3+1Any run, “Emotet”, 2021, Available from: https://any.run/malware-trends/emotet, accessed October 2022.

[6] CISA 2018-2020, “Alert [TA18-201A] Emotet Malware,” Available from: https://www.cisa.gov/uscert/ncas/alerts/TA18-201A, accessed October 2022.

[7] A. Coull, “WannaCry Malware Case Study," Cyber Security Operations 2017, UNSW.

[8] A. Coull, “How much cyber security is enough,” The Fourth International Conference on Cyber-Technologies and Cyber-Systems, CYBER 2019, September 22, 2019 to September 25, 2019 – Porto, Portugal, Available from: https://www.iaria.org/conferences2019/CYBER19.html/CYBER19.html, accessed October 2022.

[9] CVE, “CVE-2017-0144 - CVE.report,” 2017, Available from: https://cve.report/CVE-2017-144, accessed October 2022.

[10] CVE, “CVE-2017-0145 - CVE.report,” 2017, Available from: https://cve.report/CVE-2017-145, accessed October 2022.

[11] CVE, “CVE-2017-0147 - CVE.report,” 2017, Available from: https://cve.report/CVE-2017-147, accessed October 2022.

[12] CVE, “CVE-2019-0630 - CVE.report,” 2019, Available from: https://cve.report/CVE-2019-630, accessed October 2022.

[13] CVE, “CVE-2019-0633 - CVE.report,” 2019, Available from: https://cve.report/CVE-2019-0633, accessed October 2022.

[14] Fifth Quadrant, “Customer Experience Research, Design & Consulting,” 2017, Available from: https://www.fifthquadrant.com.au/customer-experience-research-design-consulting-fifth-quadrant, 2022, accessed October 2022.

[15] J. Graham, “How to Rapidly Identify Assets at Risk to WannaCry Ransomware and ETERNALBLUE Exploit,” 2017, Available from: https://blog.qualys.com/vulnerabilities-threat-research/2017/05/12/how-to-rapidly-identify-assets-at-risk-to-wannacry-ransomware-and-eternalblue-exploit, accessed October 2022.

[16] J. Hanrahan, “Suspected Conti Ransomware Activity in the Auto Manufacturing Sector,” Dragos Blog 16 March 2022, Available from: https://www.dragos.com/blog/industry-news/suspected-conti-ransomware-activity-in-the-auto-manufacturing-sector/, accessed March 2023.

[17] HC3 Health Sector Cybersecurity Coordination Centre, Office of Information Security, “The Return of Emotet and the Threat to the Health Sector,” June 2, 2022, Available from: https://www.hhs.gov/sites/default/files/return-of-emotet.pdf, accessed March 2023.

[18] A. Hern and S. Gibbs, “What is 'WanaCrypt0r 2.0' ransomware and why is it attacking the NHS?,” the guardian, Saturday 13 May 2017, Available from: https://www.theguardian.com/technology/2017/may/12/nhs-ransomware-cyber-attack-what-is-wanacrypt0r-20, accessed October 2022.

[19] H. Jankensgard, “The Black Swan problem: Risk management strategies for a world of wild uncertainty,” 2022,

John Wiley & Sons Ltd. The Atrium, Southern Gate, Chichester, West Sussex, P019 8sQ, United Kingdom.

[20] JCSC, "Joint Cyber Security Centre", 2022, Available from: https://www.cyber.gov.au/acsc/view-all-content/glossary/joint-cyber-security-centre-jcsc, accessed December 2022

[21] D. Kennedy, J. O'Gorman, D. Kearns, & M. Aharoni, "Metasploit: the penetration tester's guide," 2011, No starch press, 245 8th Street, San Francisco, CA 94103.

[22] L. Kessem, "How did the wannacry ransomware begin?" IBM Security, 26 May 2017, Available from: https://www.quora.com/How-did-the-Wannacry-ransomware-begin, accessed October 2022.

[23] M. Lee, W. Mercer, P. Rascagneres and C. Williams, "Player 3 Has Entered the Game: Say Hello to 'WannaCry'," Talos Intelligence, 12 May 2017, Available from: http://blog.talosintelligence.com/2017/05/wannacry.html, accessed October 2022.

[24] LogRhythm, "A technical analysis of wannacry ransomware, LogRhythm Labs," 16 May 2017, Available from: https://logrhythm.com/blog/a-technical-analysis-of-wannacry-ransomware/, accessed October 2022.

[25] Malwarebytes Labs, "Threat Types", 2023, Available from: https://www.malwarebytes.com/blog/threats, accessed January 2023.

[26] MalwareTech, "Botnet tracker," MalwareTech, 2017, Available from: https://intel.j.com/botnet/wcrypt/?t=1h&bid=all, accessed October 2022.

[27] A. McNeil, "How did the WannaCry ransomworm spread?" Malwarebytes, 19 May 2017, Available from: https://blog.malwarebytes.com/cybercrime/2017/05/how-did-wannacry-ransomworm-spread/, accessed October 2022.

[28] Metasploit, "Metasploit | Penetration Testing Software, Pen Testing Security," 2022, Available from: https://www.metasploit.com/, accessed October 2022.

[29] Microsoft, "Windows SMB Remote Code Execution Vulnerability CVE-2017-0144," 2017, Available from: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2017-0144, accessed October 2022.

[30] Microsoft, "Windows SMB Remote Code Execution Vulnerability CVE-2017-0145," 2017, Available from: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2017-0145, accessed October 2022.

[31] Microsoft, "Windows SMB Information Disclosure Vulnerability CVE-2017-0147," 2017, Available from: https://msrc.microsoft.com/update-guide/vulnerability/CVE-2017-0147, accessed October 2022.

[32] Microsoft, "Windows SMB Remote Code Execution Vulnerability CVE-2019-0630," 2019, Available from: https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2019-0630, accessed October 2022.

[33] Microsoft, "Windows SMB Remote Code Execution Vulnerability CVE-2019-0633," 2019, Available from: https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2019-0633, accessed October 2022.

[34] Microsoft, "Security Update Severity Rating System (microsoft.com)," available from: https://www.microsoft.com/en-us/msrc/security-update-severity-rating-system, accessed March 2023.

[35] P. Muncaster, "Wannacry didn't start with phishing attacks," says Malwarebytes, Infosecurity, 22 May 2017, Available from: https://www.infosecurity-magazine.com/news/wannacry-didnt-start-with-phishing, accessed October 2022.

[36] NIST, "CVE-2017-0144 Detail," 2017, Available from: https://nvd.nist.gov/vuln/detail/CVE-2017-0144, accessed October 2022.

[37] NIST, "CVE-2017-0145 Detail," 2017, Available from: https://nvd.nist.gov/vuln/detail/CVE-2017-0145, accessed October 2022.

[38] NIST, "CVE-2017-0147 Detail," 2017, Available from: https://nvd.nist.gov/vuln/detail/CVE-2017-0147, accessed October 2022.

[39] NIST, "CVE-2019-0630 Detail," 2019, Available from: https://nvd.nist.gov/vuln/detail/CVE-2019-0630, accessed October 2022.

[40] NIST, "CVE-2019-0633 Detail," 2019, Available from: https://nvd.nist.gov/vuln/detail/CVE-2019-0633, accessed October 2022.

[41] D. Palmer, "Emotet: The world's most dangerous malware botnet was just disrupted by a major police operation," ZDNET 27 January 2021, Available from: https://www.zdnet.com/article/emotet-worlds-most-dangerous-malware-botnet-disrupted-by-international-police-operation/, accessed March 2023.

[42] A. Perin, "Emotet Re-emerges with Help from TrickBot," 6 January 2022, Available from: https://blog.qualys.com/vulnerabilities-threat-research/2022/01/06/emotet-re-emerges-with-help-from-trickbot, accessed January 2023.

[43] A. Petcu, "Emotet Malware Over the Years: The History of an Infamous Cyber-Threat," 23 February 2022, Available from: https://heimdalsecurity.com/blog/emotet-malware-history/, accessed January 2023.

[44] Proofpoint, "Q4 2020 Threat Report," 2020, Available from: https://www.proofpoint.com/us/blog/threat-insight/q4-2020-threat-report-quarterly-analysis-cybersecurity-trends-tactics-and-themes, accessed November 2022.

[45] Qualys, "IT Security and Compliance Platform," 2023, Available from: https://www.qualys.com/, accessed January 2023.

[46] A. Rousseau, "WCry/WanaCry ransomware technical analysis," End Game, 14 May 2017, Available from: https://www.endgame.com/blog/technical-blog/wcrywanacry-ransomware-technical-analysis, accessed September 2022.

[47] J. Steinke, B. Bolunmez, L. Fletcher, V. Wang, A. Tomassetti, K. Repchick, S. Zaccaro, R. Dalal, and L. E. Tetrick 2015, "Improving Cybersecurity Incident Response Team Effectiveness Using Teams-Based Research," IEEE Security & Privacy, Multidisciplinary Security July/August 2015 Vol. 13, No. 4, Available from: https://www.researchgate.net/publication/281467215_Improving_Cybersecurity_Incident_Response_Team_Effectiveness_Using_Teams-Based_Research, accessed February 2023.

[48] Symantec, "WannaCry: Ransomware attacks show strong links to Lazarus Group," Symantec Security Response, 22 May 2017, Available from: https://www.symantec.com/connect/blogs/wannacry-ransomware-attacks-show-strong-links-lazarus-group, accessed October 2022.

[49] Symantec, "Ransom.Wannacry", Symantec, 24 May 2017, Available from: https://www.symantec.com/security_response/writeup.jsp?docid=2017-051310-3522-99, accessed September 2017.

[50] Symantec, "WannaCry variant protection details and information", Symantec Support, 26 May 2017, Available from: https://support.symantec.com/en_US/article.INFO4361.html, accessed October 2022.

[51] N.N. Taleb, "Antifragile, things that gain from disorder", Random House, Penguin Random House LLC, New York, 2021.

[52] N. N. Taleb and J. West, "Working With Convex Responses: Antifragility From Finance to Oncology," Tandon School of Engineering, New York University, 2015

[53] Talos, "Talos Threat Intelligence", 2023, Available from: https://talosintelligence.com/ , accessed January 2023.

[54] I. Thomson, "Wannacry: everything you still need to know because there were so many unanswered Qs", The Register, 20 May 2017, Available from: https://www.theregister.co.uk/2017/05/20/wannacry_windows_xp/, accessed October 2022.

[55] Trendmicro, "EMOTET malware resurges with new detections (trendmicro.com)," Available from: https://success.trendmicro.com/dcx/s/solution/1118391-malware-awareness-emotet-resurgence?language=en_US, accessed March 2023.

[56] TQM 2021, "Turning crises into opportunities in the service sector: how to build antifragility in small and medium service enterprises," The TQM Journal, December 2021.

[57] Truesec, "Threat Intelligence Report 2021," 2021, Available from: https://www.truesec.com/hub/report/threat-intelligence-report-2022, accessed January 2022.

[58] Truesec, "Threat Intelligence Report 2022", 2022, Available from: https://www.truesec.com/hub/report/threat-intelligence-report-2022, accessed January 2023.

[59] TWC, "Free vaccinator & vulnerability scanner tools for WannaCry ransomware," 15 May 2017, The Windows Club, Available from: https://www.thewindowsclub.com/free-vaccinator-vulnerability-scanner-tools-wannacry-ransomware, accessed January 2023.

[60] TWC, "WannaCrypt or WannaCry ransomware decryptors are available," The Windows Club, 19 May 2017, Available from: https://www.thewindowsclub.com/wannacrypt-wannacry-ransomware-decryptor, accessed January 2023.

[61] Verizon, "Latest Cybersecurity Risks and Events", Verizon Business, 2023, Available from: https://www.verizon.com/business/en-au/resources/security/cybersecurity-news-and-events/, accessed January 2023.

[62] S. Winterfeld and J. Andress, "The basics of cyber warfare: understanding the fundamentals of cyber warfare in theory and practice", 2013, Elsevier, Inc, United States of America.

Privacy-Preserving User Clustering: The Application of Anonymized Data to Community Detection in Large Organizations

1st Igor Jakovljevic
ISDS
Graz University of Technology
Graz, Austria
e-mail: igor.jakovljevic@cern.ch

2nd Martin Pobaschnig
ISDS
Graz University of Technology
Graz, Austria
e-mail: martin.pobaschnig@student.tugraz.at

3rd Christian Gütl
ISDS
Graz University of Technology
Geneva, Switzerland
e-mail: c.guetl@tugraz.at

4th Andreas Wagner
IT Department
CERN
Graz, Austria
e-mail: andreas.wagner@cern.ch

***Abstract*—This paper is an extension of our previous work on privacy-protected user clusters identification in large organizations. Oversharing exposes risks, such as improved targeted advertising and leakage of sensitive information. Requiring only the bare minimum of data reduces these risk factors, while simultaneously increasing the privacy of each user. Using anonymized data to find communities opens up new possibilities for large organizations under strong data protection regulations. Although related work often focuses on privacy-preserving community detection algorithms, including differential privacy, in this paper the focus was on the anonymized data itself. Channel membership information was used to build a weighted social graph and groups of interest were identified using popular community detection algorithms. Graphs based on channel membership data resembled interest groups within the network satisfactorily but failed to capture the organizational structure. Furthermore, a statistical evaluation and a user study were conducted to measure the performance of the recommender prototype. The statistical evaluation showed promising results, while the user study yielded mediocre satisfaction of the participants and revealed various potential shortcomings and limitations of the recommender system and user dataset retrieved from the notification system.**

Index Terms*—*Data Privacy; Open Data; Large Organizations; Clustering

I. Introduction

This paper is an extension of our previous work on privacy-protected identification of user clusters in large organizations, presented in [1].

Large organizations are estimated to generate a median of 300 terabytes (TB) of data weekly [2]. Data are generated from the use of various methods of communication (chat, email, face-to-face, phone, short message service, social media) between organization members, data sharing tools, internal processes, different hardware units (mobile phones, tablets, laptops, etc.) and more [2]. The publication of these data to be used for analysis and research has been an excellent source of information for researchers, promoting innovation and advancements in various areas and facilitating cooperation between various groups [3][4]. In this context, the term used to describe the data available freely to anyone to use for analysis and research is open data [5]. There have been different initiatives for collaboration based on open data, such as the Netflix Prize, OpenStreetMap, CERN (Conseil européen pour la recherche nucléaire) Open Science Initiative, Open City Initiatives, and more [3][5][6]. The purpose of these projects has been to improve existing technologies and algorithms and facilitate innovation and collaboration [3]. In addition to these projects, organizations internally analyze user behavior and user data and create new or improve existing services, generally relying on continuous user surveys and behavior tracking while invading their privacy [7].

The sharing of personal data that contain identifiers, quasi-identifiers, and sensitive attributes has been identified as a common issue with similar projects [3]. Sensitive and personal data should not be accessed freely; organizations must protect and secure them. To achieve this, organizations usually secure themselves and do not release this type of data. By doing so, the possible benefits available from private data are not explored. To avoid privacy breaches and publish organizational data, multiple data privacy preservation techniques were developed. Most of them are based on pseudo-anonymization or complete anonymization of the data [8]. The use of anonymized private data led to privacy-preserving data analysis methods. These methods offer a way to use private data safely, considering privacy requirements [9].

CERN has always stood for the principles of open data and open science, facilitating collaborative, transparent, and reproducible research and development whose results are publicly

available [6]. One such initiative is the CERN anonymized Mattermost dataset, which contains anonymized user data, relationships between users, organizations, buildings, teams, and channels. The goal of this dataset is to facilitate innovation for channel recommendations, user clustering, feature extractions, and others [10].

This paper, which is an extension of our previous work on privacy-protected identification of user clusters in large organizations, aims to analyze the provided CERN datasets and determine the privacy aspects and attributes that can be used for privacy-sensitive clustering methods and applications in recommender systems [1]. Based on the observations stated above, more specifically, the main research questions are as follows:

- **RQ1:** Which user information can be extracted from the anonymized Mattermost organizational open data?
- **RQ2:** Is it possible to detect user groups without invading user privacy?
- **RQ3:** What is the performance of clustering algorithms when applied to sparse anonymized user data?

The remainder of this paper is organized as follows. Section II covers the literature overview and discusses current topics in privacy-preserving data mining, open data, sparse data, clustering methodologies, and clustering evaluation metrics. In Section III, we discuss and describe the CERN Mattermost dataset. Section IV focuses on the analysis of various clustering methodologies and algorithms on the previously mentioned data and evaluates the best performing algorithms on the notification system dataset. Section V describes the user evaluation and analysis of the application of clustering algorithms on sparse anonymized user data from the notification system. Section VI discusses the findings of the statistical and user evaluation and explains the use of clustering methodologies. We conclude the work in Section VII with a discussion of the research questions and future work.

II. Background and Related Work

A. Networks and Graphs

Networks are defined as interconnected or interrelated chains, groups, or systems and can be found in a variety of areas, such as the World Wide Web, connections of friends, connections between cities, connections in our brain, power line links, and citation links. In essence, a network is a set of interconnected entities, which we call nodes, and their connections, which we call links. The nodes describe all types of entities, such as people, cities, computers, Web sites, and so on. Links define relationships or interactions between these entities, such as connections between people, flights between airports, links between Web pages, connections between neurons, and more. A special type of network is a social network. It is a group of people connected by a type of relationship (friendship, collaboration, or acquaintance) [11].

The data structure commonly used for the representation of networks is called a graph. A graph is defined as a set of connected points, called vertices (or nodes), that are connected via edges, also called links. The set of vertices is denoted as $V = \{v_1, v_2, v_3, \dots\}$, while the set of edges is denoted as $E = \{e_1, e_2, e_3, \dots\}$. The resulting graph G consists of a set of vertices V and a set of edges E that connect them and can be written as $G = (V, E)$. Two vertices connected by an edge are called adjacent or neighbors, and all vertices connected to a vertex are called neighborhood [12].

Graphs have a variety of measures associated with them. These measures can be classified as global measures and nodal measures. Global measures refer to the global properties of a graph, whereas nodal measures refer to the properties of nodes. The most important measures are degree measures, strength measures, modularity measures, and clustering coefficient measures. The degree measure is a nodal. It is the sum of edges connected to a node. The sum of the weights of all edges connected to a node is defined as the strength measure, while the extent to which a graph divides into clearly separated communities (that is, subgraphs or modules) is described by modularity measures [13].

B. Clustering Methods

Fundamental tasks in data mining are clustering and classification, among others. Clustering is applied mostly for unsupervised learning problems, while classification is used as a supervised learning method. The goal of clustering is descriptive, and that of classification is predictive [14].

Clustering is used to discover new sets of groups from samples. It groups instances into subsets using different measures. Measures used to determine similar or dissimilar instances are classified into distance measures and similarity measures. Different clustering methods have been developed, each of them using different principles. Based on research, clustering can be divided into five different methods: hierarchical, partitioning, density-based, model-based clustering, and grid-based methods [14][15].

Hierarchical Methods - Clusters are constructed by recursively partitioning items in a top-down or bottom-up fashion. For example, each item is initially a cluster of its own; then the clusters are merged based on a measure until the desired clusters are formed [15].

Partitioning Methods - These methods typically require a predetermined number of clusters. Items are moved between different predetermined clusters based on different metrics (error-based metrics, similarity metrics, distance metrics) until desired clusters are formed. To achieve the optimal cluster distribution, extensive computation of all possible partitions is required. Greedy heuristics are used for this computation because it is not feasible to calculate all possible partitions under time constraints [14].

Density-Based Methods - These methods are based on the assumption that clusters are formed according to a specific probability distribution. The aim is to identify clusters and their distribution parameters. The distribution is assumed to be a combination of several distributions [16].

Model-based Clustering methods - Unlike the previously mentioned methods, which group items based on similarity

and distance metrics, these methods attempt to optimize the fit between the input data and a given mathematical model [17].

Grid-based methods - The previous clustering methods were data-driven, while grid-based methods are space-driven approaches. They partition the item space into cells disconnected from the distribution of the input. The grid-based clustering approach uses a multiresolution grid data structure. It groups items into a finite number of cells that form a grid structure on which all clustering operations are performed. The main advantage of the approach is its faster processing time [18].

C. Evaluation Metrics

According to the literature, there are two main types of evaluation metrics for recommendation systems; they are statistical accuracy metrics (SAM) and decision support accuracy metrics (DSAM) [19]. SAM methods such as Mean Absolute Error (MAE) evaluate the precision of a recommender system by comparing the predicted values with the actual ratings of the original predictions and ratings [20][21]. DSAM determines the effectiveness of a prediction engine by helping users select relevant items from the available ones. The most common measures are sensitivity, specificity, and precision. Using the right model validation techniques helps to understand the models and estimate the performance of a model [19].

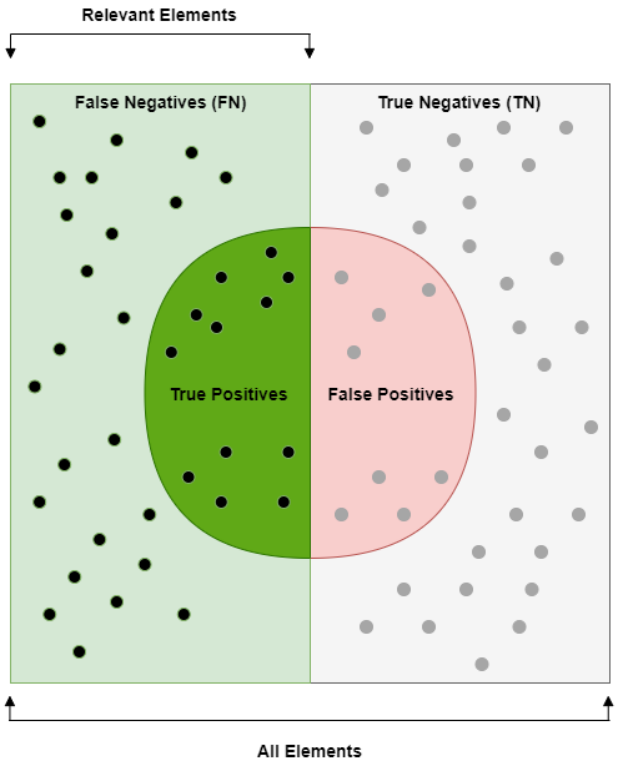


Figure 1. Classification Evaluation Representation based on [22]

Figure 1 illustrates the main elements used for classification evaluation. True positive values are values when the actual and predicted conditions are positive. False positive values are states in which the predicted value is positive, but the actual value is negative. The true negative value indicates that the actual and predicted conditions are the same and are both negative. The state in which the actual condition is positive but the prediction is negative is referred to as false negative values. [23].

Actual positive (AP) values, as seen in equation (1), refer to the number of true positives (TP) together with the number of false positives (FP).

$$AP = TP + FN \tag{1}$$

Actual negative (AN) values, as seen in equation (2), refer to the number of false positives (FP) together with the number of true negatives (TN).

$$AN = FP + TN \tag{2}$$

Predicted positive (PP) values, as seen in equation (3), refer to the number of true positives together with the number of false negatives.

$$PP = TP + FP \tag{3}$$

The predicted negative (PN), as seen in equation (4), refers to the number of false negatives along with the number of true negatives [23].

$$PN = FN + TP \tag{4}$$

Sensitivity describes the ratio of correct predictions to all actual positive conditions and is calculated as shown in equation (5) [24].

$$sensitivity = \frac{TP}{AP} \tag{5}$$

The specificity describes the ratio of correct rejections to all actual negative conditions and is calculated as shown in equation (6) [24].

$$specificity = \frac{TN}{AN} \tag{6}$$

The precision describes the ratio of correct predictions out of all positively predicted values and is calculated as shown in equation (7) [25].

$$precision = \frac{TP}{PP} \tag{7}$$

According to [26], the f-score is a measure of the accuracy of the prediction. It is the harmonic mean between precision and sensitivity and is calculated as shown in Equation (8).

$$f_{score} = 2\frac{precision * sensitivity}{precision + sensitivity} \tag{8}$$

D. Open Data, Sparse Data, and Privacy-aware Data Analysis

Open Data describes data available without restrictions for anyone to use for analysis and research [5]. Open innovation is defined as the use of purpose-oriented inputs and outputs of knowledge to stimulate internal innovation while increasing the demands for external use of innovation, respectively. The goal of open innovation and open data is to increase accountability and transparency while providing new and efficient services [27].

Sparse data is characterized by a relatively high percentage of variables that do not contain actual information. These variables contain values such as "empty" or NA [28]. Sparse data bias is a statistical bias that results from unevenly distributed data. Models trained on sparse data can be biased towards more common observations, leading to poor performance on less common observations. It can occur in unbalanced datasets or when dealing with missing data [29].

Privacy-preserving analytics are a set of methods for collecting, measuring, and analyzing data that respect individual privacy rights. These methods allow data-driven decisions while still giving individuals control over personal data. Restricting access to the data could be found to restrict support for various types of data analysis. Adopting approaches to restricting information in the data so that they are free of identifiers and free of content with a high risk of individual identification. Techniques have been proposed to release data without revealing sensitive information for various applications. Interest in the development of privacy-preserving data mining algorithms has been growing over the years [30].

III. Dataset

The Mattermost dataset was extracted from an internal PostgreSQL (Structured Query Language) database and is accessible as a JSON (JavaScript Object Notation) formatted file [10]. It includes data from January 2018 to November 2021 with 21231 CERN users, 2367 Mattermost teams, 12773 Mattermost channels, 151 CERN buildings, and 163 CERN organizational units. The dataset states the relationships between Mattermost teams, Mattermost channels, and CERN users. It contains various pieces of information, such as channel creation, channel deletion times, user channel joining, and leave times. It also includes user-specific information, such as building and organizational units, messages, and the mention count. To hide identifiable information (e.g., team name, user name, channel name), the dataset was anonymized using a combination of techniques such as omitting attributes, hashing string values, and removing connections between users, teams, and channels. It is important to note that there are various other anonymization techniques, including pseudonymization, differential privacy, and k-anonymity, that could affect the results of privacy-preserving analytics in different ways. The usage of these methods can cause algorithms applied to anonymized datasets to perform differently, since each method introduces a certain level of information loss.

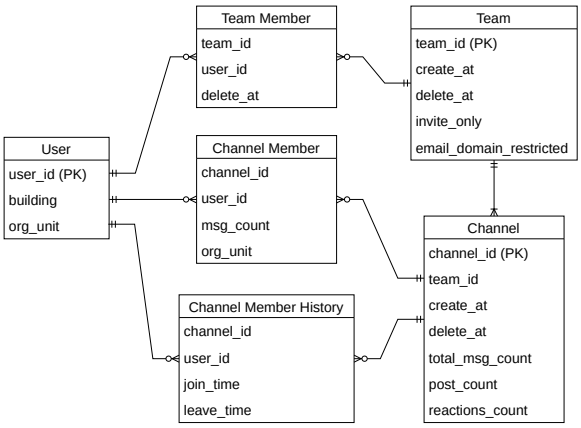


Figure 2. CERN Mattermost dataset Entity Relation Diagram

The entity relationship diagram shown in Figure 2 describes entities with data attributes and relationships between entities.

A. Data Transformation

The dataset was analyzed and prepared to filter out superfluous teams, channels, and users. According to the analysis, approximately 22.6% teams consist of one single person and can be removed as they form isolated nodes that do not contribute to the community structure.

Table I shows the five-number summary of the number of members within teams with more than one member. The five-number summary consists of three quartiles, Q_1, Q_2 or median, and Q_3, which divide the dataset into two parts with the lower part having 25%, 50% and 75% of the dataset's values, respectively. The other two values of the five-number summary consist of the minimum and maximum value of the dataset.

Using the quartiles from the five-number summary, the lower and upper team size fences can be calculated, which act as boundaries above or below which teams are considered outliers. The upper fence can be calculated by $UpperFence = Q_3 + 1.5 * IQR$, where IQR represents the interquartile range. IQR is defined as $IQR = Q_3 - Q_1$. This results in an upper bound of 51.5.

Table I
FIVE-NUMBER SUMMARY OF TEAMS WITH MORE THAN ONE MEMBER.

	Minimum	Q_1	Median	Q_3	Maximum
Team Members	2	4	10	23	4512

When counting the number of teams above that threshold, approximately 87.7% of the teams have less than 52 members. The lower fence is calculated by $LowerFence = Q_1 - 1.5 * IQR$ and yields -24.5. Since we do not have negative team sizes, we can limit the lower bound to 2, since team sizes of 1 are isolated nodes.

B. Graph Creation

Channel membership relations were used to generate graphs that act as a basis for community detection and user group analysis. A weighted edge is added between two users if they share the same channel, and the weight of the edge is increased for each additional channel they share. The idea behind channel membership for graph creation is that team members within CERN join channels related to their organization and work interest. Consequently, the more channels members have in common, the more likely they are to belong to the same organizational structure. The goal is to find the best communities that resemble the CERN organizational structure and communities.

IV. Evaluation

A. Algorithm Evaluation

Following the procedure described in Section III-B with an upper team threshold of 52, a weighted graph was produced. The igraph implementation of the Large Graph Layout (LGL) with 2000 iterations was used to visualize it [31].

LGL was used because it creates good layouts for a large number of vertices and edges and produces well-observable clusters. The graph produced is shown in Figure 3.

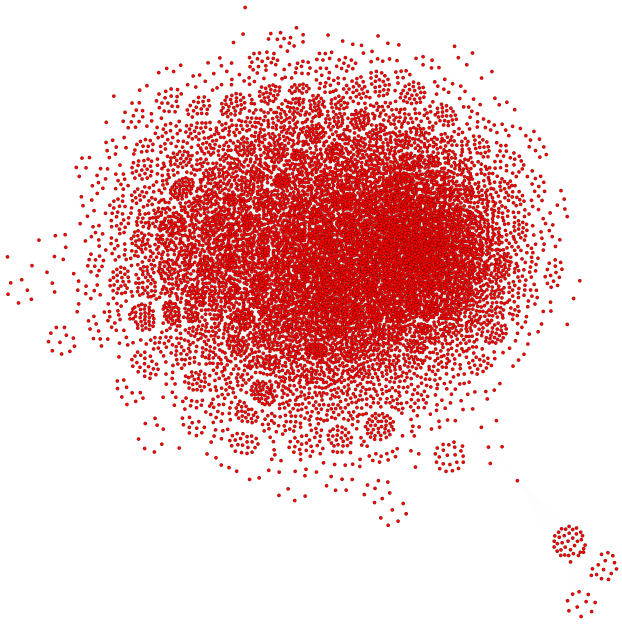


Figure 3. Graph based on channel membership relationship.

Table II lists several advanced clustering algorithms utilized to detect communities in the Mattermost dataset along with the corresponding evaluation results. The mentioned clustering algorithms were selected for evaluation because they were the commonly used algorithms for clustering and also available in the ones in the igraph library. Evaluation metrics considered include modularity, similarity, and the number of communities identified by each algorithm.

Of all the available algorithms presented in Table II, infomap (2), label propagation (3), and random walk (7) delivered the best performance with respect to modularity, similarity, and communities, as shown in Table II.

Random walk algorithms are based on the idea that a random walk on a graph tends to stay within a community and rarely cross over to other communities. It uses a spectral clustering approach to partition the graph into communities by performing a random walk on the graph and using the resulting probability distribution to compute a normalized Laplacian matrix. This algorithm is known for its ability to detect overlapping communities and has been shown to perform well on large-scale networks [38].

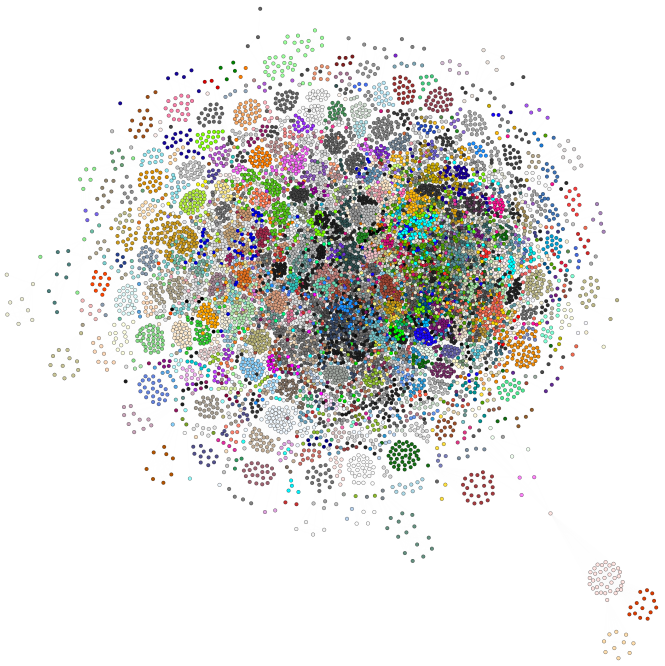


Figure 4. Communities detected by using the label propagation algorithm. A clear separation between individual clusters in the outer part of the graph can be observed.

The infomap algorithm utilizes random walks to assign special codes, known as Huffman codes, to each vertex and organizes them in a way that minimizes the description length measured in bits per vertex. These Huffman codes are binary strings assigned to objects based on their frequency, ensuring that objects visited more frequently are assigned shorter encodings, while less frequently visited objects receive longer ones. This algorithm has demonstrated its effectiveness in detecting hierarchical community structures and is widely recognized for its high accuracy [33].

The Label Propagation Algorithm begins by assigning a unique community label to each node in the network. These labels are propagated through the network iteratively. During each iteration, a node updates its label to the one that the majority of its neighbours have. The algorithm continues to propagate labels until convergence, where each node adopts the majority label of its neighbours or if the maximum number of iterations specified is reached. As the labels are propagated, densely connected nodes quickly reach a consensus on the

Table II
RESULTS INCLUDING FIVE-NUMBER SUMMARY OF SIMILARITIES BETWEEN MATTERMOST TEAMS AND FOUND COMMUNITY WITH DIFFERENT ALGORITHMS. VALUES WITHIN COLUMNS REPRESENT THE MEAN AND STANDARD DEVIATION OVER 25 ITERATIONS.

Algorithm	Communities	Modularity	Minimum [%]	Q_1 [%]	Median [%]	Q_3 [%]	Maximum [%]
1. Community structure through greedy optimization of modularity [32]	41 ± 0	0.75 ± 0.00	7.85 ± 0.00	23.43 ± 0.00	45.24 ± 0.00	66.67 ± 0.00	100 ± 0.00
2. Infomap community finding [33]	414 ± 3	0.71 ± 0.00	18.13 ± 1.18	46.52 ± 0.19	61.75 ± 0.68	75.97 ± 0.61	100.00 ± 0.00
3. Finding communities based on propagating labels [34]	463 ± 8	0.70 ± 0.00	15.68 ± 2.23	48.18 ± 1.07	61.25 ± 0.81	75.08 ± 0.28	100.00 ± 0.00
4. Community structure detecting based on the leading eigenvector of the community matrix [35]	43 ± 0.00	0.67 ± 0.00	5.85 ± 0.00	15.17 ± 0.00	26.92 ± 0.00	52.48 ± 0.00	95.65 ± 0.00
5. Finding community structure of a graph using the Leiden algorithm [36]	1290 ± 3	0.64 ± 0.00	2.04 ± 0.00	20.00 ± 0.00	42.86 ± 0.00	66.67 ± 0.00	100.00 ± 0.00
6. Finding community structure by multi-level optimization of modularity [37]	40 ± 2	0.78 ± 0.00	8.80 ± 0.77	14.79 ± 1.12	21.75 ± 1.64	50.87 ± 6.80	86.51 ± 6.57
7. Computing communities using random walks [38]	344 ± 0	0.72 ± 0.00	8.33 ± 0.00	55.56 ± 0.00	66.67 ± 0.00	80.00 ± 0.00	100.00 ± 0.00
8. Community detection based on statistical mechanics [39]	25 ± 0	0.77 ± 0.00	8.10 ± 0.71	11.23 ± 0.79	14.06 ± 1.05	17.700 ± 1.39	31 ± 8.51

label, leading to the disappearance of many labels. At the end of the propagation, only a few labels remain, and nodes with the same label are considered to belong to the same community. This algorithm is known for its simplicity, efficiency, and scalability [34] .

Calculating the community structure with the highest modularity value (community_optimal_modularity) and community structure detection based on edge betweenness (community_edge_betweenness) were not feasible in practice, since the runtime was too long. Figure 4 displays the result of the label propagation algorithm applied to the graph created previously. Each community is assigned a unique color to observe the separation of individual clusters. The label propagation algorithm finds communities with slightly less similarity than the infomap algorithm, which performs best with respect to similarity measurement. However, it finds many and much more detailed communities.

Figure 5 represents the similarities of the users between the communities found and the Mattermost teams while Figure 6 illustrates the results of 10 iterations as violin plots. An upper threshold of 52 for the teams was used for this figure, as described later in this section.
Of all communities detected, 75% have similarities above 47.79%, 50% have similarities above 61.18%, and 25% have similarities above 74.99%. Similarities are measured by comparing the discovered community with all Mattermost teams and counting the common members in both sets. The percentage value of the Mattermost team with the most common members is used.

Depending on the number of communities found, there may be overlaps such that one team fits multiple communities as the best match. This might be the case where the size of communities is smaller than the size of teams, such that communities form subgroups of the teams. However, less than 0.01% of the communities discovered correspond to the same Mattermost team. The average size of the communities discovered is 20 ± 23, the minimum is 2, the first quartile Q_1 is 6, the median is 13, the third quartile Q_3 is 26, and the maximum is 421.

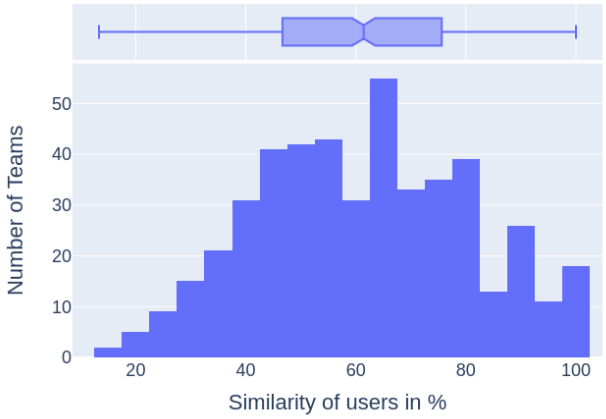


Figure 5. Sample run showing similarities of users between found communities and Mattermost teams.

Figure 7 shows the similarities of users between the detected communities and the organizational units with a threshold of 52, and Table III shows the parameters of this figure in detail. We can observe that the similarities are relatively low, with 75% of communities having at most 5.07% similarity. This indicates that the discovered communities generally do not resemble organizational units very well. The main reason is that Mattermost teams often consist of members of different organizational units. This is especially the case where users form groups of interest that are not related to work. This results in discovered communities that capture the teams and structure

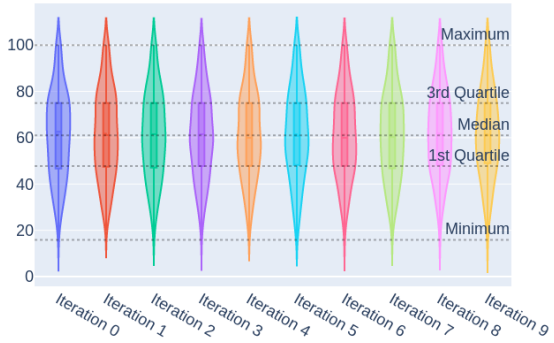


Figure 6. Similarities between discovered communities and Mattermost teams over iterations with threshold 52.

within Mattermost rather than the organizational structure of CERN.

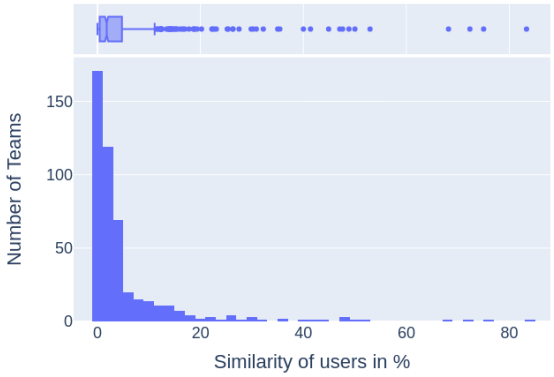


Figure 7. Sample run showing similarities of users between found communities and organizational units.

When creating the graph, two different methods were used and compared for filtering teams and channels. With the first method, the threshold was used as an upper limit for team members, i.e., only the channels of the teams below the threshold are considered for creating the graph.

Table III
FIVE-NUMBER SUMMARY OF SIMILARITIES BETWEEN ORGANIZATIONAL UNITS AND DISCOVERED COMMUNITIES USING LABEL PROPAGATION ALGORITHM. VALUES WITHIN COLUMNS REPRESENT MEAN AND STANDARD DEVIATION OVER 25 ITERATIONS IN PERCENT.

Minimum	Q_1	Median	Q_3	Maximum
0.0 ± 0.0	0.42 ± 0.04	1.77 ± 0.04	5.07 ± 0.29	74.68 ± 4.55

Due to the random nature of the label propagation algorithm, the results of each run differ slightly. The mean and standard derivation over 25 runs were calculated to obtain more precise results. With the second method, the threshold was used as an upper limit for channel members, i.e., all channels below the threshold are considered for creating the graph. The second method yields more nodes, but fewer communities, and slightly less similarity to the first. Due to this, the first method was preferred.

B. Statistical Evaluation

For the statistical evaluation, the user-channel subscription information of the most recent snapshot of the notification system database was collected. The dataset was then divided into a training set and a validation set. For a correct recommendation, an edge contained in the validation set should be in the list of recommendations for a given user. This is a classification problem as described in Section II-C. The graph is created from the training set and communities are discovered. The entire dataset contains 1270 user-channel edges. When splitting them, 1016 edges fall into the training set, while the remaining 254 edges fall into the validation set. However, there are some points that need to be considered:

1) Some users might only be part of one channel. If they are put in the validation set, they will never be recommended.
2) Some users are removed when creating the graph, as they might only be in groups above the upper user limit. If they are removed from the graph, the user-channel edges of the validation set will not be recommended.
3) There are recommendations that might be justified even though they are not in the validation set.

The first two points can be addressed while creating the graph and the recommendations, while the third point cannot. However, the third point has a direct influence on the results, as many of these recommendations fall into the false positive group, directly influencing metrics such as precision and f-score. The results of the statistical evaluation are shown in Table IV.

Table IV
RESULTS OF THE STATISTICAL EVALUATION.

Algorithm	Label Propagation		Infomap	
	Mean	Standard Deviation	Mean	Standard Deviation
True positive	42	6	40	6
True negative	35026	1484	35741	1095
False positive	3320	1484	2605	1096
False negative	12	4	13	4
Specificity	0.91	-	0.93	-
Sensitivity	0.78	-	0.75	-
Precision	0.012	-	0.015	-
F-Score	0.024	-	0.029	-

The results show a correct hit rate (Table IV - Sensitivity) of 78% for the Label Propagation algorithm and 75% for the Infomap algorithm. The correct rejection rate (Table IV - Specificity) for the Label Propagation algorithm is 91% and 93% for the Infomap algorithm. The number of false positives is high due to the user-channel edges that are in the list of recommendations, but not in the validation set.

V. User Study

For the user study, specific users of the CERN notification system were invited to participate by joining a particular channel and their user data was collected for community detection. Their data was anonymized to respect their privacy. Then, the user-channel subscription information of the most recent snapshot of the notification system database was collected and used to create the user graph and discover communities. The results of the statistical evaluation in Section IV were used to select the algorithm to group the users and create graphs. Figure 8 shows the graph created from the user-channel information. The colors represent the communities, and the numbers on the nodes represent the selected anonymized users from CERN. Since the CERN IT department was the first to fully adopt the new notification system, most of the chosen users come from this department. This can also be seen in Figure 8, where most of the users are grouped together into the same community.

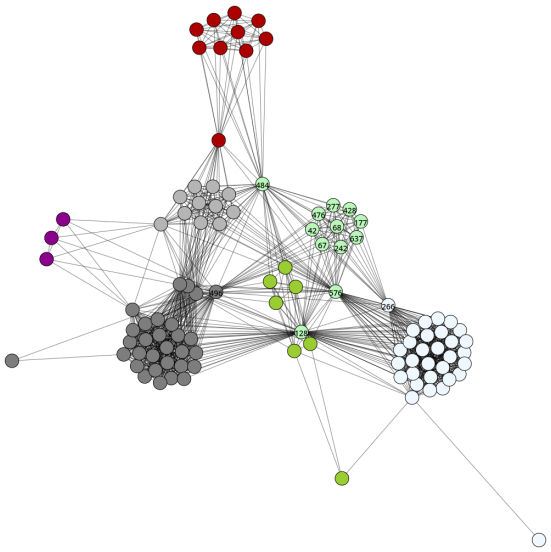


Figure 8. Graph with communities of the most recent user-channel information.

For each community, a list of channels is created to which the community members subscribed and the number of community members in each channel is enumerated. The list of channels is sorted by popularity in descending order, where the first channel contains most of the community members. This list acts as a list of recommendations for the community. The recommendations are created for each notification system user by choosing the first five channels of the most popular ones in the community to which the user has not already subscribed. The survey containing the recommended channels was sent to all participants through the notification system. Each participant was asked to rate the recommendations as personally relevant or irrelevant.

The survey was sent to 15 users of the notification system. Of the initial users, 13 responded, making the response rate 86.66%. Table V shows the results of the user study. Users were marked as active or inactive depending on their interaction with the system.

Table V
RESULTS OF THE USER EVALUATION.

User Id	Active	Relevant Channels	Irrelevant Channels	Precision
128	x	1	4	0.333
476	x	3	2	0.6
484	x	5	0	1
496	x	1	4	0.2
42		3	2	0.6
67		1	4	0.2
68		3	2	0.6
177		1	4	0.2
242		4	1	0.8
266		1	4	0.2
428		2	3	0.4
576		4	1	0.8
637		3	2	0.6

Active users are long-time members who use the system on a daily basis. While inactive users are users who recently joined the notification system or users who do not use the system daily.

The average precision on all participants for relevant and irrelevant channels is 50%, with no significant impact on user activity. Figure 9 shows the results as a graphical representation on a graph.

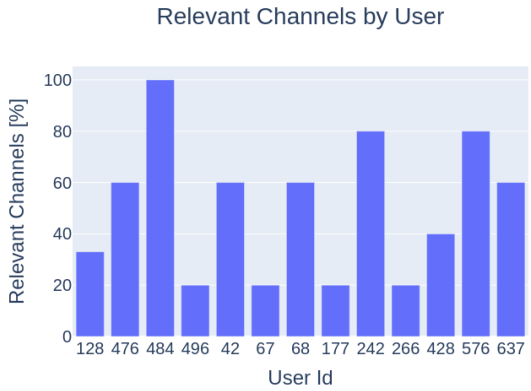


Figure 9. Relevant channels by user as shown in table V.

VI. Findings and Discussion

Based on Section IV with a higher threshold, more users are within teams and channels, increasing the edge weight between many different users. Due to this, the weight difference between the edges of the communities within and outside becomes smaller, resulting in fewer communities. Table VI shows the number of users, edges and the average and standard deviation of edge weights over different thresholds. Higher thresholds result in more nodes and edges, but the average weight decreases, as many users are only part of a few channels and teams. Without a threshold, the average weight increases due to channels increasing the weight for numerous users.

Table VI
NUMBER OF NODES, EDGES, AND AVERAGE AND STANDARD DEVIATION OF EDGE WEIGHTS OVER DIFFERENT THRESHOLDS.

Threshold	Nodes	Edges	Weight
52	9520	151501	2.94 ± 2.35
200	14906	809012	2.82 ± 2.25
500	17124	1909964	2.65 ± 1.88
1000	17948	3104814	2.53 ± 1.66
1500	18721	5000668	2.34 ± 1.58
None	19682	15194697	2.44 ± 1.62

Higher thresholds do not improve community discovery, as the typical size of teams is up to 52, as previously stated. On the basis of our experiments, the clustering tendency depicted by the modularity value decreased with higher thresholds and fewer communities were found.

The results of the user study show mediocre performance in terms of relevance, which, however, might not be a good indication of performance due to various factors that affect the outcome:

1) Low number of participants
2) Low number of interactions between participants
3) Participants are primarily from the IT department
4) High chance that participants already subscribed to channels that are relevant to them
5) Low channel diversity across the whole notification system

Since these points mentioned above can only be addressed when the notification system has more diverse channels and a higher number of users, especially from other departments, the user study gives a good first impression of the prototype.

VII. Conclusion and Future Work

Data privacy preservation is one of the key issues in open innovation and open data. This research aims to analyze the provided CERN dataset and determine privacy aspects and attributes that can be extracted and used for privacy-protected identification of user clusters in large organizations. Information such as user group matching has been the focus of this research. Different clustering algorithms were used for user group detection without invading user privacy. To achieve this, only user communication and interaction data from the CERN Mattermost dataset was used for cluster formation. The dataset includes 21231 CERN users, 2367 Mattermost teams, 12773 Mattermost channels, 151 CERN buildings, and 163 CERN organizational units. It was expected to rediscover an organizational structure that closely matches the organizational hierarchical structures (Organizational Units, Departments, Groups, Sections, etc.). Our research shows that fitting detected clusters to existing organizational structures was not successful and yielded poor results. Matching detected clusters with interest groups, such as Mattermost teams, produced satisfactory results. The main reason for this finding is that users interact and communicate with individuals who share their interests (the same channels or Mattermost teams). These individuals might not be in the same organizational units, or users from different organizational units might be in the same channel, introducing noise to the data.

The algorithm evaluation results also showed that the clustering tendency depicted by the modularity value decreased with higher thresholds and fewer communities were found. In addition, new metrics for weighting user-to-user connections could be used to identify not only interest groups, but also organizational connections between users.

Furthermore, the findings of the analysis of the CERN Mattermost dataset were applied to a new dataset retrieved from the CERN notification system. Since this dataset resembled the Mattermost dataset, it was expected that the clustering algorithms produce similar results on this dataset. The user study showed that the average precision of the best-performing clustering algorithm is 50%. The decrease in performance could be a product of the high level of sparsity in the dataset, the low number of existing channels to recommend, and the high level of users already subscribed to existing channels.

Future work might include the use of novel neural network-based clustering algorithms. Additionally, new metrics for weighting user-to-user connections could be used to identify not only interest groups, but also organizational connections between users. In addition to these improvements, the data could be connected to external data to identify certain teams, users, or organizational structures, and the level of communication between them.

To fully evaluate the effectiveness of channel recommendations, it would be beneficial to provide a baseline of relevant channels. The baseline would be formulated per user by proposing random channels to users and checking their relevance. This would allow for a better understanding of the impact of the recommendation and provide better means of evaluation. We suggest that future research includes such an evaluation when the notification system is fully adopted by other groups at CERN and when more users engage with the system.

References

[1] I. Jakovljevic, M. Pobaschnig, C. Gütl, and A. Wagner, "Privacy aware identification of user clusters in large organisations based on anonymized mattermost user and channel information", in *Proceedings of the 11th International Conference on Data Science, Technology and Applications - IARIA DATA ANALYTICS*, 2022, pp. 62–67, ISBN: 978-1-61208-994-2. [Online]. Available: https://www.thinkmind.org/index.php?view=article&articleid=data_analytics_2022_2_80_60050.

[2] I. Jakovljevic, A. Wagner, and C. Gütl, "Open search use cases for improving information discovery and information retrieval in large and highly connected organizations", 2020. DOI: 10.5281/zenodo.4592449. [Online]. Available: https://doi.org/10.5281/zenodo.4592449.

[3] J. Zhang, Y. Wang, Z. Yuan, and Q. Jin, "Personalized real-time movie recommendation system: Practical prototype and evaluation", *Tsinghua Science and Technology*, vol. 25, pp. 180–191, Apr. 2020. DOI: 10.26599/TST.2018.9010118.

[4] G. Navarro-Arribas, V. Torra, A. Erola, and J. Castellà-Roca, "User k-anonymity for privacy preserving data mining of query logs", *Information Processing Management*, vol. 48, no. 3, pp. 476–487, 2012, Soft Approaches to IA on the Web, ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2011.01.004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306457311000057.

[5] S. Antony and D. Salian, "Usability of open data datasets", in Oct. 2021, pp. 410–422, ISBN: 978-3-030-89021-6. DOI: 10.1007/978-3-030-89022-3_32.

[6] K. Naim *et al.*, "Pushing the Boundaries of Open Science at CERN: Submission to the UNESCO Open Science Consultation", Jul. 2020. DOI: 10.17181/CERN.1SYT.9RGJ. [Online]. Available: http://cds.cern.ch/record/2723849.

[7] P. Rao, S. Krishna, and A. Kumar, "Privacy preservation techniques in big data analytics: A survey", *Journal of Big Data*, vol. 5, pp. 1–12, Sep. 2018. DOI: 10.1186/s40537-018-0141-8.

[8] I. Jakovljevic, C. Gütl, A. Wagner, and A. Nussbaumer, "Compiling open datasets in context of large organizations while protecting user privacy and guaranteeing plausible deniability", *In Proceedings of the 11th International Conference on Data Science, Technology and Applications (DATA 2022)*, pp. 301–311, 2022, ISSN: 2184-285X. DOI: 10.5220/0011265700003269.

[9] S. R. M. Oliveira and O. R. Zaiane, "A privacy-preserving clustering approach toward secure and effective data analysis for business collaboration", *Computers Security*, vol. 26, no. 1, pp. 81–93, Feb. 2007, ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2006.08.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404806001222.

[10] I. Jakovljevic, C. Gütl, A. Wagner, M. Pobaschnig, and A. Mönnich, "Cern anonymized mattermost data", version 1, Mar. 2022. DOI: 10.5281/zenodo.6319684. [Online]. Available: https://doi.org/10.5281/zenodo.6319684 (visited on 06/27/2022).

[11] F. Menczer, S. Fortunato, and C. A. Davis, *A first course in network science*. Cambridge University Press, 2020, ISBN: 9781108471138. DOI: 10.1017/9781108653947. [Online]. Available: https://www.cambridge.org/highereducation/books/first-course-in-network-science/EE22722F27519D8BB1443C7225C57BAF.

[12] V. Voloshin, *Introduction to Graph and Hypergraph Theory*. Nova Kroshka Books, Jan. 2013, p. 231, ISBN: 978-1606923726.

[13] L. Tang and H. Liu, *Community Detection and Mining in Social Media*. Morgan Claypool Publishers, Jan. 2010, vol. 2. DOI: 10.2200/S00298ED1V01Y201009DMK003. [Online]. Available: https://www.morganclaypool.com/doi/abs/10.2200/S00298ED1V01Y201009DMK003 (visited on 08/14/2022).

[14] L. Rokach and O. Maimon, "Clustering methods", in *Data Mining and Knowledge Discovery Handbook*. Springer US, 2005, pp. 321–352, ISBN: 978-0-387-25465-4. DOI: 10.1007/0-387-25465-X_15. [Online]. Available: https://doi.org/10.1007/0-387-25465-X_15.

[15] C. Hennig, "An empirical comparison and characterisation of nine popular clustering methods", *Advances in Data Analysis and Classification*, vol. 16, no. 1, pp. 201–229, Mar. 2022, ISSN: 1862-5355. DOI: 10.1007/s11634-021-00478-z. [Online]. Available: https://doi.org/10.1007/s11634-021-00478-z (visited on 06/27/2022).

[16] J. D. Banfield and A. E. Raftery, "Model-based gaussian and non-gaussian clustering", *Biometrics*, vol. 49, no. 3, pp. 803–821, 1993, ISSN: 0006341X, 15410420. [Online]. Available: http://www.jstor.org/stable/2532201 (visited on 06/27/2022).

[17] P. D. McNicholas, "Model-based clustering", *Journal of Classification*, vol. 33, no. 3, pp. 331–373, Oct. 2016, ISSN: 1432-1343. DOI: 10.1007/s00357-016-9211-9. [Online]. Available: https://doi.org/10.1007/s00357-016-9211-9 (visited on 06/27/2022).

[18] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2012, ISBN: 0123814790. [Online]. Available: http://www.amazon.de/Data-Mining-Concepts-Techniques-Management/dp/0123814790/ref=tmm_hrd_title_0?ie=UTF8&qid=1366039033&sr=1-1.

[19] A. Bobic, I. Jakovljevic, C. Gütl, J. Le Goff, and A. Wagner, "Implicit user network analysis of communication platform open data for channel recommendation", in *9th International Conference on Social Networks Analysis, Management and Security - SNAMS 2022*, Apr. 2022, pp. 1–8. DOI: 10.1109/SNAMS58071.2022.10062597. [Online]. Available: https://ieeexplore.ieee.org/document/10062597.

[20] N. Good *et al.*, "Combining collaborative filtering with personal agents for better recommendations", in *Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence*, American Association for Artificial Intelligence, 1999, pp. 439–446, ISBN: 0262511061. [Online]. Available: https://dl.acm.org/doi/10.5555/315149.315352.

[21] T. Chai and R. Draxler, "Root mean square error (RMSE) or mean absolute error (MAE)?", *Geoscientific Model Development*, vol. 7, no. 3, pp. 1247–1250, Jan. 2014. DOI: 10.5194/gmdd-7-1525-2014. [Online]. Available: https://gmd.copernicus.org/articles/7/1247/2014/.

[22] "Sensitivity and specificity". (Jan. 2023), [Online]. Available: https://en.wikipedia.org/wiki/Sensitivity_and_specificity.

[23] T. Fawcett, "An introduction to ROC analysis", *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006, ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2005.10.010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016786550500303X.

[24] W. Mumtaz, S. S. Ali, A. Mohd Zasin, and A. Malik, "A machine learning framework involving eeg-based functional connectivity to diagnose major depressive disorder (MDD)", *Medical biological engineering computing*, vol. 56, Jul. 2017. DOI: 10.1007/s11517-017-1685-z.

[25] T. de Greef, S. Masroor, M. A. Peletier, and R. Pendavingh, "Precision and sensitivity in detailed-balance reaction networks", *SIAM Journal on Applied Mathematics*, vol. 76, no. 6, pp. 2123–2153, 2016. DOI: 10.1137/15M1054869. eprint: https://doi.org/10.1137/15M1054869. [Online]. Available: https://doi.org/10.1137/15M1054869.

[26] N. Ye, K. M. A. Chai, W. S. Lee, and H. L. Chieu, "Optimizing f-measures: A tale of two approaches", in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, Omnipress, 2012, pp. 1555–1562, ISBN: 9781450312851. [Online]. Available: http://dblp.uni-trier.de/db/conf/icml/icml2012.html#NanCLC12.

[27] J. West, A. Salter, W. Vanhaverbeke, and H. Chesbrough, "Open innovation: The next decade", *Research Policy*, vol. 43, no. 5, pp. 805–811, Jun. 2014, ISSN: 0048-7333. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0048733314000407 (visited on 08/14/2022).

[28] Oracle. "Oracle9i olap services". (2023), [Online]. Available: https://docs.oracle.com/cd/A91034_01/DOC/olap.901/a86720/esdatao6.htm.

[29] S. Greenland, M. A. Mansournia, and D. G. Altman, "Sparse data bias: A problem hiding in plain sight", *British Medical Journal*, vol. 353, Apr. 2016. DOI: 10.1136/bmj.i1981. [Online]. Available: https://www.bmj.com/content/352/bmj.i1981.

[30] I. Pramanik *et al.*, "Privacy preserving big data analytics: A critical analysis of state-of-the-art", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 11, pp. 207–218, Jan. 2021. DOI: https://doi.org/10.1002/widm.1387. [Online]. Available: https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1387 (visited on 08/14/2022).

[31] A. Adai, S. Date, S. Wieland, and E. Marcotte, "Lgl: Creating a map of protein function with an algorithm for visualizing very large biological networks", *Journal of molecular biology*, vol. 340, pp. 179–90, Jul. 2004. DOI: 10.1016/j.jmb.2004.04.047. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022283604004851.

[32] M. Newman and M. Girvan, "Finding and evaluating community structure in networks", *Physical Review E*, vol. 69, no. 2, p. 026113, Feb. 2004, ISSN: 1539-3755, 1550-2376. DOI: 10.1103/PhysRevE.69.026113. [Online]. Available: http://www.cse.cuhk.edu.hk/~cslui/CMSC5734/newman_community_struct_networks_phys_rev.pdf.

[33] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure", *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, Jan. 29, 2008, ISSN: 0027-8424, 1091-6490. DOI: 10.1073/pnas.0706851105. arXiv: 0707.0609. [Online]. Available: http://arxiv.org/abs/0707.0609.

[34] A. Rezaei, S. M. Far, and M. Soleymani, "Near linear-time community detection in networks with hardly detectable community structure", *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pp. 65–72, 2015. DOI: 10.1145/2808797.2808903. [Online]. Available: https://doi.org/10.1145/2808797.2808903.

[35] M. E. J. Newman, "Modularity and community structure in networks", *Proceedings of the National Academy of Sciences of the United States of America*, vol. 103, no. 23, pp. 8577–8582, Jun. 2006, ISSN: 0027-8424. DOI: 10.1073/pnas.0601602103. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1482622/.

[36] V. Traag, L. Waltman, and N. J. van Eck, "From louvain to leiden: Guaranteeing well-connected communities", *Scientific Reports*, vol. 9, no. 1, pp. 5233–5233, Dec. 2019, ISSN: 2045-2322. DOI: 10.1038/s41598-019-41695-z. [Online]. Available: http://arxiv.org/abs/1810.08473.

[37] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks", *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, pp. 10008–10020, Oct. 2008, ISSN: 1742-5468. DOI: 10.1088/1742-5468/2008/10/P10008. [Online]. Available: http://arxiv.org/abs/0803.0476.

[38] P. Pons and M. Latapy, "Computing communities in large networks using random walks", *Proceedings of the 20th International Conference on Computer and Information Sciences*, pp. 284–293, 2005. DOI: 10.1007/11569596_31. [Online]. Available: https://doi.org/10.1007/11569596_31.

[39] J. Reichardt and S. Bornholdt, "Statistical Mechanics of Community Detection", *Physical Review E*, vol. 74, no. 1, p. 016110, Jul. 2006, ISSN: 1539-3755, 1550-2376. DOI: 10.1103/PhysRevE.74.016110. [Online]. Available: http://arxiv.org/abs/cond-mat/0603718.

Binding the Battery to the Pass: An Approach to Trustworthy Product Life Cycle Data by Using Certificates Based on PUFs

Julian Blümke
CARISSMA Institute of Electric, Connected and Secure Mobility (C-ECOS)
Technische Hochschule Ingolstadt
Ingolstadt, Germany
e-mail: julian.bluemke@carissma.eu

Hans-Joachim Hof
CARISSMA Institute of Electric, Connected and Secure Mobility (C-ECOS)
Technische Hochschule Ingolstadt
Ingolstadt, Germany
e-mail: hans-joachim.hof@thi.de

***Abstract*—Reusing batteries of electric vehicles in second life is one pillar of the European Union's Green Deal and its derivatives in order to foster the reduction of greenhouse gases. Product life cycle data plays an important role in improving and simplifying the process of finding the most suitable second life application for a used battery. Such data collected throughout the product's life cycle will be summarized in a digital product pass mandatory for future batteries. Having trustworthy data is a key element of the battery pass in order to provide authentic batteries. This paper presents a concept to securely bind the pass to the battery itself by using physical unclonable functions for creating unique cryptographic keys per battery. Inhomogeneities and cell-to-cell variations in a battery pack enable the use of batteries as physical unclonable functions. The approach combines the cryptographic keys derived from the battery with certificates and makes use of Certificate Transparency promoting trust in the issued certificates. Initial security analysis shows that attacks on product life cycle data and certificates as well as the introduction of manipulated and counterfeit batteries can be detected.**

***Index Terms*—physical unclonable function; Certificate Transparency; electric vehicle battery; battery identity; battery pass; cybersecurity.**

I. Introduction

This paper extends [1]. The European Union's (EU) Green Deal aims to reduce greenhouse gases towards net-zero emissions by 2050 [2]. One of the measures is to lower the use of fossil energy in the transportation sector. Electrically driven vehicles foster this goal and are expected to achieve high sales numbers in the upcoming years: The Faraday Institute forecasts a worldwide demand of more than 5 900 GWh in the year 2040 (2020: 110 GWh) [3]. The rise of Electrical Vehicles (EV) is accompanied by an increasing need for high-voltage batteries. However, batteries degrade during usage and charging. They can only be used in an EV until their capacity degraded to 80% [4, 5]. This will result in a large number of dismounted and unusable EV batteries having a negative economical, ecological, and social impact [6]–[8]. However, these batteries may be still fine for other use cases. To support the recycling and reusing of products and materials the EU introduced the Circular Economy Action Plan containing the reuse of batteries as one pillar [9]. Its goal is to set up applications for a battery's second life either as a complete product in a different environment or dismantled in new products.

The new mass market for EV batteries will also encourage the production of counterfeit batteries. Non-certified or non-qualified batteries can introduce safety risks due to deviations from the specifications of genuine products and especially due to cost-savings in risk-reducing controls and management systems [10]. Reduced capacity and lifetime, overheating, and self-ignition, as well as social aspects like underpaid workers and bad working conditions during manufacturing, are examples of likely effects when using counterfeit EV batteries.

Circular economy and the fight against counterfeiting emphasize a need for authentic batteries that we define as the following: trust in the battery's quality, evidence in the correct implementation of the specification, and traceability of the product life cycle enhance the opportunities for second life applications and lower the risk of introducing low quality and dangerous products into the market. Both, the readiness for circular economy and the circulation of only high-quality batteries, shall be regulated within the new EU-regulation about the treatment of (old) batteries [11] introducing the Battery Passport as an electronic record for batteries of EVs, among others.

This paper presents an approach to inherently bind the digital pass to the physical battery by using certificates based on Physical Unclonable Functions (PUF) managed within the method of Certificate Transparency. The following paragraphs introduce the basic techniques of the presented approach.

Battery Passport: As of today the final adoption of the new regulation by the European Council and the European Parliament is still open [12]. However, no significant modifications of the regulation are expected until then. Therefore, the following requirements can be summarized: The Battery Passport shall be unique for each individual battery and shall consist

of data relevant to the battery's model, and static and dynamic data specific to a single battery. The latter shall accommodate performance and durability parameters, the status of the battery, the number of charging and discharging cycles, negative events like accidents, and operating environmental conditions including the temperature and State of Charge (SoC).
The Battery Passport shall be available through an online database. In case of a second life application, the existing data shall be transferred into a new passport and the legacy passport shall be deleted. Test reports shall be available to notified bodies, market surveillance authorities, and the European Commission to enable examination of compliance with the battery-related requirements.

The regulation provides for a physical code as an identifier of a battery. However, we want to solve the identification of a battery by means of PUFs.

Physical Unclonable Function: A PUF uses physical deviations that occur during production to create a unique and unclonable identifier [13]. It is described as a challenge-response-pair (CRP) where a device to be authenticated needs to prove the ownership of the PUF-identifier. According to McGrath et al. [14] a PUF needs to fulfill the following properties: robust, unique, easy to evaluate, challenging to replicate, and impossible to predict. In general, a distinction is made between weak and strong PUFs. Weak PUFs only comprise one or few CRP, which brings the advantage of storing cryptographic keys without requiring non-volatile memory. An example of a weak PUF is the SRAM-PUF: An SRAM memory cell has two stable states that represent 0 and 1. However, before the first write operation has been executed, the cells tend either to 0 or to 1. This undefined state is used to derive a cryptographic key [15].
On the other hand, strong PUFs are defined as having so many CRP that an attacker cannot solve or recover the PUF in a finite time. One example of these types of PUFs is the optical PUF: It consists of a movable laser beam, a scattering medium, and a sprinkle detector (Figure 1). The orientation of the laser beam is the challenge, whereas the sprinkles comprise the response. It is hardly possible to create equal scattering media and therefore, this principle is suitable as a source of randomness [15].

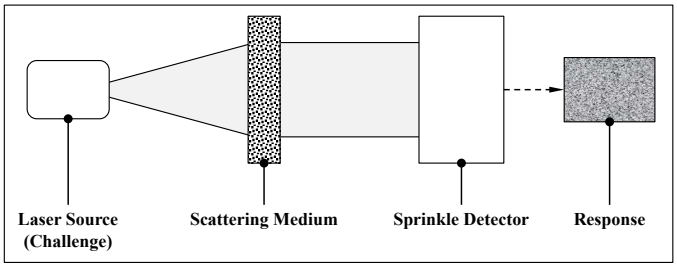


Fig. 1. Process of optical PUF (illustration based on [16])

Regardless of the PUF's type, the general process of deriving cryptographic keys applies to both. Since both the physical source and the measurement process are subject to noise and variabilities, the PUF's response differs slightly, even if the same challenge is used. Therefore, an additional step needs to be done in order to correct these errors [15, 17]. To enhance general security, the random number created from the PUF is hashed using a cryptographic hash function [18]. The process of deriving cryptographic keys from a PUF is shown in Figure 2 in which a fuzzy extractor is used for stabilization and uniformity of the plain PUF response.

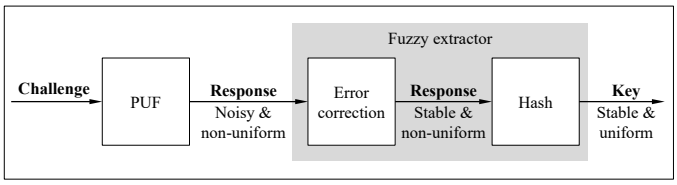


Fig. 2. Process of deriving cryptographic keys from PUF (illustration based on [18])

The two main areas of application of physical unclonable functions are a secret key generation (weak PUFs) and authentication at low cost (strong PUFs) [13]–[15]. The advantages of using PUFs instead of dedicated random number generators are the following: they are simple as they are using existing hardware structures and do not rely on pseudo-random number generators. The secret is only available in a powered mode, which makes it more difficult for attackers to read out the keys, the chance for invasive attacks is reduced, and they are more cost-efficient as they do not need expensive security hardware modules [15].

As introduced later in Section III, the Battery Passport consists of certificates. To enable trust and transparency in the issued certificates, the methods of Certificate Transparency are used.

Certificate Transparency: Certificate Transparency (CT) was originally developed by Google and is about the transparent and trust-worthy issuing of certificates used in the Web PKI [19]. It is summarized in the experimental RFC 6962 [20] and deals with the difficulties of trusting Certificate Authorities (CA) in general: private keys associated with a certificate may be stolen or created in a wrongful way such that encryption itself would not be damaged but an attacker might be able to decrypt the communication without knowledge of the necessary key. A common way to check the trustworthiness of CAs is to examine audits. However, audits often check for formal aspects only than for the correct implementation of technical processes.
The idea of CT is about storing certificates in publicly available append-only logs that can be validated by everyone. Figure 3 shows the steps needed to implement CT: The owner of the domain requests a certificate by the CA, which creates a pre-certificate and sends it to the log. The latter is managed as a Merkle Tree [21]. A Signed Certificate Timestamp (SCT) ensuring that the certificate is added to the log is sent to the CA. The certificate is extended with the SCT and transferred

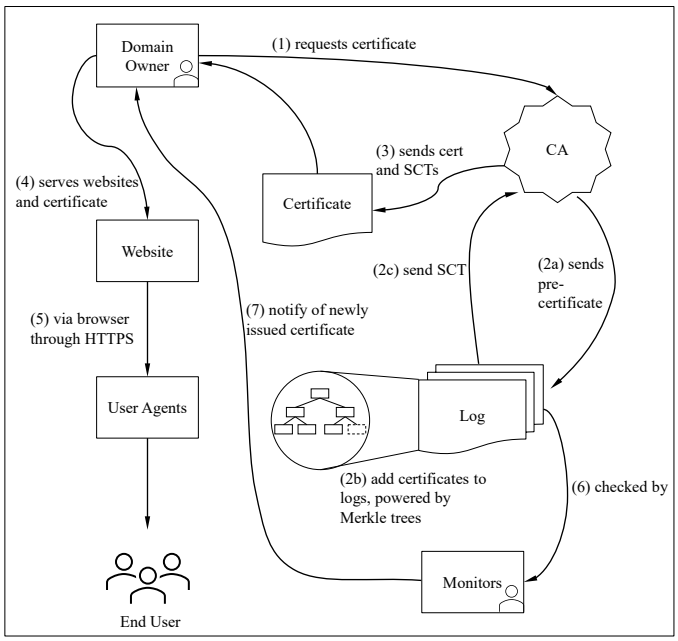


Fig. 3. Implementation of certification transparency (illustration based on [19]).

to the domain owner. From this time on, the domain owner can use it as a normal certificate, e.g., for hosting websites. At the end user's site, the certificate is checked for the existence of SCTs, e.g., during TLS handshake. Some internet browsers require that the certificate is signed with at least two SCTs. The certificate logs are checked periodically by external monitors. The domain owner is informed if there are new and especially odd activities with certificates of its domain.

Furthermore, there are other methods for detecting counterfeit products, e.g., by statistical measures [22], physical inspection, or electrical examination [23]. However, the presented concept is triggered by the EU regulation concerning the battery pass and therefore, the concept of logging and auditing is reasonable.

The remaining paper is structured as followed: Section II describes related work as a basis for a concept for authentic batteries, which is introduced in Section III. A brief security and performance evaluation of the presented approach is given in Section IV. Current and future activities are summarized in Section V.

The conference paper's extensions comprise additional information on the regulation concerning the Battery Passport, a more detailed explanation of PUFs and their characteristics, an overview of research works related to inhomogeneities and cell-to-cell variations in lithium-ion batteries, initiatives on the implementation of the Battery Passport, and an elaborated explanation of the presented approach to combine the physical battery with the Battery Passport.

II. Related Work

To the best of our knowledge, the idea of a digital product pass for single products is unique to batteries. Other applications do have static product records or they are only implemented for a group of products and not for single devices. Additionally, the battery pass will be the first pass that is required by law in the EU. The following related research results introduce only comparable parts of the presented concept.

A. Product Passes

The general concept of product passes is not novel. Several initiatives for passports in other domains already exist. Exemplary three implementations are introduced.

The International Material Data System (IMDS) is a collection of life cycle data in the automotive industry. Original equipment manufacturer (OEM) and supplier store information on component and material data of vehicles in the IMDS to reduce the workload and required time of life cycle assessment. The system has been introduced in the year 2000 and is commonly used by more than 50 OEMs and 120 000 suppliers [24].

Reusing building materials is important for a circular economy in the civil industry. The Building Information Management (BIM) based Material Passport shall enable Urban Mining by comprising data on materials across the full life cycle, e.g., of volume and geometry of multilayered components (concrete, insulation, plaster) or assessment of demolition acquisition [25].

The Danish ship manufacturer and logistic company Maersk created a Cradle-to-Cradle passport for enhanced recycling opportunities at the end-of-life of a container vessel. The document contains information on built materials as well as disassembly and recycling activities. This decreases the need for new materials fostering sustainability and reducing overall costs [26].

These initiatives show that product passes can enhance and simplify the assessment of the life cycle and of potentials for reuse and recycling. However, these product passports do not take into consideration dynamic activities and modifications of the product during life.

B. PUFs based on batteries

In [27], Vittilapuram Subramanian and Madhukar Lele describe the calculation of PUF identifiers out of a set of different parameters: pressure drop between two sides of the battery, the batteries natural frequency, the temperature pattern, the open circuit voltage (OCV), or the air leak rate. The created PUF identifier is saved as a physical tag on top of the battery or in the battery management system's memory. However, the identifier can only be calculated in a dismounted state. This method shows the possibility of a battery PUF creation in general.

Zografopoulos and Konstantinou [28] presented a method to authenticate an outstation in a distributed energy storage network. This work takes advantage of the fact that the cells' voltages differ at the same SoC. Both, the outstation and the

master station, sanitize a challenge-reply-table with continuously updated measurements presenting a model of every cell. The authentication challenge is formed out of a selection of cells. The SoC and the voltages are measured and sent back to the master station. If the actual measurements match with the values in the challenge-reply-table the outstation is accepted as authentic.

Both works demonstrate that it is feasible to use PUFs on batteries. However, existing works use PUFs as a mechanism to create an identity. We want to extend this to use PUFs to derive keys.

C. Inhomogeneities and cell-to-cell variations in lithium-ion batteries

Durbarry et al. [29] state that variations from cell to cell are the origin of a battery's uniqueness. Differences are noticeable in capacity, current, impedance, open circuit voltage, and so in the SoC [30, 31]. The origins of these inhomogeneities can be split into intrinsic and extrinsic influences.

Intrinsic inhomogeneities: As many parts of the used materials are natural products they are subject to variations. For example, the material used for the electrodes differs in composition, purity, defects, and morphology. An identical manufacturing process is also hardly achievable due to its complexity. Differences may also arise with respect to the production volume [32]. Even if the cells are produced identically, variations between them may also result from uneven cell connections [30]–[32].

Extrinsic inhomogeneities: The environment of the battery pack can also have an influence on differences between cells as well as the pack design: Unmatched cells and asymmetric design can result in inhomogeneous cell utilization. The same applies to an ineffective cooling strategy and external heat sources resulting in local temperature peaks. Finally, cells in serial or parallel architecture lead to differences [29]–[32].

The overall result of inhomogeneous parameters leads to inhomogeneous aging of different cell components and this again amplifies the variations of parameters. Aging accelerates over time [30, 31].

These research works show that significant cell-to-cell variations in lithium-ion batteries exist and are measurable. They also demonstrate the effects of aging on these inhomogeneities, which are mentioned later in Section III-E as one major challenge of the presented concept.

D. Blockchain with PUFs

A common mechanism to implement digital product passes is the use of blockchain [33, 34]. Casino et al. described a blockchain as a "distributed append-only timestamped data structure" [35, p. 56] where no central and trusted authority is involved. Exchanging assets, digital or physical, between two blockchain participants is achieved and recorded with transactions. They have to be validated by other participating nodes using a consensus algorithm in order to prevent corruption or forgery of branches. Blockchains in the sector of supply chain management can increase trust, traceability, transparency, and accountability. They are installed for better visibility and enhanced optimization of a supply chain [35].

Mohanty et al. [36] introduced PUFChain, which is a method that combines blockchain with PUFs within the Internet-of-Everything (IoE) domain where trusted nodes authenticate IoE-data collected from client nodes. The process is divided into three phases: During the enrollment, the client's PUF-CRP are calculated and stored in a secure database. The phases of transactions consist of data collection, PUF response generation, and hashing of both. The data and the hash are added to the blockchain and need to be authenticated by trusted nodes. These nodes recalculate the hash by using the client data and the pre-calculated PUF response retrieved from the database and validate the block if both hashes match. An application of PUFChain in the Internet-of-Energy was given by Asif et al. [37].

An approach to enable trust in the supply chain by tracing was presented by Cui et al. [38]. Newly manufactured devices need to be registered in a blockchain with a unique ID, e.g., a PUF. Device transfers are recorded in the blockchain. The contractual ownership alters only after a transfer confirmation, which is done by calculating the unique device ID of the received device and comparing it with the ID mentioned in the transaction payload. End users can check the device's authenticity by matching the computed ID with the blockchain content.

Whereas blockchain is a popular method for storing tamper-proofed data, we decided to use a different approach. In our opinion, the system consists of trusted partners: A generally trusted collaboration across the supply chain of EV batteries has to exist already, meaning that contracts describing a trade relationship are in place. One major motivation for using blockchain methodology is to create a network between parties that do not trust and know each other. Both are not applicable to the application presented in this concept. Therefore, decentralized distribution of data is not necessary and so, a central database fits the requirements and can be hosted, e.g., by the EU enforcing the battery regulations. In a blockchain, consensus mechanisms shall ensure the correctness of new transactions. In this specific application, these mechanisms are useful only to a limited extent as they will only perform a proof of formal attributes of a transaction, i.e. a new record added to the Battery Pass. A blockchain party validating a new block cannot check the validity and legitimacy of, e.g., a new temperature maximum or a degradation of the capacity. This is only possible with direct access to the battery system itself. As a summary, using a blockchain will add extra effort without having additional advantages. The key functionality of generating trusted and tamper-proofed data records can also be achieved by using the methods of Certificate Transparency.

E. Initiatives working on Battery Passport

The new EU regulation concerning the Battery Passport is expected to be mandatory within the next years. Therefore,

several initiatives for the battery pass's implementation exist. Due to the early phase of these projects, only introductory content has been published and so, an assessment of the architectural concepts is hardly possible. However, a brief overview of existing projects is given.

The Global Battery Alliance (GBA) is a collaboration platform consisting of more than 120 partners and organizations and puts the focus on a sustainable supply chain in the battery industry. It was founded at the World Economic Forum (WEF) in 2017. The GBA published proof-of-concept pilots for a Battery Passport at the Annual Meeting of the WEF 2023 [39]. The pilots contain information on a specific battery: EV manufacturer, battery producer, battery cell producer, cell type, chemistry, capacity, and other parameters are shown as well as information on materials including the origin of raw materials. The focus of the pilots is on ESG (environmental, social, and corporate governance) data. Whereas the underlying architecture and processes are not known so far, the proof-of-concept pilots give an insight into a potentially implemented Battery Passport.

A project closely related to the GBA is called *Battery Pass*, which is a consortium formed by companies from the automotive, battery recycling, and data processing industries [40]. It is funded by the German Federal Ministry for Economic Affairs and Climate Action and shall make use of the automotive information exchange system Catena-X. A report [41] guiding through the new EU regulation's content relevant to the battery passport has been published. It introduces the consortium's interpretation of the requirements and outlines an overview of the possible data stored in the Battery Passport. This project aims to give guidelines for interpreting and implementing the EU regulation and therefore, it should be kept in view. A demonstrator is expected to be published in 2024.

III. Concept for authentic batteries

A. Introduction

The general aim of our method is to have one single source of truth containing information about the battery's life cycle including the manufacturing process, product acceptance tests (PAT), measures of quality control, and usage history. Tracing materials and processes fosters consumers' trust in the battery and enables an easier and more precise assessment of the batteries' status for recycling or reusing.

The data of the life cycle record is stored in a database that can be restricted in order to control the read and write access of the supply chain parties involved. Access control also protects the parties' intellectual property (IP). It is mandatory to have a secure binding between the life cycle record and the battery itself ensuring the correspondence between both. The secure binding is established by the use of certificates in combination with PUFs that provide unique identifiers for each battery.

B. Data for battery pass's records

Data is added to the battery pass during manufacturing, product testing, and quality control. This data brings added value to the end user, as the user receives information on a remarkable downgrade of, e.g., the state of health (SoH) or capacity and on minimum and maximum temperatures, voltages, and currents. The latter parameters are important to assess the battery's health for a second life application. The data acquisition building the life cycle record is split into three phases (see Figure 4).

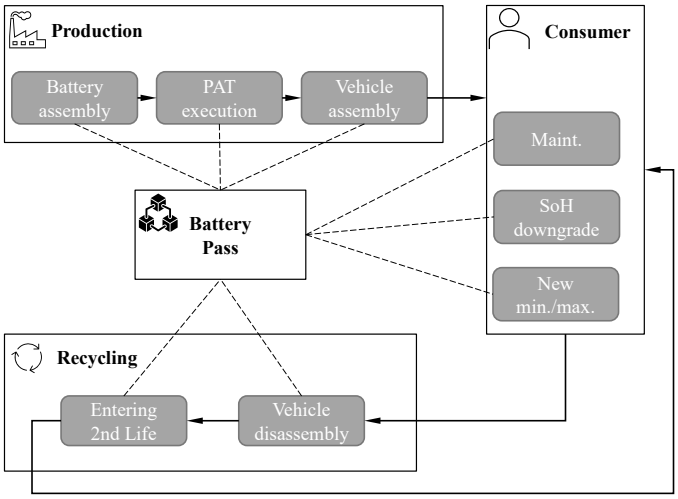


Fig. 4. Battery pass as life cycle record.

Assembly and initial product testing takes place during the production stage at the battery OEM. Information about the manufacturer, working conditions, date of production, and results of acceptance tests are stored in the battery pass. It may be split into battery cell OEM and battery pack OEM having similar data. Afterwards, the battery is transferred to the vehicle's OEM to be built into the intended vehicle. Again, information about the vehicle manufacturer, working conditions, and the vehicle including the vehicle identification number (VIN) are stored in the record. The storage of information concerning the working conditions shall enable a socially acceptable supply chain, which goes along the new EU regulation.

We are assuming the car to be delivered to the consumer directly after production. At this stage, the battery will be used in its intended environment of the first life. Significant changes in the battery's quality will be logged in the life cycle record. These changes include temperature, voltage and current maxima and minima, and SoH and capacity downgrade. The collection of data at this stage is of high relevance in order to execute a sophisticated life cycle assessment of the battery before entering a second life.

The preparation of the second life is divided into two steps: First, the battery is dismounted from the vehicle and the date and the implementing company are stored in the life cycle record. This marks the end of the first life. The activity of entering the second life contains events like firmware or configuration updates required for a new environment or applications, quality tests, and maintenance activities. Again, the battery will be transferred to a consumer. We assume an

environment in which the life cycle record can be sanitized. Therefore, the stage of the second life equals the consumer stage. Depending on the new area of application other or additional data than before may be stored in the battery pass.

The format of the battery pass's data is not defined here. However, the JSON data format may be reasonable as it is widely used and easy to read and process.

C. Security Requirements

The security demands for the presented concept are mainly derived from the high-level requirements of the Battery Pass as presented in Section I. The following security-related aspects shall be considered:

- The battery pass and its records shall be bound to the physical battery. This guarantees that the records are only valid for one specific battery.
- It shall be possible to detect a manipulated battery pass. Shaping of data towards, e.g., less charging cycles or better historic environmental conditions, may increase the resale value of an EV and therefore, these malicious activities must be prevented.
- The circulation of counterfeit batteries having a stolen or no battery pass shall be recognized as well.
- Updates of the records shall only be possible from the battery itself or from a system that has access to the battery. This ensures the validity of the data without the possibility of data being added by a third party not involved in the process.
- Trust and transparency shall be treated to foster the battery pass's acceptance by the user and in general a successful assessment of second life applications.
- It shall be possible to restrict access to the data records of the Battery Pass to a limited number of users or user groups in general. This ensures the enforcement of the GDPR (general data protection regulation) and the protection of intellectual properties.

D. Security Architecture

The technical implementation of our method is based on signed battery data whereas the keys are derived from the battery's PUF. Figure 5 shows the overall process of adding data and verifying the battery's identity. We are assuming the process of deriving a key from the PUF has already been carried out. As elaborated in the section on related work (Section II), this assumption is reasonable.

The general implementation is split into four phases: In the enrollment phase, the keys and an initial certificate are created. It is followed by the creation and storage process of a new data record. In this phase, the battery is the only active part, apart from storing the certificate in the log. In the third phase, the battery is not involved anymore, as the verification of a record can be carried out by using only the entries of the database and the certificate stored in the log. To verify the identity of the battery, the battery must prove possession of the private

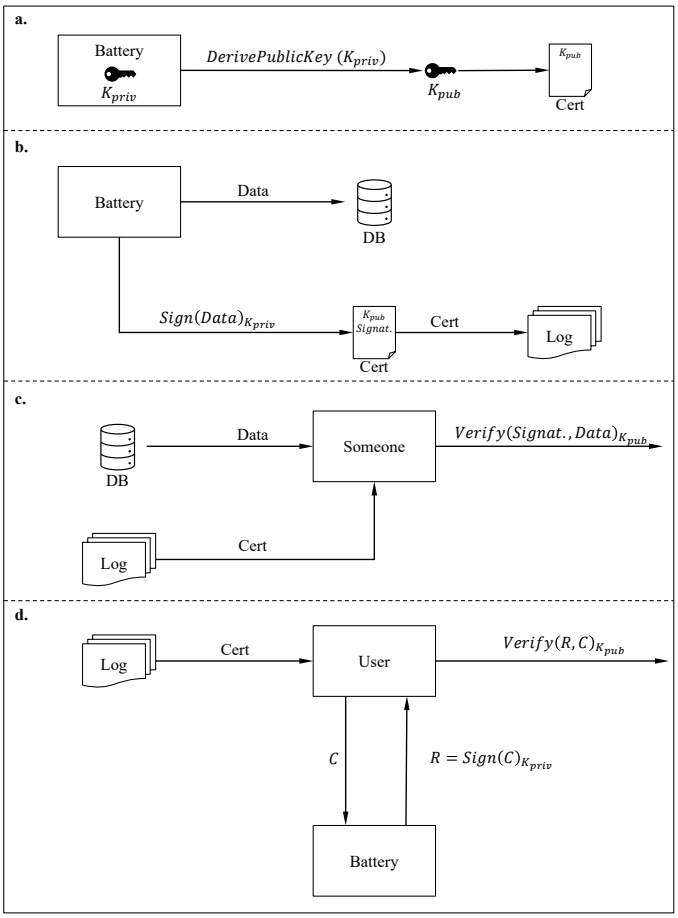


Fig. 5. Implementation of digital pass with certificates. **a.** Enrollment phase **b.** Update of battery records **c.** Verify that the certificate belongs to records **d.** Verify that the battery belongs to the certificate.

key to a user, which is comprised in the fourth phase. The four phases introduced are outlined in the next paragraphs. Except for the first phase, the stages do not have to be processed one after another. The activities can be executed independently.

The first phase is to enroll a private and public key and an initial certificate. As mentioned earlier we assume that a key has been derived from the battery's PUF. This key is the private key and will never leave the battery. A public key is calculated from the private key and added to an initial certificate. At this stage, the certificate may contain only metadata. However, battery-related data will be added in the next step that immediately follows the enrollment phase in order to fill the Battery Pass with relevant data.

The most functional part of the method is adding and updating the data of the battery. The relevant data is described in Section III-B. If new data is generated it will be sent to a central database containing historic and current data of this specific battery (Figure 5b). In the battery or more precisely in the battery management system the data is signed with the private key derived from the battery's PUF. Only the signature is added to a battery-specific certificate also containing the

public key. If a certificate already exists for the battery a reissuing is needed and the old one has to be revoked. The certificate itself is attached to an append-only log. We are relying on Certificate Transparency, which is a commonly used method developed by Google to store and handle identity certificates in a trusted and verifiable way, which has been introduced in Section I. Whereas the log itself does not fulfill any functional requirement, it provides additional trust and transparency into the certificate as it can be validated and monitored by external and public parties.
One could argue to add the battery data to the certificate introducing the advantage of having one single document containing all relevant information about the battery. However, having this, the battery's data is publicly available, and therefore, IP may be revealed as well as the opportunity for malicious analysis of production statistics and performance of a battery OEM. A dedicated database can be restricted to a reduced number of users. This is also in line with the EU regulation concerning the definition of the Battery Pass, presented in Section I. Test reports certifying the quality of the battery shall only be accessible to a certain group of users.

In order to check the validity of the data in accordance with the corresponding certificate, access to the data and the certificate is needed. Using the public key stored in the certificate the signatures can be verified (Figure 5c). In this context, another opportunity to avoid disclosure of IP may be possible by letting the signatures be checked by the database itself and letting it deliver a summary of data not revealing IP.

In the last phase, it is checked that the certificate belongs to the battery as described in Figure 5d. Therefore, a challenge-response mechanism is used where the user sends a challenge consisting of a random number to the battery. The challenge is signed using the battery's private key and the response is sent back to the user. If the received signature can be verified using the corresponding public key it is proven that the certificate belongs to the battery as the private key is directly derived from the battery's PUF.

To reduce the risk of stolen or reproduced keys by an attacker the derived key may be stored in a Hardware Security Module (HSM), e.g., placed on the Battery Management System (BMS). However, the cost-efficiency of HSMs in the context of industrial applications with large quantities having high pressure on costs has to be evaluated [42].

E. Challenges

The main challenge of the presented method is the derivation of keys from the battery's PUF. It is required that the keys do not change over time. However, due to the aging of cells and the battery pack the PUF and so, the keys may change. The validation steps mentioned above cannot be executed anymore resulting in a failure of the complete method. The same applies to genuine repairs or maintenance activities of the battery. Single cells will not be exchanged probably, but battery packs. This would result in a new PUF and so in invalid existing private and public keys.
To overcome both, two approaches might be appropriate: First, using a model forecasting the cell and battery aging in order to create static cryptographic keys. And second, if an imminent change is foreseeable having a mechanism to modify the existing keys, e.g., with pre-calculated challenges and a hash chain for tracking expired keys.
Instead of using the battery's cells to create unique identifier, one could also use the surrounding electrical components as an origin for physical unclonable functions. The entropy might be enough to create cryptographic keys as there are many components built into one battery pack. These components do not age in the same way as cells do.

The creation of the PUF shall only be possible using the measured data available to the BMS installed in the vehicle. Measurements that are only obtainable under laboratory conditions cannot be used for these calculations. However, in-situ measurements reduce the opportunities to retrieve cell-to-cell variations as stated by Prosser et al. [43]. Therefore, it has to be analyzed if it is possible to measure the mentioned inhomogeneities sufficiently within the BMS and if these parameters offer enough entropy to be applied to cryptographic applications.

Challenges also arise in the general use of the battery pass. Standardization across companies is mandatory to enable comparability of batteries. This also applies to the update procedure of the battery pass. Questions concerning the frequency and the resolution of record updates have to be answered.

IV. Evaluation

A brief analysis of the security and efficiency aspects of the presented approach is given in the following section. It is evaluated if the concept meets the requirements outlined in Section III-C. The approach is also examined with regard to its efficiency.

A. Security Analysis

The assumed model of the adversary is presented, followed by an evaluation of the individual requirements.

Adversary Model: We assume that the attacker has read and write access to the database. As the certificates are stored publicly following the methods of Certificate Transparency the adversary can also read certificates. However, the attacker cannot read or re-create the battery's private key as we assume that the physical access to the battery and its related components is restricted or destroys the physical characteristics resulting in a modified PUF.

Binding battery pass and battery: The data of the battery stored in the database is signed using a private key that is derived from the components of the physical battery. The signature itself is saved in a certificate, which is the actual battery pass. Therefore, the physical battery and the battery pass are distinctly linked.

Detection of manipulated battery pass: Manipulation of a battery pass can be possible in two ways: First, manipulation of data in the database and second, manipulation of the certificate. Data manipulation will be recognized if the signature is verified. Signature verification should be a mandatory step when working with these batteries, e.g., for an assessment of the second life applications. A manipulation of the certificate can be detected by the monitoring instances within Certificate Transparency. However, if an attacker can calculate a signature using a key he controls and if he can add the signature and the corresponding public key to the certificate, a manipulated battery pass cannot be detected by only verifying the signature. To overcome this effect, it must be checked if the physical battery, i.e. the private key, belongs to the public key stored in the certificate.

However, in general manipulation or deletion of data can result in financial and ecological damage as it is the basis for further use of the battery. If the data is deleted, assumptions based on statistical measures have to be consulted, which may result in a worse assessment of the state of health.

Circulation of counterfeit batteries: If an attacker duplicates the certificate in order to sell a counterfeit battery with a pseudo-valid certificate, the attack may not be recognized until the link between the certificate and the battery is verified. Whereas the signature for the data is valid, the challenge-response as mentioned in Section III-D will fail: The public key stored in the certificate does not match the private key of the battery as the public key is a derivative of the private key. Therefore, the decrypted response will not match the initial challenge.

Update of battery pass only with access to battery: In theory, records can be added to the database without having access to the battery. As we assume that the attacker has access to the database values can be added or deleted arbitrarily. Even a signature can be created by an attacker. However, the signature cannot be validated correctly as the key used for signing the data does not match the public key used to validate the signature. The signature will be validated correctly only if the private key derived from the battery is used. Therefore, a valid update of the battery pass is only possible with physical access to the battery. Nevertheless, the validation must be done actively and continuously in order to prevent the theoretical opportunity of adding data without access to the battery. The monitoring feature of Certificate Transparency supports this requirement.

Generating trust and transparency: Trust and transparency for user's acceptance and for trustworthy assessment of second life applications is created with the use of cryptographic keys on the one hand and on the other hand with the use of Certificate Transparency where certificates can be validated by external parties.

Several attack scenarios have been described. None of them can be executed on their own as there need to be attacks on multiple system parts to be successful. However, it also showed that a continuous verification of the different links between certificate, data, and battery is mandatory to ensure the system's security. Nevertheless, a complete and in-depth security analysis will be executed in the future to strengthen the given statements.

B. Efficiency of Data Transfer and Verification

In the current EU project MARBEL (Manufacturing and assembly of modular and reusable Electric Vehicle battery for environment-friendly and lightweight mobility [44]) the efficiency of data transfer with a state-of-the-art BMS has been analyzed in a proof-of-concept. Tests have been made with a frequency of data transfer ranging from 5 Hz to 200 Hz sending single MQTT (Message Queuing Telemetry Transport protocol) messages. Authentication and encryption were established using the Transport Layer Security (TLS) protocol adding a security-related overhead to every message. The average message size summed up to 90 bytes, which corresponded to a measured maximum data rate of 144 kBits/s. The findings from these tests appear to support the assumption of an efficient data transfer. However, a continuous stream of battery data might not be required as the degradation of the battery's SoH is a slow process. Data may be also buffered over a defined time and sent in blocks.
Data will be verified on servers that can be highly optimized. Therefore, it is expected that the verification can be carried out efficiently as well.

V. Conclusion and Future Work

Circular economy and the fight against counterfeiting emphasize a need for authentic products. Digital product passes are one example to increase trust and transparency in a product's life cycle. Within the next years, a digital product pass will be mandatory for all EV batteries entering the EU's market. This paper presented an approach to inherently bind the battery with the pass by using certificates with PUFs. Variations from cell-to-cell exist and therefore, it seems feasible to derive cryptographic keys from a battery-based PUF. The certificates are managed and validated within the environment of Certificate Transparency. Challenges arise in the inconsistency of PUFs due to cell aging and in the availability of measurement controls in the BMS. An initial security analysis showed that the presented method enables traceability of and trust in the product life cycle data and detectability of counterfeit products and passes.
Future work includes an analysis of cell parameters usable for a PUF directly retrievable within the BMS. An assessment of the random data in terms of entropy is also to be done as well as further investigations on the consistency of PUFs in the context of EV batteries. The results will be used to create a proof-of-concept followed by a performance and in-depth formal security analysis in order to evaluate the functionality and the security measures of the presented method. Other mechanisms for detecting counterfeit electronic products will be analyzed and set into comparison to PUFs.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 963540.

We want to thank our research colleagues for supporting us during the concept creation. We also want to acknowledge the research group of Prof. Dr. rer. nat. Hans-Georg Schweiger for discussions about the topic of PUFs for batteries.

REFERENCES

[1] J. Blümke and H.-J. Hof, "Authentic Batteries: A Concept for a Battery Pass Based on PUF-enabled Certificates," in *SECURWARE 2022*, G. O. M. Yee, Ed. Wilmington, DE, USA: IARIA, 2022, pp. 77–81.

[2] European Commission, "Regulation (EU) 2021/1119 of the European Parliament and of the Council of 30 June 2021 establishing the framework for achieving climate neutrality and amending Regulations (EC) No 401/2009 and (EU) 2018/1999 ('European Climate Law'): European Climate Law," 2021. [Online]. Available: http://data.europa.eu/eli/reg/2021/1119/oj [Accessed: 01.06.2023]

[3] "Lithium, Cobalt and Nickel: The Gold Rush of the 21st Century." [Online]. Available: https://faraday.ac.uk/get/insight-6/ [Accessed: 01.06.2023]

[4] E. Wood, M. Alexander, and T. H. Bradley, "Investigation of battery end-of-life conditions for plug-in hybrid electric vehicles," *Journal of Power Sources*, vol. 196, no. 11, pp. 5147–5154, 2011.

[5] E. Hossain, D. Murtaugh, J. Mody, H. M. R. Faruque, M. S. Haque Sunny, and N. Mohammad, "A Comprehensive Review on Second-Life Batteries: Current State, Manufacturing Considerations, Applications, Impacts, Barriers & Potential Solutions, Business Strategies, and Policies," *IEEE Access*, vol. 7, pp. 73 215–73 252, 2019.

[6] L. A.-W. Ellingsen, G. Majeau-Bettez, B. Singh, A. K. Srivastava, L. O. Valøen, and A. H. Strømman, "Life Cycle Assessment of a Lithium-Ion Battery Vehicle Pack," *Journal of Industrial Ecology*, vol. 18, no. 1, pp. 113–124, 2014.

[7] J. F. Peters, M. Baumann, B. Zimmermann, J. Braun, and M. Weil, "The environmental impact of Li-Ion batteries and the role of key parameters – A review," *Renewable and Sustainable Energy Reviews*, vol. 67, pp. 491–506, 2017.

[8] C. Thies, K. Kieckhäfer, T. S. Spengler, and M. S. Sodhi, "Assessment of social sustainability hotspots in the supply chain of lithium-ion batteries," *Procedia CIRP*, vol. 80, pp. 292–297, 2019.

[9] European Commission and Directorate-General for Communication, *Circular economy action plan: for a cleaner and more competitive Europe*. Publications Office, 2020.

[10] A. B. Lopez, K. Vatanparvar, A. P. Deb Nath, S. Yang, S. Bhunia, and M. A. Al Faruque, "A Security Perspective on Battery Systems of the Internet of Things," *Journal of Hardware and Systems Security*, vol. 1, no. 2, pp. 188–199, 2017.

[11] European Commission, "Proposal for a REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL concerning batteries and waste batteries, repealing Directive 2006/66/EC and amending Regulation (EU) No 2019/1020," 17.03.2022. [Online]. Available: http://data.consilium.europa.eu/doc/document/ST-7317-2022-INIT/X/pdf [Accessed: 01.06.2023]

[12] Council of the EU, "Council and Parliament strike provisional deal to create a sustainable life cycle for batteries," Press release, 2023. [Online]. Available: https://www.consilium.europa.eu/en/press/press-releases/2022/12/09/council-and-parliament-strike-provisional-deal-to-create-a-sustainable-life-cycle-for-batteries/ [Accessed: 01.06.2023]

[13] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proceedings of the 44th annual Design Automation Conference*, ser. ACM Conferences, S. P. Levitan, Ed. New York, NY: ACM, 2007, p. 9.

[14] T. McGrath, I. E. Bagci, Z. M. Wang, U. Roedig, and R. J. Young, "A PUF taxonomy," *Applied Physics Reviews*, vol. 6, no. 1, p. 011303, 2019.

[15] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical Unclonable Functions and Applications: A Tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.

[16] C. Mesaritakis, M. Akriotou, A. Kapsalis, E. Grivas, C. Chaintoutis, T. Nikas, and D. Syvridis, "Physical Unclonable Function based on a Multi-Mode Optical Waveguide," *Scientific reports*, vol. 8, no. 1, p. 9653, 2018.

[17] M. Hiller, "Key Derivation with Physical Unclonable Functions," Dissertation, Technische Hochschule München, München, 2016. [Online]. Available: https://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20161219-1311665-1-7 [Accessed: 01.06.2023]

[18] A. Scholz, L. Zimmermann, A. Sikora, M. B. Tahoori, and J. Aghassi-Hagmann, "Embedded Analog Physical Unclonable Function System to Extract Reliable and Unique Security Keys," *Applied Sciences*, vol. 10, no. 3, p. 759, 2020.

[19] Google, "Certificate Transparency: How CT works," 2022. [Online]. Available: https://certificate.transparency.dev/howctworks/ [Accessed: 01.06.2023]

[20] B. Laurie, A. Langley, and E. Kasper, "Certificate Transparency," 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc6962 [Accessed: 01.06.2023]

[21] R. C. Merkle, "A Digital Signature Based on a Conventional Encryption Function," in *Advances in Cryptology — CRYPTO '87*, C. Pomerance, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1988, pp. 369–378.

[22] K. Huang, Y. Liu, N. Korolija, J. M. Carulli, and Y. Makris, "Recycled IC Detection Based on Statistical Methods," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 947–960, 2015.

[23] U. Guin, K. Huang, D. Dimase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1207–1228, 2014.

[24] F. B. de Oliveira, A. Nordelöf, B. A. Sandén, A. Widerberg, and A.-M. Tillman, "Exploring automotive supplier data in life cycle assessment – Precision versus workload," *Transportation Research Part D: Transport and Environment*, vol. 105, p. 103247, 2022.

[25] M. Honic, I. Kovacic, P. Aschenbrenner, and A. Ragossnig, "Material Passports for the end-of-life stage of buildings: Challenges and potentials," *Journal of Cleaner Production*, vol. 319, p. 128702, 2021.

[26] T. Adisorn, L. Tholen, and T. Götz, "Towards a Digital Product Passport Fit for Contributing to a Circular Economy," *Energies*, vol. 14, no. 8, p. 2289, 2021.

[27] K. Vittilapuram Subramanian and A. Madhukar Lele, "A SYSTEM AND METHOD FOR GENERATION AND VALIDATION OF PUF IDENTIFIER OF A BATTERY PACK," Patent WO2 022 023 280A2, 2022.

[28] I. Zografopoulos and C. Konstantinou, "DERauth: A Battery-Based Authentication Scheme for Distributed Energy Resources," in *2020 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2020, pp. 560–567.

[29] M. Dubarry, C. Pastor-Fernández, G. Baure, T. F. Yu, W. D. Widanage, and J. Marco, "Battery energy storage system modeling: Investigation of intrinsic cell-to-cell variations," *Journal of Energy Storage*, vol. 23, pp. 19–28, 2019.

[30] M. Baumann, L. Wildfeuer, S. Rohr, and M. Lienkamp, "Parameter variations within Li-Ion battery packs – Theoretical investigations and experimental quantification," *Journal of Energy Storage*, vol. 18, pp. 295–307, 2018.

[31] K. Rumpf, M. Naumann, and A. Jossen, "Experimental investigation of parametric cell-to-cell variation and correlation based on 1100 commercial lithium-ion cells," *Journal of Energy Storage*, vol. 14, pp. 224–243, 2017.

[32] D. Beck, P. Dechent, M. Junker, D. U. Sauer, and M. Dubarry, "Inhomogeneities and Cell-to-Cell Variations in Lithium-Ion Batteries, a Review," *Energies*, vol. 14, no. 11, p. 3276, 2021.

[33] M. Kouhizadeh, J. Sarkis, and Q. Zhu, "At the Nexus of Blockchain Technology, the Circular Economy, and Product Deletion," *Applied Sciences*, vol. 9, no. 8, p. 1712, 2019.

[34] T. K. Agrawal, V. Kumar, R. Pal, L. Wang, and Y. Chen, "Blockchain-based framework for supply chain traceability: A case example of textile and clothing industry," *Computers & Industrial Engineering*, vol. 154, p. 107130, 2021.

[35] F. Casino, T. K. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues," *Telematics and Informatics*, vol. 36, pp. 55–81, 2019.

[36] S. P. Mohanty, V. P. Yanambaka, E. Kougianos, and D. Puthal, "PUFchain: A Hardware-Assisted Blockchain for Sustainable Simultaneous Device and Data Security in the Internet of Everything (IoE)," *IEEE Consumer Electronics Magazine*, vol. 9, no. 2, pp. 8–16, 2020.

[37] R. Asif, K. Ghanem, and J. Irvine, "Proof-of-PUF Enabled Blockchain: Concurrent Data and Device Security for Internet-of-Energy," *Sensors (Basel, Switzerland)*, vol. 21, no. 1, 2020.

[38] P. Cui, J. Dixon, U. Guin, and D. Dimase, "A Blockchain-Based Framework for Supply Chain Provenance," *IEEE Access*, vol. 7, pp. 157 113–157 125, 2019.

[39] Global Battery Alliance, "Battery Passport Pilot," 2023. [Online]. Available: https://www.globalbattery.org/action-platforms-menu/pilot-test/ [Accessed: 01.06.2023]

[40] Battery Pass, "Advancing the implementation of the battery passport in Europe and beyond: Towards a truly sustainable and circular battery life through digital value chains," 2023. [Online]. Available: https://thebatterypass.eu/ [Accessed: 01.06.2023]

[41] Battery Pass consortium, "Battery Passport Content Guidance." [Online]. Available: https://thebatterypass.eu/assets/images/content-guidance/pdf/2023_Battery_Passport_Content_Guidance.pdf [Accessed: 01.06.2023]

[42] Y. Xie, Y. Guo, S. Yang, J. Zhou, and X. Chen, "Security-Related Hardware Cost Optimization for CAN FD-Based Automotive Cyber-Physical Systems," *Sensors (Basel, Switzerland)*, vol. 21, no. 20, 2021.

[43] R. Prosser, G. Offer, and Y. Patel, "Lithium-Ion Diagnostics: The First Quantitative In-Operando Technique for Diagnosing Lithium Ion Battery Degradation Modes under Load with Realistic Thermal Boundary Conditions," *Journal of The Electrochemical Society*, vol. 168, no. 3, p. 030532, 2021.

[44] MARBEL Project, 2023. [Online]. Available: https://marbel-project.eu/ [Accessed: 01.06.2023]

A Survey on Secure Android Apps Development Life-Cycle: Vulnerabilities and Tools

Mohammed El Amin TEBIB
Mariem GRAA
Oum-El-Kheir AKTOUF
Univ. Grenoble Alpes, Grenoble INP, LCIS lab., 26000 Valence,France*
**Institute of Engineering Univ. Grenoble Alpes*
mohammed-el-amin.tebib@univ-grenoble-alpes.fr
mariem.graa@univ-grenoble-alpes.fr
oum-el-kheir.aktouf@univ-grenoble-alpes.fr

Pascal ANDRE

LS2N Lab
UMR 6004 CNRS, Nantes University,
F-44000 Nantes, France

pascal.andre@ls2n.fr

***Abstract*—Mobile devices are increasingly used in our daily lives. To fulfill the needs of smartphone users, the development of mobile applications has been growing at a high rate. As developers are not necessarily aware of security concerns, most of these applications do not address security aspects appropriately and usually contain vulnerabilities. Therefore, it is essential to incorporate security into the app development life-cycle. To help development teams to address security issues, several security integrated development environment (IDE) plugins have been proposed. In this paper, we aim to review the effectiveness of existing IDE plugins in detecting known Android vulnerabilities. We developed a classification framework that highlights the salient features related to 16 selected IDE plugins including: (1) the analysis-based approach, (2) the vulnerabilities checks coverage, and (3) the development stage, on which these tools could be employed. We proceeded to a deep analysis process where each tool effectiveness is investigated against 19 vulnerabilities. Each vulnerability has been demonstrated by executing a corresponding attack scenario on the recent version 12 of Android. The study results provide an overview of the current state of secure Android application development and highlight limitations and weaknesses. Limits such as: tools unavailability, benchmarks incompleteness, and the need of dynamic analysis approaches adoption are among the main findings of this study. The paper synthesizes valuable information for future research on IDE plugins for detecting Android-related vulnerabilities.**

Keywords- Android; Software development; DevSec; Secure Coding; Classification Framework; Security IDE Plugins.

I. INTRODUCTION

With the increase of mobile devices usage, a growing number of applications have been developed to satisfy Android users' requirements. In October 2022, approximately 97000 mobile apps were released through the Google Play Store[1]. However, most of these applications are developed without integrating proper security needs. Due to the lack of security awareness among developers, many of these applications contain vulnerabilities. According to the official MITRE data-source for Android vulnerabilities, *Common Vulnerabilities and Exposures (CVE) details*,[2] recent years witnessed the most significant increase of Android security threats, *"1034 vulnerabilities the last couple years"* , and it continues to increase with *"34 vulnerabilities only the two first months of 2022"*. These vulnerabilities could be exploited to create harmful actions such as developing malwares and stealing users private information. Thus, many updates and patches are provided to the published applications. Therefore, there is an urgent need to fix source code vulnerabilities at the design and development stages and to integrate security-by-design concepts and practices. More precisely, security verification must belong to Integrated Development Environments (**IDE**), as *plugins*, rather than being just external tools, otherwise developers would often consider security as a secondary concern. Security verification feedback should appear as syntax or type errors in the IDE to be part of the developer's activity. Industry and academia tools started recently to integrate security to the software development life-cycle, shifting *from* just ensuring the development speed and leaving the security checks to external stakeholders, *to* employing new software development paradigms such as *development, security, and operations (DevSecOps)* [2]. In these paradigms, developers adhere to a secure development process by means of training sessions and analysis tools.

To assist non-expert Android developers in addressing security concerns, it is essential to provide them with an up-to-date overview of IDE tools that can help to secure their applications. This is the main contribution of our article. We selected a sample of IDE plugins (tools). This sample includes the four most well-known industrial plugins, as extracted from the OWASP list (Open Source Application Security Project) [3]. Additionally, we included academic tools for more comprehensive analysis.

To study these tools, we propose a classification framework based on three dimensions: (1) the analysis-based approach (static or dynamic); (2) the covered security vulnerabilities by each tool; and (3) the development stage, on which these tools could be employed. To limit the scope of our study, we consider the following constraints: (a) Only tools usable during the development life-cycle are considered; i.e., *the tools integrated in the IDE environment*. (b) For academic tools, if the tool code is not provided, our analysis is based only on reading the corresponding published scientific publications and related documentations. (c) For industrial tools, only free available ones are considered.

This classification framework allows to highlight salient features of 16 selected tools by a shallow analysis. We proceeded to a deep analysis process by running the tools on a relevant security benchmark. Each tool effectiveness is investigated against 19 vulnerabilities. Each vulnerability has been demonstrated by executing a corresponding attack scenario on the recent version 12 of Android. The study results allowed to establish a picture of the current secure Android application development.

This article is an extended version of a paper published in Secureware 2022 [1]. The tool information has been revisited to add more details. New material includes a detailed presentation of the

[1]https://fr.statista.com/

[2]https://www.cvedetails.com/product/19997/Google-Android.html

framework in Section V, including the list of potential vulnerabilities; a review of the security analysis approaches and a position of the topic in the Secure Software Development Life-Cycle. The main contribution of our work is an empirical evaluation of the IDE plugins (16 IDE plugins) in detecting vulnerabilities (19 vulnerabilities) for Android 12. We add new research questions in Section IV and new experiments with methodological feedback.

The article is organised as follows. Section II introduces material to understand the context and the comparison methodology. Section III summarizes the existing related works in the literature, reviewing the IDE plugins used for securing Android applications development. We describe our work methodology in Section IV. The classification framework is exposed in Section V. In Section VI, we overview the 16 tools selected according to the selection criteria provided in Section IV. Based on the proposed classification framework, we present in Section VII the main results of the search and analysis phases of our study methodology. We discuss the main findings of our study in Section VIII as well as methodological limitations. Finally, Section IX concludes the paper and provides tracks for future work.

II. Background

This section illustrates the main concepts related to Android applications. Android provides a layered software stack composed of native libraries and a framework as an environment for running Android applications. The framework exposes a set of system services in the form of Applications Programming Interfaces (APIs). An application uses a specific service by instantiating the interface of the corresponding API; the framework launches a remote procedure call to invoke the real implementation that resides on the kernel level.

Based on this software stack, developers implement different types of Android applications: (1) **native** applications that restrict their access to the APIs provided by the framework and (2) **hybrid** applications that could also be web applications. Since considering the security of hybrid applications should cover a wide range of potential security issues related to the web, our study only covers native Android applications (also called **apps** in this article).

A native Android application is built using four types of components, categorised into two families: (1) `Foreground components` such as *activities*, and (2) `Background components` such as *services*, *broadcast receivers*, and *content providers*. Activities provide graphical interfaces, which allow the user to interact with the app to perform different tasks (following the Model-View-Controller (MVC) pattern). The other components are merely used to handle background processing and communications. The core functionalities of the application are implemented through *services* that are used for long-running operations. `Content providers` manage the data layer (storage, read/write accesses), they are used to share data between apps; and finally *broadcast receivers* that receive and respond to `Broadcasts` (i.e., messages that the Android system and Android apps send when events that might affect the functionality of other apps occur). There are two types of broadcasts: system broadcasts, that are delivered by the system, and custom broadcasts, that are delivered by apps. `Custom Intent` actions are defined to create a custom broadcast. There are four ways to deliver a custom broadcast: normal, ordered, local, sticky. Normal broadcasts are sent to all the registered receivers at the same time, in an undefined order. Ordered broadcasts are sent in defined order, the `android: priority` attribute determines the order of the broadcast sending. If more than one receiver with same priority are present, the sending order is random. The intent is propagated from one receiver to the next one. A receiver has the ability to update the intent or cancel the broadcast. Local broadcasts are specifically sent to receivers within the same app. On the other hand, sticky broadcasts are a unique type of broadcast where the intent object that is sent remains in the cache even after the broadcast has been completed. This allows other components within the app to access and retrieve the intent at a later time. The system may broadcast once again sticky intents to later registrations of receivers. Unfortunately, the sticky broadcasts API suffers from numerous security-related shortcomings. Sticky broadcasts can lead to sensitive data exposure, data tampering, unauthorized access to execute behavior in another app, and denial of service. The *intents* manage the communication aspects between the application components. Communications could be implicit in case the message target is not specified. Application can pass a `PendingIntent` to another application in order to allow it to execute predefined actions. The unfilled fields of a pending intent can be modified by a malicious app and allow access to otherwise non-exported components of a vulnerable application.

To ensure secure execution of Android applications, two main security artefacts are considered: *sand-boxing* and *permissions*. An application runs within its own context or sandbox, which is an isolation mechanism between Android applications. So applications cannot interact with other installed applications or the Android operating system (OS) without proper permissions. Thus, the permission system restricts the access to applications, application components and system resources (contacts, locations, images, etc.) to those having the *required permissions*. Android categorizes permissions into different types, including install-time permissions and runtime permissions. The install-time permissions allow an app to perform restricted actions that minimally affect the system or other apps. Thus, they are automatically granted when the app is installed. There are several sub-types of install-time permissions, such as normal permissions and signature permissions. Normal permissions allow access to data that present very little risk to the user's privacy. The system assigns the normal protection level to normal permissions. Signature permission are granted by system to an app only when the app is signed by the same certificate as the app or the OS defining the permission. The system assigns the signature protection level to normal permissions. Runtime permissions give the app access to restricted data such as private user data or allow the app to perform restricted actions that more substantially affect the system and other apps. Thus, runtime permissions are not automatically granted, since their access could be given or denied by the user. The system assigns the dangerous protection level to runtime permissions.

Permissions are declared by developers in the manifest file. Many studies showed that the manifest file can be the source of many security issues [4]: privilege escalation resulting from the over declaration of permissions [5], communication issues resulting from the use of undocumented message types of intents [6], etc.

III. Related works

In this section, we resume the related works aiming to review or evaluate the tools assisting developers in securing Android applications and we show how the current survey extends these works.

Significant effort has been made by the research community to assess security analysis tools for software development in general, and mobile applications more specifically. Recent works [7] and [8] present general reviews of existing IDE plugins for detecting security vulnerabilities in software applications. In [8], the authors selected plugins for the 5 main IDEs (Eclipse, Visual Studio, IntelliJ, NetBeans, Android Studio) and specifically focused on 17 plugins that provide support for input-validation-related vulnerabilities (as described in the Common Weakness Enumeration (CWE) repository). By reading documentations, they listed salient related features such as their supported IDEs, applicable languages and their capabilities in detecting security vulnerabilities. No experimentation is done to assert the documentation sources. In [7], only five open-source plugins were selected. These plugins were also studied in [8], except FindSecBugs/SpotBugs (selected as it is more recent than Findbugs). The authors evaluated coverage and performance by experimenting 14 CWE entries for the 10 top Open Worldwide Application Security

Project (OWASP) categories, using the Juliet Test Suite[3] that is a collection of intentionally vulnerable artificial code written in C, C++, C#, Java or PHP. They also evaluated usability by analysing the result quality and the plugin suggestions. Both works are complementary and present interest but none of them focuses on the Android system, its usual programming languages (Java & Kotline) and its vulnerabilities. In our work, we only focus on Android apps, and we cover even a larger scope by handling a more complete and up-to-date set of existing IDE plugins, which still miss evaluation. Consequently, we will provide a more complete and consistent analysis by covering a wider set of vulnerabilities and finer applicability assets.

In [9], Mejía et al. conducted a systematic review to establish the state of the art of secure mobile development. They found 7 assisting solutions for secure development. These solutions are classified based on: 1) the type of the use (methodologies, models, standards or strategies); and 2) the related security concern (authentication, authorisation, data storage, data access and data transfer). After analysing the results of this research, we consider that the number of handled solutions is limited regarding the real existing ones in the literature. In addition, we found that none of the presented solutions is proposed as a tool or a plugin for secure development. In our work, the search and the analysis processes are deeper. Indeed, we present a higher number of assistant solutions, which are intended to be used as IDE plugins. In addition, our classification is related to software development life-cycle of Android applications.

Janaka S. et al. [10] presented a Systematic Literature Review (SLR) on the Android vulnerability detection tools based on machine learning techniques. 118 technical studies were carefully studied. The results showed that machine learning techniques are used to increase the security of an Android application based on three steps: 1) considering analysing, 2) detecting, and 3) preventing vulnerabilities. In the same line, another SLR was proposed in [11] that classifies 300 security analysis tools based on 1) the analysis dimension (malware detection, vulnerability detection) and 2) the type of security threat (spoofing, Denial of Service, Privilege Security, etc). Regarding these works, the studies are too generic and there is no evaluation process included in the study.

The closest works to our research are [12] and [13]. In [12], Bradley et al. proposed an assessment and evaluation study of Android applications analysis tools. The tools were categorized based on the security issue they solve. To evaluate these tools, the authors recruited eight computer science students with confirmed Java programming skills for a period of 10 weeks. The students evaluated each tool against open source applications extracted from known benchmarks, such as DroidBench and GooglePlay. The results of each evaluation showed the time needed to configure the requisite environment, the problem to address, the vulnerabilities to detect and the type of analysis to perform. Many limitations related to usability, time of execution and analysis precision were outlined as results of the study. The assessment study proposed by Mitra et al. in [13] evaluated the effectiveness of vulnerability detection tools for Android applications. The authors considered 64 security analysis tools and empirically evaluated 19 of them against the *Ghera* benchmarks [14]. It captures 42 known vulnerabilities, each implemented inside a single Android application. As a result, they found that the evaluated tools for Android applications are very limited in their ability to detect known vulnerabilities. The sample of tools in these studies are oriented for use by pen-testers after the application release. All of them are not IDE plugins (except FixDroid) In addition, the evaluation process is limited to the academic tools. In our work, we are interested in academic and industrial free tools, which are specifically intended as security assisting tools. We did not find existing research work that studies precisely Android IDE plugins from a security perspective. After analysing the existing benchmarks, we consider that *Ghera* repository [14] is the most useful means for evaluating the analysis tools. Indeed, *Ghera* summarises a non-exhaustive list of well know vulnerabilities related to the development of Android applications. It provides an open source Android application implementing each vulnerability. Moreover, having numerous test cases with single vulnerabilities is better, in terms of evolution, than big test cases covering several vulnerabilities. Indeed new Android releases remove old vulnerabilities. Therefore, to conduct our study, we used the same benchmark as Mitra et al. [13] to evaluate the list of our selected plugins.

In summary, existing comparisons are interesting at first sight but have limited scope or panel of tools, some are deprecated because security evolves with the OS releases. In the next section, we propose a new methodology to cover a large range of development and security fields and a consistent panel of available tools.

[3]https://samate.nist.gov/SRD/testsuite.php

IV. Research Methodology

This section illustrates the proposed methodology for conducting a precise analysis and comparison study of existing tools dealing with security concerns throughout the Android app development process.

A. Research questions

The study aims to answer the following research questions:

- `RQ1` *Which tools are being employed in the development of secure Android applications?* The goal is to conduct a review of significant related works and identify tools to aid in secure code development.
- `RQ2` *Is security considered in all the design activities during the development process of Android apps?* This research question explores the coverage of security analysis solutions throughout the entire software development life-cycle (Section V-A).
- `RQ3` *Which analysis techniques are being adopted by the existing security development solutions?* This research question aims to map tools to the existing analysis techniques described in Section V-C.
- `RQ4` *Are the studied IDE plugins effective in detecting known vulnerabilities?* Through this research question, we aim to investigate the capabilities of the IDE plugins in detecting the list of Android vulnerabilities presented in Section V-B.

B. Classification framework

A classification framework enables to structure our comparison study by grouping the search space on axes. Four main dimensions are explored:

- **what** to find as security issues;
- **where** to assist developers in detecting security vulnerabilities;
- **when** to handle the security issues during the software development;
- **how** to proceed to detect security vulnerabilities.

This framework will be detailed in Section V. It serves as a structuring basis for our analysis approach. Next, we present the followed search methodology to identify relevant security assistance tools. These tools will be examined to refine all the dimensions of the presented framework.

C. Research methodology process

The proposed research methodology is depicted in Fig. 1. Three main phases are considered in the process:

1) **Tools search & selection**: This phase highlights the tools used by designers and/or developers to prevent security issues in Android applications. To define this list, our primary source of information were mainly:
   - For academic tools, we focused on published academic reviews [7] [8], systematic mapping studies [9] [15], and public GitHub repositories [16] [17];
   - For industrial plugins, we considered only free and available ones such as SonarLint [18], Findbugs [19],

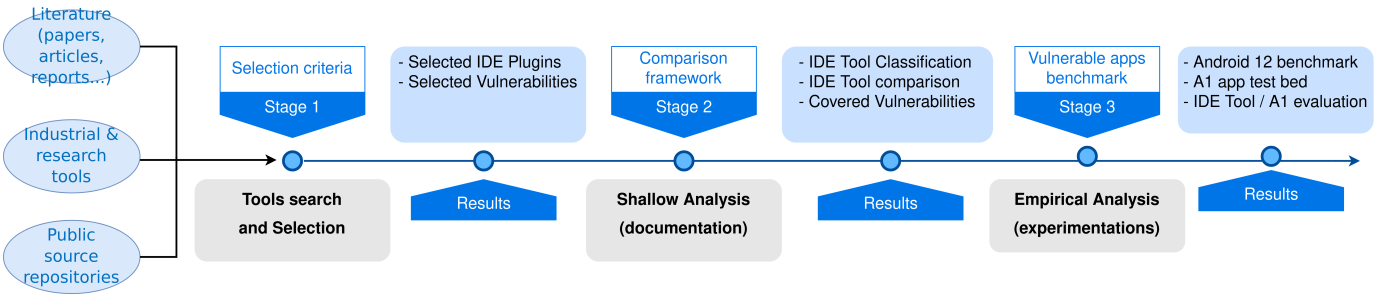


Fig. 1. Proposed Research Methodology

AndroidLint [20], and Snyck [21]. We had the opportunity to experiment them during the development of many IT industrial projects in France. They are also listed among the well known tools extracted from the OWASP list [3]. We also included solutions built on top of AndroidLint [20] such as: Lintent [22], and icc−lint [23].

- As an additional selection criteria, we included the tools that we had the opportunity to investigate while we are building the PermDroid tool [24] in a complementary research work. This tool is based on formal analysis to prevent permissions related security issues in Android applications.

To continue the selection process, we also considered the following excluding criteria:

- Tools that are not oriented for detecting vulnerabilities during the development process like Anadroid [25], which is used for malware detection, and MassVet [26], which is used for analysing packaged applications in Google-Play store;
- Tools that cannot be used within the IDE, e.g., ComDroid [27], which warns pen-testers of exploitable inter applications communication errors related to the released applications. The investigation of this kind of tools has already been done by Ul Haq et al. in [15], and J. Mitra in [13];
- Tools that are integrated in the IDE but are not concerned by security vulnerabilities, like PMD [28], and CheckStyle [29]. These tools are used for checking coding standards, class design problems, but cannot be used for identifying code smells related to security issues;
- Industrial tools such as: Fortify [30] and Checkmarx [31], which are well-known tools, but we were unable to install them due to their pay access policy.

2) **Shallow analysis**: In this phase, we conduct an evaluation of the IDE plugins security coverage against the list of vulnerabilities presented in Section V-B. This analysis is *shallow* because it is only performed through investigating the available documentation and/or the published papers. Three teams are constituted: Team1 is the first author. Team 2 is a group of three final year students (Rémi AGUILAR, Tristan Boura and Nicolas MEGE) of a cyber-security curriculum in Grenoble Alpes Univ., France. Team 3 is composed of all the authors. The dig of the documentations is performed by different teams in three iterations. Team 2 conducted the initial investigation of all vulnerabilities. Then, Team 1 performed a second iteration. Finally, the results were reviewed by Team 3.

3) **Deep analysis**: In this phase, we perform an experimental analysis that completes the shallow analysis. It consists of performing an empirical evaluation of the selected tools against a subset of vulnerabilities. The evaluation process is realized by Team 2 led by Team 1 and reviewed by Team 3. It was organized according to agile practices. The backlog describes the tasks focusing on the list of vulnerabilities to check on the plugins:

- To DO branch: contains the list of vulnerabilities to be investigated during the week;
- Doing branch: contains the list of vulnerabilities under investigation, this helped to perform a quick feedback between the team members;
- Review: contains the list of investigated vulnerabilities during the week;
- Done: contains the work verified and validated.

To accomplish each task among those presented in the backlog, we follow an ordered list of steps described in Fig. 2 and below:

a) We install the various tools and plugins on the Intellij IDE, the goal is to check whether a tool detects a given

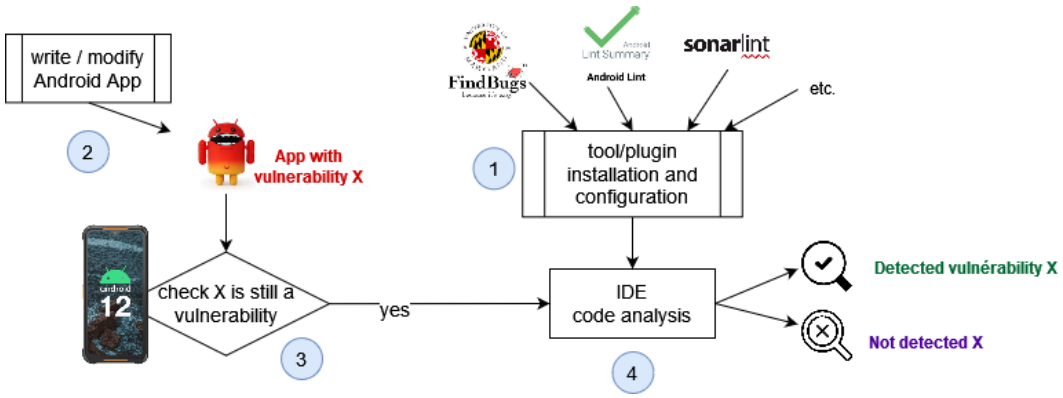


Fig. 2. Deep Analysis Process (experimentation part)

vulnerability in the entry application.

b) We write or modify Android applications in order to have vulnerable applications. The vulnerabilities are those we selected from the Ghera repository [14].

c) We check that Android 12 does not prevent the vulnerability (the issue has not been closed yet). It is useless to require from a tool to detect "old" vulnerabilities.

d) Once we ensure that the vulnerability could be exploited on the recent Android version, we analyse the vulnerable application by the selected IDE plugins and we report the results.

This evaluation has been conducted only with available and free tools such as SonarLint [18], FindBugs [19], Android–Lint [20], lint –icc [23], and FixDroid [32]. We attempted to experiment more tools but it was not possible due to the unavailability of the tools. We contacted the authors of PerHelper [33], 9Fix [34] and Vandroid [35] but we did not receive an answer yet. Consequently, we decided to perform a second iteration on the documentation analysis for the unavailable tools instead of experimenting them (which was not possible). Finally, as our study focuses on vulnerabilities that could be found at the code level (*cf.* Section V-B), our deep analysis could not apply to tools such as Sema [36], PoliDroid–As [37], Page [38] because the inputs of these tools are respectively: GUI Storyboards for Sema [36], Textual specification of the application for PoliDroid–As [37] and Page [38], and not the application source code.

In the following, we present our framework in Section V, the selected tools in Section VI and the analysis and evaluation results in Section VII.

V. Classification Framework

The classification framework is a comprehensive analysis of the relevant criteria of security concerns in the development cycle of Android applications. Our classification framework contains four main dimensions as illustrated in Fig. 3.

- **IDE plugins**: As a first dimension the goal is to define the IDE plugins Where the presented security vulnerabilities will be checked. We conducted a review of significant related works and identified tools to aid in secure code development. We thoroughly discussed each tool, its capabilities, and the mitigation process during development. Finally, we determined the vulnerabilities covered by each tool.
- **Design level**: This dimension explores When the selected tools could be employed regarding the entire software development life cycle. The study in this dimension is conducted based on the engineering phases presented in Section V-A.
- **Analysis approaches**: In this dimension, we manually inspected How the selected tools behave to analyse security vulnerabilities. This enabled us to figure out the adopted analysis approaches of IDE security plugins regarding the analysis methods presented in Section V-C such as Fuzzing, Instrumentation, Symbolic Execution, and Formal verification.
- **Security vulnerabilities**: In the final dimension, we examine the selected tools effectiveness against a non exhaustive list of vulnerabilities that we selected from the Ghera repository [14] (*cf.* Section V-B). The goal is to investigate What are the well known vulnerabilities covered by the selected tools. In addition, we tested each vulnerability based on corresponding attack scenarios on recent versions of the Android OS to confirm the associated risks.

Next, we present each dimension in detail.

A. Secure Development Life-cycle (when)

Software developers are pointed out in many exploratory studies [40] [41] as the main reason of security vulnerabilities. This is because they often consider security as an afterthought concern. Software developers are mostly not security experts. Considering security requirements would need many interactions with software security experts, hence adding delays to the core software development. Consequently, these interactions are not considered in the development life cycle like many non-functional requirements, while functional testing starts in the requirements analysis phase by providing user scenarios that will be the source of acceptance testing. To overcome these limitations, software organizations have recently initiated new software development paradigms allowing to secure the SDLC [42], [43] (see Fig. 4 borrowed from [39]). The goal is to ensure security requirements throughout the entire software development pipeline: requirements analysis, design, coding, testing deployment/runtime and decommissioning. With this regards, software development and operations (DevOps) approaches are of high interest as one of their objectives is to improve communication and interaction between involved actors in the software development process.

- Specification: considering security at software specification level is recommended at the head of the 10 proactive controls

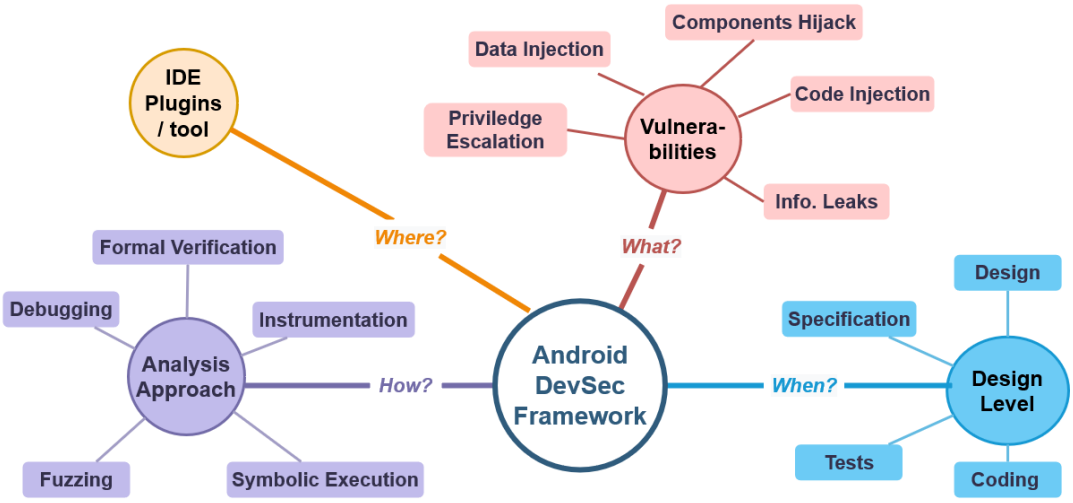


Fig. 3. Android DevSec Classification Framework

list of OWASP[4]. The goal is to ensure that the software design meets security requirements and minimizes the risk of security vulnerabilities. Potential vulnerabilities analysis, security requirement and risk assessment become part of this phase. Security requirements define or redefine features to solve specific security problems or eliminate potential vulnerabilities.

- Design: At this level, security may be ensured based on the following steps: (1) to conduct threat modeling by designing potential security threats and vulnerabilities; and (2) to select well known design patterns for fundamental aspects such as: access control, encryption, etc.
- Implementation: Considering security during the coding phase can help to identify and fix security vulnerabilities early in the development process, before they become serious issues that could be exploited by attackers. At this stage, security could be ensured in many ways such as: implementing applications by following secure coding best practices, using static and dynamic code analysis, using secure libraries, and automate security checks.
- Testing: considering security at the testing stage by performing different types of structural and functional tests such as: unit, instrumentation, fuzzing tests. *Pentests* or security tests should also be applied at this stage.
- Deployment & Post-Deployment: at this stage, operational security processes and tools are needed. In particular, they include testing build routines and user acceptance testing.

One of the first approaches that have been proposed to consider security needs in the software development lifecycle is the Security Development Lifecycle (SDL) by Microsoft. SDL defines twelve stages, in which objective is to consider security throughout the whole development process. Among these stages, education and awareness, security best practices, security documentation, etc. have been highlighted. Today, these security specific stages are implemented throughout security practices.

OWASP has strengthened the Software Development Lifecycle with a Software Assurance Maturity Model (SAMM) that encompasses security aspects throughout the development process. In the OWASP SAMM model, security governance is considered as a specific stage and highlights, as in the Microsoft Secure Development Lifecycle, the importance of training and education. Traditionally, IT people were in charge of security testing, once the application has been released. Such delayed security tests are no longer sufficient in the nowadays context, with highly interconnected applications and numerous security breaches.

Thus, *DevSecOps* approaches have been introduced. At the core of DevSecOps is the principle of keeping security as a priority and adding security controls and practices into the DevOps cycle [44]. All these initiatives that aim to handle security aspects put the light on the methodological concerns including models, processes and people, tools and automation.

In the context of Secure Android Apps Development Life-Cycle, models are those of the vulnerabilities, actors are mobile app developers, process, tools and automation are implemented and integrated in IDEs. We will detail these aspects in the next sections.

B. Security vulnerabilities related to the development process (what)

This section introduces the identified vulnerabilities in the recent Android versions and illustrates the attack scenarios we will implement.

1) Identified Vulnerabilities: In this work, we are mainly interested in security vulnerabilities **(Vi)** that could be mistakenly introduced by developers and exploited to craft attacks **(Ai)**. Based on available benchmarks such as *Ghera* [14] that contains open source applications implementing vulnerabilities, we started by considering vulnerabilities (V), which belong to the following class of attacks (A): (i) privilege escalation, (ii) data injection, (iii) code injection, (iv) information leaks and (v) components hijacking. These vulnerabilities are summarised in Fig. 5. Below, we briefly describe vulnerabilities for each class.

1) **A1**. Privilege escalation **(PE)**: this attack occurs when an application with less permissions gains access to the components of a higher privileged application. Situations where such an attack can occur are mainly related to:
   - **A1.V1**. The use of *PendingIntent* with empty *base action*: a *PendingIntent* object is a token that is given to a foreign application to allow it to execute a predefined action. When a *PendingIntent* is sent, the receiver application will execute the corresponding action using the sender permission. If a malicious app receives a *PendingIntent* whose base action is empty, then the malicious app can escalate its own privilege,

[4]https://owasp.org/www-pdf-archive/OWASP_Top_10_Proactive_Controls_V3.pdf

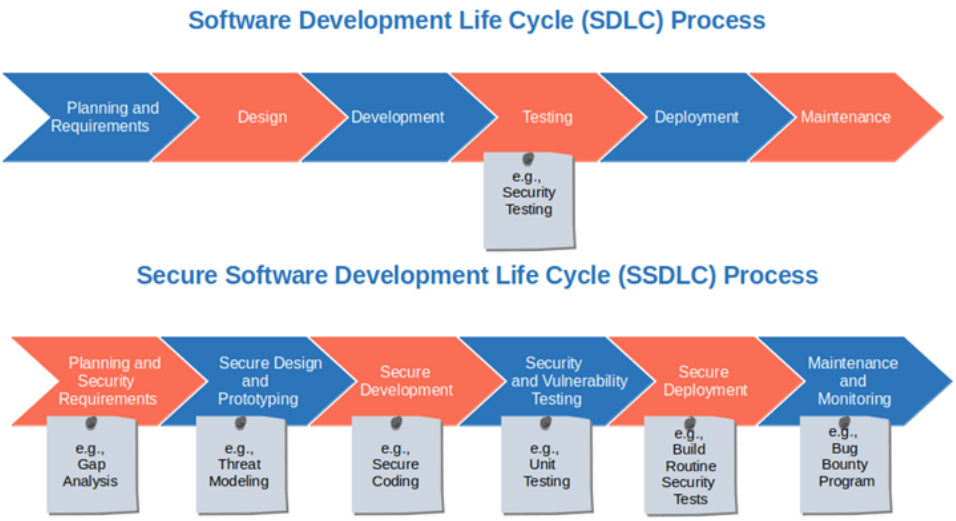


Fig. 4. Secure Software Development Life-cycle (SSDLC) [39]

set an action and execute it in the context of the benign app that sent the *PendingIntent*.

- **A1.V2**. Fragments Dynamic Load: Java reflection is used to dynamically load fragments into an activity, where a fragment is a part of a user interface. But, the fragment name could be provided by a malicious application to make fragment injection attacks. For example, an activity that accepts fragment names as input from other components, and loads the fragment dynamically into the activity is vulnerable to executing a fragment provided as input by a malicious app on the device.
- **A1.V3**. High Privileged component export: If an app has the permission to perform a particular privileged operation and if that operation is performed via an app component that is exported for public consumption, then a malicious app can invoke the component method that performs the privileged operation and make the app perform the operation on behalf of the malicious app. This is a privilege escalation attack or a confused deputy attack.
- **A1.V4**. Permission over-privilege: Android applications privileges are managed through the concept of Permissions. A Permission is declared on the XML manifest configuration file as an XML attribute. Declaring a Permission on the Manifest file means that the application needs this permission to access a system resource(s) such as: CAMERA, GPS, etc. The security design flow here is when developer mistakenly declares permission(s) on the manifest file that are not necessary for the application functioning. Malware could gain these permissions to create harmful actions.
- **A1.V5**. Permission Enforce: If an application uses the method *enforcePermission* to grant permission at run-time, it won't throw a SecurityException when another component in the same process was granted permission earlier. Thus a malicious application could get privilege permission. The exploit of the attack scenario related to this vulnerability does not work on Android 12, 11 and 10 (API 29..33) but works on Android 9 (API 28).
- **A1.V6**. EnfroceCallingOrSelfPermission: When the method enforceCallingOrSelfPermission is used to grant permission, it will not throw further SecurityException from the moment it has already granted one permission to another application, meaning a malicious app could ask a permission after a benign app, and get that permission granted.

2) **A2**. Data injection: Data injection is a type of attack where malicious actors are able to inject malicious data into a software system. It has serious consequences on sensitive data, access control, and quality of service. In Android ecosystem, this can occur through the exploitation of various vulnerabilities, such as:

- **A2.V1**. Ordered broadcasts: As mentioned in Section II, a broadcast is a component that allows an application to send and/or receive messages (intents) to/from the applications and/or the system. Each component could be registered to a broadcast to be notified whenever this broadcast is generated. On its configuration, a broadcast receiver can be declared as either ordered or non-ordered. In non-ordered mode, Android will deliver all broadcasts to all receivers at the same time. On the other hand, ordered broadcast receivers define a priority of transmission. The receiver with higher priority responds first and forwards it to lower priority receivers. Hence, a malicious receiver with high priority can intercept the broadcast, change its content, and forward the malicious payload to the receivers with low priority.
- **A2.V2**. Sticky broadcasts: A sticky broadcast in Android is a broadcast message that is saved by the system and sent to all registered receivers. This type of broadcast can be potentially dangerous because a malicious receiver can modify the message and broadcast it again, causing all receivers to receive the updated message. Sticky broadcast could be sent using the method *sendStickyBroadcast()*, which is deprecated in the latest version of Android. Starting from Android 8.0 (Oreo), the use of sticky broadcasts is discouraged.
- **A2.V3**. *Weak Checks On Dynamic Invocation:* to invoke any provider-defined method. Android does no permission checking on this entry into the content provider. Whenever the developer calls this method without doing its own permission checks, unauthorised components are allowed to interact with the content provider. So, apps that use the call() in the Content Provider API are vulnerable to exposing the underlying data store to unauthorized reads and writes.
- **A2.V4**. *Uncontrolled External Storage Reads* are used to store application data with a public share. If an application reads from any file stored in *External Storage*, even if the file is in the private storage directory, then the application has no control over the file it is reading. If a malicious app changes the file being read by a victim application, then this

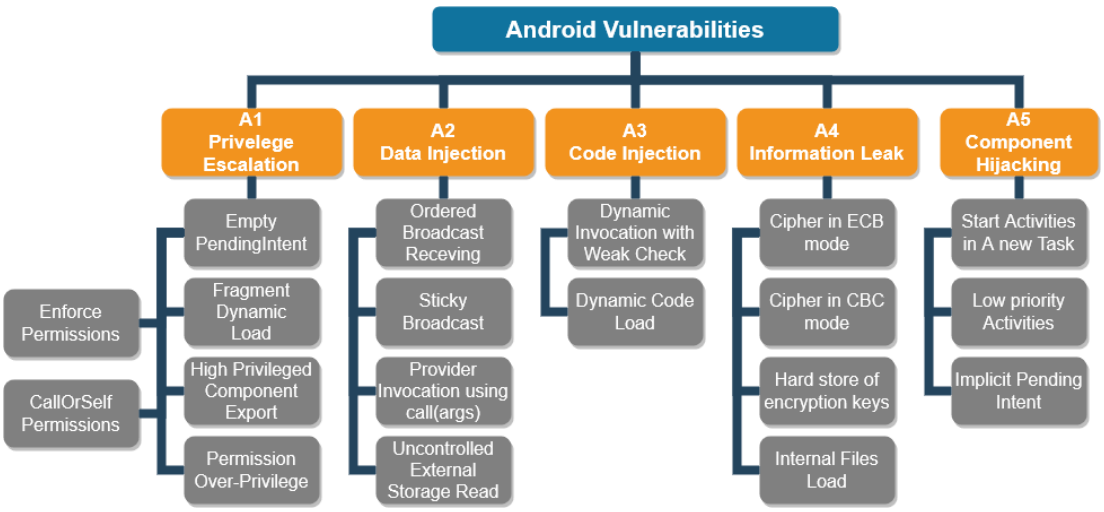


Fig. 5. A taxonomy of Android vulnerabilities

one will unknowingly read malicious content.

3) **A3**. Code injection: consists of injecting potentially malicious code that is after that interpreted/executed by the application. Situations where such an attack can occur are mainly related to:
   - **A3.V1**. Dynamic code loading: Since the apps can load classes from local archives (via PathClassLoader) or remote archives (via URLClassLoader), malicious actors can change such archives and affect the behavior of apps that use the archive. So, Android apps that rely on dynamic code loading without verifying the integrity and authenticity of the loaded code may be vulnerable to code injection.
   - **A3.V2**. Dynamic invocation with weak checks: Android allows developers to share data across apps through the Content Provider API. The Content provider API provides a method call() to call any provider-defined method. But, Android has no idea whether call will read or write data in the provider, so it cannot enforce those individual permissions. Therefore, malicious apps with read and without write permission can write data through the Content Provider.
4) **A4**. Information leak: It occurs when an application private data are accessed by unauthorised applications or when developers improperly use the cipher modes (ECB, CBC) and AEAD (Authenicated Encryption With Additional Data) cipher like GCM (Galois Counter Mode) or by exploiting weaknesses in random number generation (RNG) process. So, an attacker could be able to guess the encrypted message. Situations where such an attack can occur are mainly related to the use of:
   - **A4.V1**. Block Cipher algorithm in Electronic Codebook Block (ECB) mode: ECB mode leaks information about the plaintext because identical plaintext blocks produce identical ciphertext blocks, it does not hide data patterns well. An attacker could be able to guess the encrypted message. So, applications that use a block cipher algorithm in ECB mode for encrypting sensitive information are vulnerable to leaking sensitive data. Thus, ECB is not recommended for use in cryptography protocols.
   - **A4.V2**. A block cipher algorithm in cipher block chaining (CBC) mode: The greatest advantage CBC has over ECB is that, with CBC mode, identical blocks do not have the same cipher. This is because the initialization vector adds a random factor to each block; hence, the same blocks in different positions will have different ciphers. But, applications that use a block cipher algorithm in CBC mode with a non-random initialisation vector for encrypting sensitive information are vulnerable to leaking sensitive data.
   - **A4.V3**. Applications that store the encryption key in the source code are vulnerable to leaking encrypted information. An attacker with decompiler and static analyzer tools can identify the encryption function and find encryption key. Someone who has the assembly code will even be able to call Decrypt directly, not bothering with extracting the key.
   - **A4.V4**. Ability to load files from internal storage (private access) to external storage (public access). Android allows apps to save data in files. These files can be stored in Internal Storage or in External Storage. Files stored in Internal Storage are private to the app. External Storage can be accessed by all apps that have permission to access ExternalStorage.
5) **A5**. Android components hijack: this attack can occur in the following situations:
   - **A5.V1**. Activities that start in a new task: This situation occurs when the user can navigate from a benign activity A to a malicious activity M by pressing the back button, instead of navigating to an intended benign activity C. Therefore, malicious activity M will hijack benign activity C.
   - **A5.V2**. Applications with low priority activities: If a malicious activity has a higher priority than a benign one, then it will appear before the benign activity, making it more likely for the user to choose the malicious activity over the benign one.
   - **A5.V3**. Pending Intent with implicit base intent: If base intent is an implicit intent then, when the pending intent is performed, it can be intercepted by a malicious application. If the implicit intent has sensitive information in it, then it will be leaked.

As we can see, developing Android applications without a prior knowledge and/or a specific focus on security aspects could lead to critical security attacks. It is worth-noting that the above list of vulnerabilities is not exhaustive, and it is intended to be extended in future research.

2) Attack scenario: To clearly outline the exploits utilizing these vulnerabilities, we provided, based on Ghera repository a scenario of attack related to each vulnerability. Each scenario was documented using Gherkin, a language supported by Martin Fowler[5] for outlining functional software test scenarios. In this Section we describe as an example the attack scenario related to vulnerability *A1.V1*. The remaining scenarios can be found in the technical report [45].

Given A simple application called BenignApp
And BenignApp has a module called BenignAppPartner
And BenignAppPartner implements a broadcast receiver
When BenignApp sends a pending intent with empty action to the BenignPartner
Then BenignAppPartner receives this Pending Intent, then launches a remote service that resides on the benign app (until now, it is a normal behavior)
When MalicousApp intercepts through its broadcast receiver the Pending Intent sent to the BenignPartnerApp
Then MaliciousApp manipulates the empty action of this Pending Intent
And MaliciousApp executes the action it wants into the Benign app' service by escalating its own permission

In order to test **A1.V1**, we use three apps: Benign, BenignPartner, and Malicious. Benign is a benign app that sends an empty pending intent to BenignPartner, which has a broadcast receiver MyReceiver.java that takes the pending intent and starts a service MyService.java in Benign app. Malicious app has a broadcast receiver MyReceiver.java that intercepts the pending intent sent from Benign app and starts an internal service in Benign that performs some sensitive operation. This service is not exported and is meant for internal use within the benign app. However, because of the empty pending intent intercepted by the malicious broadcast receiver, Malicious app can escalate its own privilege and can start MySensitiveService eventhough it is not supposed to.

C. Security analysis Approaches and Tooling (how)

Determining the effectiveness of each security analysis tool in detecting known vulnerabilities is closely related to the analysis method used by the tool. Hence, in order to examine the studied IDE plugins effectiveness based on their analysis techniques, we describe in this section common analysis approaches used in software engineering. As summarized in Table I, these approaches are generally classified into 3 groups: Static, Dynamic, and Hybrid analysis.

- **Static analysis approaches**. Static analysis inspects the program without running it. It examines source or compiled binary code against coding flaws. Generally, it is used to identify potential bugs, vulnerabilities, or other security issues. Many automated tools are able to perform static code analysis to report security problems related to Android applications and systems,

[5]https://martinfowler.com/bliki/GivenWhenThen.html

such as: (1) ComDroid [27], which statically analyses DEX code of third party Android applications to detect inter application communication vulnerabilities: Broadcast theft, Activity and Service hijacking, and Broadcast injection; (2) Lintent [46], which is based on formal verification techniques (formal calculus) to reason about the application behaviour and prevent security attack surfaces from privilege escalation and communications; (3) FlowDroid [47], which tracks the flow of data between different components of an Android app. Starting from an Android application, it constructs a full control flow graph. This graph can then be used to identify potential security issues based on data tainting; (4) Arcade [48], NatiDroid [49] and Pscout [50] that perform static analysis on the Android framework to build a permission mapping used for detecting over-privileged applications.

Several techniques could be used to perform static analysis in Android application security analysis, such as formal verification and symbolic execution:

- **Symbolic Execution** can be used to simultaneously explore multiple paths that a program could take under different inputs. This allows sound analyses that can give strong guarantees on the checked property. Rather than taking on fully specified input values, the technique abstractly represents them as symbols, resorting to constraint solvers to construct actual instances that would cause property violations [51]. These symbols will be used along the execution path to determine the path condition. A path condition is a first order logic formula that describes the conditions satisfied by the branches taken along that path. The mapping between variables and symbolic expressions or values will be stored to determine if the property is satisfied at the end of the the execution path. Klee [52] is a symbolic execution tool built on the LLVM compilation framework that automatically generates test cases for high coverage of complex and environmentally intensive programs. *Symbolic Execution* can be conducted on the basis of some pre-extracted models to ensure the reachability of some branches [53]. Many approaches, such as: Scandroid [54], Cassandra [55], etc., analyse data-flow to formally check various system level information-flow properties.
- **Formal Verification**. Formal methods are among the most used techniques in Android security analysis. They cover many application areas such as: security protocols, the implementation of access control policies, intrusion detection and even source or binary code security. Generally, they are categorized in three families as presented in [56]: 1) Specification and Process algebra where the approach that deals with the system behavior is algebraic and axiomatic. 2) Model checking or Property checking, which is a model-based specification technique that aims to develop visual models for the specified system and analyzing their properties. 3) Theorem proving that is an axiomatic technique, in which the system is expressed as a set of axioms and a set of inference rules, and the desired property is expressed as a theorem to be proved. For Android security, this technique has been widely adopted such as in [57] for mobile malware analysis based on model checking. Theorem proving techniques provide a high level of code coverage. However, they suffer from over-approximation results.

Static analysis approaches have some limitations due to undecidability problems [58]. It is impossible to determine if a program will terminate for a given input. Another limitation of static analysis tools is the fact that they report problems that are not really bugs in the code [59] i.e. they identify incorrect behaviors that cannot occur during any run of the program (false positives).

- **Dynamic analysis approaches**. In contrast to static analysis, dynamic analysis is performed at run-time. The goal is to identify problems that cannot be detected by static analysis, such as: race condition, performance, and concurrency problems. In comparison to static analysis, dynamic analysis provides sound results. It does not require an access to the source code, it traces a binary code to understand the system behaviour.

  Many automated tools have been proposed to perform dynamic security analysis on Android applications and system, such as: TaintCheck [60], LIFT [61], Valgrind [62], just to name a few. They instrument the bytecode to control the information flows in the program and detect security attacks. Thus, they suffer from significant performance overhead that does not encourage their use in real-time applications.

  Several techniques could be used to perform dynamic analysis in Android application security analysis, such as fuzzing and instrumentation:

  - **Fuzzing** is an automated technique used for software testing and security. The main idea behind fuzzing is to use randomly generated inputs to fuzz the software under analysis in order to find bugs or vulnerabilities. The set of inputs are called *Oracle*. In software testing, fuzzing could be performed based on different methods, such as: (1) generation-based fuzzing that involves generating inputs from scratch; (2) mutational-based fuzzing (called also feedback-based fuzzing) that modifies the existing inputs during the testing process, in order to redirect the execution paths; and (3) model-based fuzzing that constructs a model for the input data, then generates inputs that conform to that model.

    Fuzzing could be conducted dynamically. It has been used in significant works analysing security vulnerabilities and attacks in Android applications. The results showed that it could cover numerous boundary cases using invalid data as application input to better ensure the absence of exploitable vulnerabilities [63]. Among the existing tools allowing to apply fuzzing for Android security analysis we found: (1) Jazzer, which is a coverage-guided fuzzer for the Java Virtual Machine (JVM). It works on the bytecode level and can be applied directly to compiled Java applications and to targets in other JVM-based languages such as Kotlin or Scala. It is composed by a native binary that links in libFuzzer and runs a Java fuzz target through the Java Native Interface (JNI) and by a Java agent that runs in the same JVM as the fuzz target and applies instrumentation at run–time. These fuzzers can often find out previously unknown vulnerabilities [64]; (2) Dynamo [65] uses mutational based fuzzing to provide inputs that fuzz the Android framework code related to checking the API access control. This process helps to construct a permission-mapping allowing developers to identify over-privileged applications.

    Despite the advantages that fuzzing can offer it suffers from some limitations such as: 1) Low code coverage due to the fact that the executed paths are related the selected inputs; 2) The set of inputs could be computationally intensive; and 3) Inadequate inputs generation, which can result in a high number of false negatives.
  - **Instrumentation.** Generally, fuzzing is combined with instrumentation techniques to perform dynamic analysis. Instrumentation provides the possibility to modify a program under analysis at run-time, and add some functionalities such as: logging messages or new instructions in order to understand the software behaviour.

    There are several toolkits, e.g., Frida [66], ARTIST [67], DaVinci [68] that are used for dynamic instrumentation in Android system. Frida allows developers, reverse-engineers, and security engineers to inject snippets of JavaScript into

TABLE I
SOFTWARE ANALYSIS APPROACHES: STRENGTHS, WEAKNESSES, AND TOOLS

Approach	Technique	Pros	Cons	Tools
Static Analysis	Formal Verification, Symbolic Execution	Efficient, Scalable, High Code coverage	Decidability, Low Soundness	ComDroid, Lintent, FlowDroid, NatiDroid
Dynamic Analysis	Fuzzing, Instrumentation, Debugging	Race Condition, Concurrency, Performance	Time Consuming, Resource-Intensive	Frida, Dynamo, stowaway

native Android apps to monitor and debug running processes. ARTIST is an open source instrumentation framework for Android's apps and Java middleware. It is based on the new ART runtime and the on-device dex2oat compiler to monitor the execution of Java and native code. ARTIST modifies code during on-device compilation. In contrast to existing instrumentation frameworks, it preserves the application's original signature and operates on the instruction level. DaVinci is an Android kernel module for dynamic system call analysis. It provides pre-configured high-level profiles to easily analyze the low level system calls.

The instrumentation adoption faces also some challenges. First, adding some functionalities to a software system will add significant complexity, which may cause bugs and crashes. In addition, there is a high risk of execution interference between the real and the added code. Finally, instrumentation can have a significant impact on performance because the added code can slowdown execution time.

The dynamic analysis limitation is that it consumes resources and has difficulty covering all execution paths, so it generates false negatives.

- **Hybrid analysis approaches.** Hybrid analysis combines and benefits from the advantages of static and dynamic analyses. Regarding the provided tools performing static or dynamic analysis, fewer tools combine both of them due to the difficulty of combination. For instance, AM Detector [69] is an automatic malware detection prototype system based on attack tree model. This approach employs a hybrid static-dynamic analysis method. Static analysis tags attack tree nodes based on application capability. It filters the obviously benign applications and highlights the potential attacks in suspicious ones. Dynamic analysis selects rules corresponding to the capability and conducts detection according to run-time behaviours. In dynamic analysis, events are simulated to trigger behaviours based on application components, and hence it achieves high code coverage.

Overall, static and dynamic analysis are complementary approaches. We presented in this section the commonly used techniques for detecting security issues related to Android applications. Starting from the presented strengths and weaknesses of each technique, we investigate in the coming section the based-analysis approach of the selected IDE plugins in order to study their effectiveness in detecting the selected vulnerabilities.

VI. SELECTED TOOLS (RQ1)

The result of the *Tools search & selection* (phase 1) of the research methodology is a set of 16 IDE plugins that match the selection criteria provided in Section IV-C. This set is an answer to **RQ1**. The list of selected tools is summarised in column 1 of TABLE II. In the reminder of the section, we overview these plugins.

- **Curbing** [5] is the first proposed tool assisting developers in utilizing least privilege principle when developing Android applications. It notifies developers if one or more unnecessary permission(s) is (are) declared unintentionally by the developer. At the time when Curbing was developed in 2011, there was no ready data-set of Android application source code. So, the tool was analysed empirically,
- **SonarLint** [18] is a popular linter that is largely used in industrial IT projects. It contains a large set of code smells for many programming languages, including those for Android security. The tool can also provide suggestions for remediation. SonarLint performs static code analysis (SAST) by whether inspecting the program AST, or using data taint propagation. It also performs dynamic analysis (DAST) even if it is still limited to few dynamic functionalities such as: Memory Error Detection, Invariant Inference, Concurrency Error, which may cause race-condition, resource/memory leak, etc. SonarLint can also be used in many ways: IDE integration (Eclipse, IntelliJ, Android Studio, etc.), or through DevOps pipeline (github, jenkins, etc.),
- **FindBugs** [19] provides static code analysis to look for bugs in Java code from within IntelliJ IDE. For more than 200 bug patterns related to different topics, such as performance, correctness, bad practices, FindBugs also provides the opportunity to investigate the code security, and detect the malicious code vulnerability. As a code review tool, it investigates the AST of the program against predefined rule patterns. The last available version (2016) of FindBugs is not compatible with the final version of IntelliJ. Since 2016, FindBugs is integrated into FindSecBugs [70] tool.
- **Snyk** [21] is one of the leading industrial vulnerability analysis tools. It has a large open-source database of the common vulnerabilities related to different programming languages including Java and Kotlin for Android application development.
- **Android-Lint** [20] (also called Lint) is the official analysis tool provided by Google for Android application development. It checks common issues in an Android project's source code and provides suggestions for improving the code's quality and security. Lint offers a robust API that can be utilized to create custom Android lint checks for additional analysis rules. Several tools based on this API have been found in the literature.
  - **Lintent** [22] is a security analysis plug-in integrated with the Android development tools suite. Based on a static analysis approach, it implements a static formal calculus based analysis to reason on the Android inter-component communication API. Lintent analyzes Java source code through reasoning about types. The goal is to statically prevent privilege escalation attacks on well-typed components. Lintent also detects over-privileged application based on the permission mapping described on the corresponding readme description file of the tool in github [6]. To analyse the program elements, Lintent is built on top of Android–Lint library,
  - **icc-lint** [23] called also AndroidICC. Like Lintent and 9Fix its security checker is based on Lint Checker. It serves to analyse Android application source code and reveals vulnerabilities related to inter-component-communications. These vulnerabilities are extracted from *Ghera* repository. This tool is open-source and ready for use,
  - **9Fix** [34] is another recent plugin supporting secure program-

[6] https://github.com/alvisespano/Lintent

ming. It inspects the vulnerable code and instantly suggests an alternate secure code for developers. The security properties covered by 9Fix are classified as general security code smells related to different topics such as password stealing, the use of vulnerable cryptography algorithms, SSL and TLS communications, just to name a few. 9Fix is integrated as an Android Studio plugin.

- **PerHelper** [33] is an IDE plugin that guides developers to declare permissions automatically and correctly and identify permission-related mistakes, such as over-privilege and under-privilege permissions, unprotected APIs, etc. By inferring the candidate permission sets, PerHelper helps to set permissions more effectively and accurately.
- **VanDroid** [35] is an Analysis tool based on Model Driven Reverse Engineering approach (MDRE). In a first step, Vandroid analyzes the code source of the application based on its Graphical Abstract Syntax Tree (GAST), then generates a corresponding security model that facilitates and improves accuracy of the analysis. As a second step, it launches a formal verification process that identifies security risks related to the Android communication model. The formal verification in Vandroid checks the satisfaction of specified security properties in the generated model. Vandroid is provided as an Eclipse plugin, which limits its adoption on the current Android development projects. We contacted the corresponding authors of Vandroid by email in order to gain access for experimenting the tool. The tool cannot be used in new IDE supporting Android development such as IntelliJ and Android Studio.
- **Sema** [36] is a new methodology for designing secure Android applications from an app's storyboards. Storyboards are used as state-transition diagrams. The states specify the screens of the application, and the transitions present the operations to be launched to transit from one screen (state) to another one. The goal of Sema is to help application designers to care about security at the design stage. While specifying the application functioning and designing storyboards, Sema has the ability to check security properties at this stage based on formal analysis. Finally, Sema is able to generate a secure source code starting from the designed storyboards. The tool is open-source and can be integrated in Idea IDEs such as IntelliJ and Android Studio.
- **PoliDroid-As** [37] is a tool allowing Android developers to check if the created code adheres to privacy security policies. These policies are not created by developers but by legal experts using natural language. The main contribution of PoliDroid−As is to automatically align the code source of the application to the specified security policies. By using natural language processing algorithms, PoliDroid-As maps key phrases existing in the documentation of the used APIs to low-level technical terminology in the API.
- **Page** [38] is a tool that supports developers creating natural language privacy policies during the development process. It is proposed as an Eclipse plugin to document privacy tasks and notes during the construction and testing of the application. There is no automatic alignment between the specified textual policies and the source code of the application. The goal of the tool is just to make an internal IDE repository to remind the developers checking the validity of these policies after each code evolution.
- **PermitMe** [71] has the same goal as Curbing, Perhelper and Lintent. PermitMe is a code review tool (CR) used to notify developers when they intentionally include extra permissions in their apps. By statically investigating the program Abstract Syntax Tree (AST). PermitMe decides whether a declared permission is used in the program or not. If at least one permission is not used, then the application is flagged as over-privileged.
- **Coconut** [72] has been developed on 2018 as an IDE plugin for Android Studio. It helps developers to handle privacy based on the concepts of annotation. Through heuristics (H), Coconut detects the code that handles personal data, and asks the developer to add an annotation that describes how and why the personal data is used. To simplify the handling of annotations, Coconut proposes a quickfix to automatically generate an annotation skeleton with several fields (i.e., 'dataType', 'frequency'). These fields could be filled automatically if they could be inferred from the code, or manually in the opposite case. Filling out the annotation makes the developers think to the use of private data, such as examining the collections manipulating the private data, checking that there is a legitimate reason to collect the private data, etc. Coconut is available and open-source.
- **FixDroid** [73] is an Android Studio plugin (it is not available on IntelliJ). It provides helpful security alerts, explanations and quickfixes whenever possible. It is updated periodically to add new features and fix software bugs.

TABLE II
IDE SECURITY ANALYSIS TOOLS FOR ANDROID APPLICATIONS

Tool Name	Year	SDLC	Focus	Approach	Method	Availability	AV
Curbing	2011	Dev	Permission Over-privilege	Static, Manual	AST	No	2.2
Lintent	2013	Dev	Communication	Static	FM	Yes	4.x
PermitMe	2014	Dev	Permission Over-privilege	Static	AST	No	5.0
Page	2014	Spec	Privacy policies	Static	NLP	No	-
Vandroid	2018	Design, Dev	ICC	Static	FM	No	9.0
AndroidLint	2019	Dev	General Code Smells	Static	AST	Yes	all
Sema	2019	Design	General Security Properties	Static	FM	Yes	10
PerHelper	2019	Dev	Permission Over-privilege	Static	AST	No	10
PoliDroid−As	2017	Spec	Privacy security policies	Static	NLP	No	8
9Fix	2021	Dev	General code smells	Static	AST	No	12
SonarLint	2021	Dev	General code smells	Hybrid	DAST	Yes	12
Find Bugs	2016	Dev	General code smells	Static	AST	Yes	7
Cocunut	2018	Spec	Privacy policies	Static	H	No	-
FixDroid	2017	Dev	General Code smells	Static	AST	Yes	7
icc−lint	2019	Dev	ICC	Static	AST	Yes	10
Snyk	2014	Dev	General code smells	Hybrid	DAST	Yes	all

[1] AST: Abstract Syntax Tree; CR: Code Review; FM: Formal Methods; Spec: Specification;
[2] SD Stage: Software Development Stage; AV: Android Version; NLP: Natural Language Processing;
[3] DAST: Dynamic Application Security Testing

The following section presents the results of the analysis process.

VII. Analysis and Evaluation Results

We analysed the selected tools according to the classification framework of section V. The results are summarised in Fig. 2, and the technical report [45]. provides a detailed description for the performed analysis and results. As mentioned in Section IV-C, we proceeded in two steps, which results are exposed below.

A. Shallow analysis results

This section presents the analysis results deduced from reading available documentation, communications and research articles. A global overview is presented in Table II.

For each studied tool, we identify the software development stage (SDLC stage), the type of covered security vulnerabilities (Focus), the analysis approach (Approach), the analysis method (Method) and its availability (Availability). We also identify the Android Version (AV) attribute that represents the version upon which the tool is constructed. This will help to measure the tools updates.

1) Considered development phases for security analysis (RQ2): It is broadly admitted that security concerns should be handled as early as possible during the application development lifecycle. Secure development life-cycle (SDLC) methodologies have been adopted by many software organisations, e.g., Microsoft through their Microsoft Security Development Lifecycle (SDL) [74], OWASP with their SDLC and Software Assurance Maturity Model (SAMM) processes [75], etc. The following outcome of our work aims at clearly identifying development phases during which studied tools could be used, thus helping developers in hardening developed applications with regards to security requirements. In our study, most identified tools can be used at one or several of the following development phases: specification, design, coding and testing, as described below.

- **Specification**: during this phase, security policies are acquired usually from natural language descriptions, stating mainly how private data should be managed. Such descriptions could be legal documents. As a consequence, application developers need to clearly state how applications collect, use, and share personal information, introducing relevant security policies in the design of the application. As shown in Table II, PoliDoid–AS, Page, Cocunut are used in the specification phase.
- **Design**: threat modelling should be part of the application design in order to allow designers to analyse the designed application architecture for potential security issues, that could be then mitigated. During this phase, issues related to the use of third-party components or libraries could also be handled. As shown in Table II, Sema through storyboards, and Vandroid through modelling are used in the design phase.
- **Coding & Testing**: analysis approaches could be used during this phase to assess the application code with regards to security vulnerabilities. As shown in Table II, Curbing, Lintent , PermitMe, Vandroid, 9Fix, AndroidLint , PerHelper , SonarLint , FindBugs, FixDroid, icc – lint , Snyk are used in the coding and testing phases. We combined the coding and testing phases here because we consider only solutions integrated in the IDE. Although coding and testing are two separate sub-processes in the software development life-cycle, in the context of our study, testing solutions serve only as a means of code review to identify coding flaws within the IDE. This classification does not include penetration tests or application post-deployment tests.

Referring to the results displayed in Table II, we classified the existing plugins by development activity. Fig. 6 maps identified tools with design phases where they can be used advantageously with regards to security enhancement during the development life cycle.

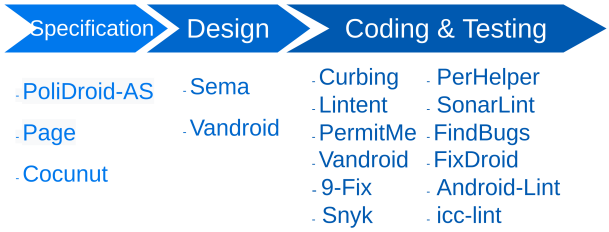


Fig. 6. Classification Per Design Level

On the one hand, we found that most of IDE plugins are considered at the *coding* phase of the development life cycle. They act as code review tools notifying developers about their ”unconscious” security issues. On the other hand, few works allowing security checks at *specification*, *design* and *testing* phases have been proposed. As a consequence, efforts towards filling this gap are expected and could allow significant enhancement in tackling security issues.

2) Used security analysis approaches (RQ3): We found that 88% of the adopted analysis approaches are static. AST analysis and formal methods are among the most used analysis methods by our sample of plugins.

- **Static AST Analysis**: Most of IDE plugins investigate statically the program Abstract Syntax Tree (AST) provided by the IDE such as SonarQube [18], Findbugs [19] and AndroidLint [20]. The goal is to extract information that enables to check the validity of predefined security properties patterns. Other tools, such as PerHelper [33], PermitMe [71] and Curbing [5] also investigate the **AST** to find the declared permissions in the application and the list of API calls requiring those permissions. The goal is to detect extra declared permissions that are not associated to any API call.
- **Formal Methods**: Other tools, such as Lintent [46] analyse the data-flow to formally check information flow with regards to security properties. Lintent [46] uses the **formal calculus** for reasoning on the Android inter-component communication API, and **type and effect** to statically prevent privilege escalation attacks on well typed components. In the same line, Sema [36] uses **formal verification** of security properties in order to generate a secure code.

As a consequence, when comparing our observations with the security analysis methods presented in Section V-C, we found that only some static ones are adopted by studied plugins. Dynamic and hybrid approaches are not referred despite their advantages. We underline this point in detail in Section VIII.

3) Considered security vulnerabilities (RQ4): The visual matrix presented in Fig. VII-A2 summarises the plugins capabilities in analysing the considered security vulnerabilities (*cf.* Section V-B). The final results of the analysis (Fig. VII-A2) cover all the vulnerabilities of all the selected IDE plugins. For each category of attacks, we present which associated vulnerabilities are covered (or not) by the tools. We used dark colors to specify the results obtained following an experimental evaluations, while light colors were used to indicate results obtained from a deep analysis of the corresponding tool's documentation. Green colors are associated to the True Positive (TP) cases. This means that the tool has detected the vulnerability that actually exists in the application. Red colors are associated to the False Negative (FN) cases. This means that the tool has not detected the vulnerability that actually exists in the application.

In this first analysis iteration, we classify the analysis difficulty in three levels:

- Tools that are specialised in a specific and unique security concern were *easy* to investigate. Based on the corresponding published papers for the plugins: Curbing [5], PermitMe [71]

and PerHelper [33]. They are clearly specialised in detecting privilege escalation attacks **(A1)** resulting from the extra use of permissions **(V4)** in the application. For other tools such as 9Fix [34], Fixdroid [73], and icc−lint [23], the list of covered vulnerabilities was explicitly declared in the related paper. So, it was easy to know that these tools detect **A3.V1** vulnerability.

- Tools specialised in detecting a specific type of attacks but the number of covered vulnerabilities is too large, are *less easy* to investigate. As an example, Lintent [46] could theoretically detect a large number of vulnerabilities as it formalises a notion of safety against privilege escalation. Based on the related published paper, it was not easy to decide whether the tool detects the vulnerability or not as the described formal model was too general. Fortunately, we found the list of covered vulnerabilities mentioned in the corresponding git repository [22]. Thus, we found that the tool covers:
  - EmptyPendingIntent **A1.V1** because the authors declares explicitly that the tool covers secrecy of pending intents,
  - Over-privilege application **A1.V4** because it contains the permission mapping for some Android APIs,
  - The remaining vulnerabilities related to privilege escalation **A1.V2**, **A1.V3**, **A1.V5**, and **A1.V6** are also covered following the github readme file where it is declared that the tool covers attack surfaces for privilege escalation,
  - It also detects **A5.V3** with Vandroid related to component hijacking.

  For Page, Coconut and POLIDROID−AS, the inputs are security requirements specified by natural languages, so they are not specialized in detecting our list of vulnerabilities.
  For Sema [36] it is explicitly declared that it covers all the vulnerabilities present in *Ghera*. However, we could not experiment the tool as the inputs of Sema are graphical storyboards and not source code.
- Finally, for industrial tools such as AndroidLint [20], SonarLint [18], Snyk [21], and findbugs [19], it was hard to investigate the covered vulnerabilities based on the documentation. The scope of these tools is too general and the documentation is too large. We found that the following vulnerabilities: **V1.A2**, **A4.V1, A4.V2, A4.V3** are covered by SonarLint. For the remaining properties, we did not find any information indicating whether they are covered by these tools or not.

After performing a long analysis process of the documentation, we decided to confirm the obtained findings by a second analysis iteration, in the form of experimental work for the available tools. For the unavailable tools, a second analysis iteration of the corresponding documentation is performed by another team member. Details of this second analysis are explained in the next subsection.

B. Deep analysis results

The objective of this part of our study is to confirm shallow analysis results with an experimental evaluation using Android application benchmarks. Compared to our previous work [1], we extended our deep analysis process to cover all vulnerabilities related to the 5

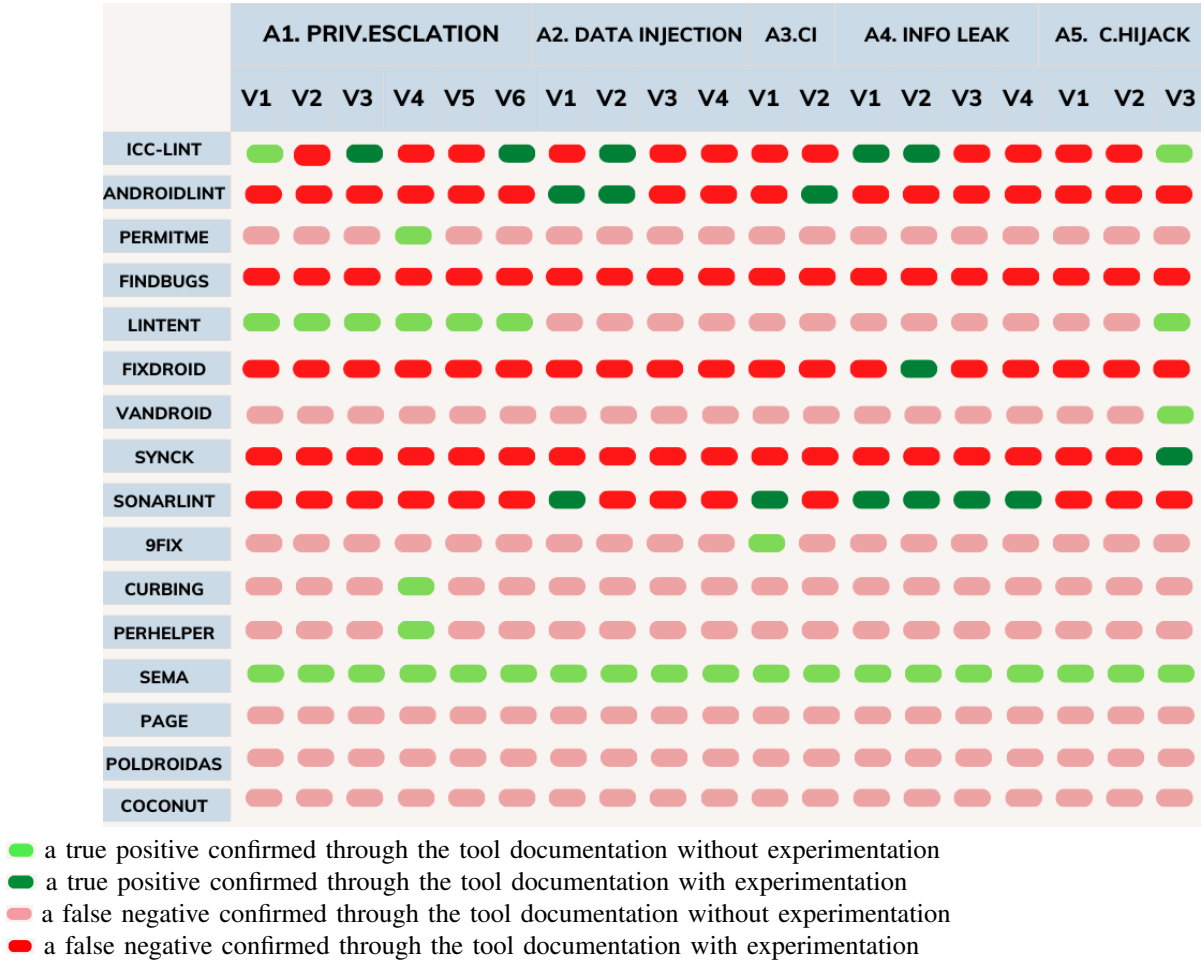


Fig. 7. Analysis Results - IDE plugins effectiveness in detecting known vulnerabilities

attack families: Privilege escalation, information leaks, data and code injection, and component hijacking.

We can observe that the deep evaluation confirmed that the following tools: Curbing [5], PermitMe [71], and PerHelper [33] are specifically oriented to detect over-privilege vulnerabilities (A1.V4); and not all the other vulnerabilities. Our deep evaluation also confirmed that none of the privilege escalation vulnerabilities are covered by SonarLint [18], FindBugs [19], FixDroid [32] and AndroidLint [20]. For SonarLint the documentation mentions the use of least privilege principle among the list of coding best practices to avoid A1.V4. However, during the analysis of the over-privileged application, SonarLint did not detect the vulnerability as shown in the technical report [45]. For icc−lint [23], the analysis of vulnerable apps revealed the presence of **A1.V3, A1.V6, A2.V2, A4.V1**, and **A4.V2** vulnerabilities. In the same line, **A1.V3** was also detected by icc−lint and fixDroid.

For A1.V2 and A5.V3, we marked the result because, even though the tool didn't detect them during the deep analysis phase, we found in the code two implemented rules for *EmptyPendingIntent* and *ImplicitPendingIntent*. We assume it is not a false negative, but rather a result explained by the fact that the tool needs maintenance.

For tools that are not available (Vandroid [35] and 9Fix [34]), an additional careful documentation-based analysis also confirmed that none of the privilege escalation vulnerabilities is covered.

All false negatives (FN) resulted from the shallow analysis iteration were confirmed by the deep analysis iteration, whether through experimental evaluation of tools or through a second round of documentation analysis. In order to provide a rigorous and explicit evaluation, each realized experimentation is described in details in the technical report [45].

As a conclusion of the deep analysis phase, we are surprised by the low detection rate of the selected IDE plugins against well known Android vulnerabilities. We confirmed that none of the studied IDE plugins covers all the vulnerabilities marked by the red color which are related to critical security attacks such as: privilege escalation, data and code injections, and sensitive information leaks. We consider this as a research gap to be tackled during future research.

VIII. Discussions

Section VII presented the analysis results, we now discuss on lessons learnt and key findings, mitigated by validity threats of our empirical study.

A. Key findings

Our analysis study raised some lessons:

- **Lack of maintenance.** Since the creation of the first version of Android on 2008, the system and the framework levels have shown many security improvements to protect users privacy. A new Android version is released every 6 months. As a consequence, most of the security assisting IDE plugins become outdated, and not able to deal any more with new types of application components, or new released APIs. Table III provides an overview of some open-source tools updates on git repositories, and considered IDEs. Four factors are of interest when considering outdated tools: (i) the date of the last commit (at the time of our study), (ii) the supported IDE type, (iii) information leaks, (iv) the integration of the tools within the last IDE versions. Besides observing that the date of the last commit for many tools is old, most tools are still supported by Eclipse only, which is no more used for developing Android applications.
- **Tools availability.** In order to assist developers in identifying and fixing the listed vulnerabilities, the availability of the security analysis tools is crucial for the software community. Indeed, among the proposed tools, only few are available for use in real Android development projects. Hence, among the 16 analysed tools, 8 academic tools are not available for use (See Table II). As an example, for over-privilege related vulnerability, among the 16 tools, 11 tools could not detect this vulnerability. On the other hand, the remaining 5 tools could detect it only at theoretical level (based on the related research paper). As a result, we do not find any solution that could effectively assist developers in detecting over-privileged applications.
- **Analysis effectiveness**. Our study shows that none of the assessed industrial plugin covers over-privilege vulnerabilities. Furthermore, tools such as Lintent, PerHelper, PermitMe are based on Fel et al. [76] permission mapping (PM) for detecting over-privileged applications. This PM is outdated and does not consider an accurate permission set. As a result, these tools will generate more false negatives during the analysis process. To overcome this limitation, these tools need to be updated to consider a newer PM that covers more API calls such as the permission mapping proposed by Dynamo [65], which is based on dynamic analysis and provides PM for last Android versions APIs.
  On the other hand, we found that some vulnerabilities detection rules do not conform to the Android specification. As an example 9Fix, which does not allow any component to be exported by changing the value of the attribut "exported" as false, and this to prevent any other application to access that component.
  Overall, we observed a dearth of tools capable of effectively detecting the most of listed vulnerabilities. Among the 19 presented vulnerabilities, only 11 are covered by the set of IDE plugins (which is the number of vulnerabilities for which the analysis yielded at least one 'dark green' result). No one of the analysed IDE plugins is able to detect the following vulnerabilities: A1.V1, A1.V2. A1.V3, A1.V4, A1.V5, A2.V3, A2.V4, A5.V1, A5.V2. Except, for some vulnerabilities, it is only theoretically possible based on the documentation, and may not be practically detectable due to the tools unavailability.
- **Analysis approaches for security**: as observed in Section V-C, most tools are based on a static analysis approach for extracting information that enables to check the validity of predefined security properties patterns. Figure 8 displays the analysis techniques used by the sample of tools we analyzed. Among these tools, 88% are based on static analysis techniques. Natural language processing techniques are used to specify security requirements in case of Page, Coconut, PoliDroid−AS. Few other tools are based on formal verification methods. And the

TABLE III
Measure tools updates based on the last git commit

Tool Name	Publication Year	Last Commit	Supported IDE
Curbing	2011	-	Ec
Lintent	2013	25/03/2013	Ec
PermitMe	2014	-	Ec
Page	2014	-	Ec
Vandroid	2018	-	Ec
Android−Lint	-	-	Ec, AS
Sema	2019	03/2020	AS
PerHelper	2019	-	IJ
PoliDroid−AS	2019	08/2019	AS, IJ
9Fix	2022	-	IJ
SonarLint	-	02/2022	Ec, AS, IJ
FindBugs	-	12/2018	Ec, AS, IJ
Cocunut	2018	09/2019	AS
FixDroid	2017	11/2017	IJ, AS
icc−lint	2018	07/2021	IJ, AS
Snyk	-	2023	IJ, AS

[1] AS: Android Studio: IJ: IntelliJ EC: Eclipse
[2] - : unknown

remaining static based analysis tools investigate the program AST to analyse the program structure and check if it conforms to the specified security rules defined by the tools. On the other hand, only 12% of the selected tools enable dynamic application security testing, which are: sonarlint and snyk. In addition to static analysis, they enable developers to examine a running build and detect problems related to security.

As a first direction of improvement, static analysis effectiveness of IDE plugin could be improved by adopting complementary analysis techniques such as Symbolic execution, to allow sound results in case of inter-component communication analysis. We were surprised to observe that none of the investigated tools takes advantage from the integrated IDE Android simulator to perform dynamic analysis. Adopting dynamic analysis approaches could be an interesting direction to improve security IDE plugin analysis results. This makes it possible to analyse API calls performed dynamically (eg. through reflection). Furthermore, other dynamic analysis techniques could be used such as dynamic code instrumentation to exploit run-time source code, and fuzzing as a software testing technique for automatic input generation.

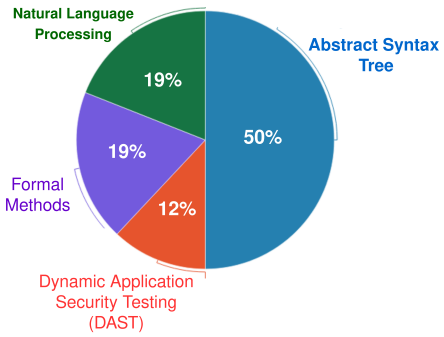


Fig. 8. Security Analysis Approach

- **The lack of native code analysis support.** We noticed a substantial shortfall in the tools' abilities to analyse native code, which is crucial for detecting vulnerabilities. For instance, overuse of permissions in Android apps cannot be accurately detected without analyzing API calls at the native code level. Currently, Per-Helper is the only plugin among the studied tools that analyzes native code, using specific regular expressions for C and C++ code. However, it only performs static analysis of the program's AST and does not provide a complete static analysis process. Additionally, it cannot detect dynamic native API calls.
- **Poor tools evaluation and validation.** After reviewing the tools' validation process, we found that most tools do not focus on the validation techniques used to confirm their ability to detect the stated vulnerabilities. Except from icc−lint that has a few unit tests for each detection rule, and Lintent, which uses formal models and rigorous validation methods to assess its analysis capabilities. None of the other tools follows a well formalised validation process, except for testing the tool on an open-source database of applications. However, there is no guarantee that the vulnerabilities are present, neither that the tool is able to detect the vulnerability.
- **Vulnerability presence per Android version** we showed through the deep analysis step that most of listed vulnerabilities (18 among 19 vulnerabilities) are prone to be exploited on last version of Android 12 (see TABLE IV) Except A2.V4 (External storage) where the vulnerability was automatically prevented by the Android system under the following versions: 12, 11, 10. However, the corresponding attack scenario of *A2.V4* is successfully launched on Android 9. This information is critical for developers of security analysis tools to target their detection rules towards specific versions of the Android API.

TABLE IV
VULNERABILITY PRESENCE PER VERSION

Vulnerability	Android version	Vulnerability	Android version
A1.V1	12	A1.V2	12
A1.V3	12	A1.V4	12
A1.V5	12	A1.V6	12
A1.V7	12		
A2.V1	12	A2.V2	12
A2.V3	12	A2.V4	9
A3.V1	12	A3.V2	12
A4.V1	12	A4.V2	12
A4.V3	12		
A5.V1	12	A5.V2	12

- **Vulnerability based vs Tool based Evaluation.** We found through deep analysis that analyzing vulnerable applications yields positive results. However, we think that a more accurate evaluation could be achieved by constructing more challenging scenarios for the tool's analysis. For certain vulnerabilities like **A1.V4**, the current scenario only considers stated permissions that have no relation to Java API calls. To properly assess the tool's detection capabilities, we need to enrich the attack scenario by adding API calls, either dynamically or through native code. This will help in evaluating the tool's effectiveness in better detecting over-privileged apps. The purpose here is to move from analyzing simple vulnerable apps to analyzing a larger attack surface sample.
- **Tools documentation.** Some tools do not mention the source of some selected vulnerabilities. In addition, no scenario validating the existence of the vulnerability was proposed. Adding a detailed description for each vulnerability, with and attack scenario demonstrating its exploitation would be a better approach to ease the use of the tool and increase the trust of end-users.
- **Benchmark availability and incompleteness**: Ghera [14] is a valuable reference for evaluating security analysis plugins that focus on open-source projects, as it implements an open-source application with common vulnerabilities. However, it lacks some vulnerabilities, such as service hijacking, and other vulnerabilities related to component hijacking. As a recommended improvement, more vulnerabilities could be found in CVE details. The goal is to enrich Ghera benchmark with new vulnerabilities. As an example, we can implement scenarios exploiting new vulnerabilities related to the manipulation of customer permissions [77]. The availability of more relevant benchmarks could lead to more comprehensive security analysis.

B. Threats to validity

In this section, we discuss various potential threats to the validity of our empirical study results, classified in the four categories of [78] according to [79].

a) Conclusion Validity: focuses on how sure we can be that the treatment we used in an experiment is really related to the actual outcome we observed [79].

- The proposed classification focuses on security tools used during: specification, design, coding and testing stages of software development. Other main stages such as: integration, deployment, and the various steps of the DevOps pipeline can also be investigated regarding the existing security tools.

- The tools are continuously evolving. Only after we conducted our evaluation, we found that a recent version of Vandroid2 has been published.
- For the included tools where all the line is in red color, e.g., POLIDROID–AS and Findbugs, this does not mean that the tool is not able to detect other vulnerabilities.

b) Internal Validity: focuses on how sure we can be that the treatment actually caused the outcome [79]. The main threat focuses on the *Tools search & selection* phase, detailed in Section IV-C.

- The set of analysis tools is not exhaustive. Despite we searched in a wide space, we may miss existing tools because the search key words differ, especially because our search target is crossing fields of the classification framework of Fig. 3. To mitigate this limitation, we crossed the references, we reviewed tools studies and related work, had a look to development forums and proceeded with snowballing to enrich the database.
- The selection (and rejection) criteria have been motivated in Section IV-C. To fix these criteria we experimented **more** plugins than those selected. All the selected tools are not dedicated to Android, some have a more general purpose but include Android vulnerability lookup. We may miss tools in that intersection field. A major weakness is that most industrial tools could not be included in the selection, and therefore some outcomes would change, surely be improved.
- The set of security vulnerabilities is not exhaustive (completeness). We based the vulnerability benchmark on the Ghera repository, referring also to CVE details because only reference repositories are sound for benchmarks. But security is a long term race and new vulnerabilities are continuously discovered.
- Besides, the selected tools target neither the same Android versions, nor the same vulnerabilities. So, we had to be very careful while extracting related information to mitigate these threats to validity. In particular, in the tool papers, the authors do not have the same reference model, and further the one we used (see also construct validity).

c) Construct Validity: focuses on the relation between the theory behind the experiment and the observation(s) [79]. While completeness is an internal validity criteria, consistency is a construct validity criteria.

- The main threat here is the heterogeneity of the available information for each tool. We can classify along a 4-level scale. (a) The minimal information is articles or scientific publications, this is a partial information to lead the shallow analysis. (b) The information collected is better with tool documentations or author answers to our queries. (c) Available and deployable tools enable experimentation. (d) Git repositories enable deep knowledge. Unfortunately we did not have the same information on each approach.
- Identification failure is important construction threat. We based the vulnerability benchmark on the Ghera repository but all the tool providers do not use this repository. We may miss vulnerabilities detected by one tool if the tool documentation is not explicit enough to identify with the Ghera vulnerability. This is a kind of homonym/synonym lexical issue. This may affect the high rate of false negatives.
- The classification framework covers many complementary points of view but the tools are not aligned on these projection axes. This may affect some interpretation.
- We used the IntelliJ IDE to process the tests, because it is of widespread use for Android development (Android Studio). But using several IDE plugins and standalone tools would provide different results (merely better if we select the union of the passing tests).
- The tools were tested against simple scenarios specified in the repository for each vulnerability. If the vulnerability could be presented in many ways in the application code and the tool implements a security detection rule that does not match the specified manner in Gherkin, then the analysis results may be inaccurate.

d) External Validity: is concerned with whether we can generalize the results outside the scope of our study [79].

- Our observations are based on the evaluation of 16 IDE plugins against 19 known vulnerabilities. Although it is a large set, it does not represent the population of the analysis tools. The study does not include commercial IDE plugins, tools that could be used out of the IDE, etc. As a result, the above observations should be considered only for similar IDE plugins, and further exploration should be conducted before generalizing the observations to all the existing tools.
- The research protocol is generic since the selected tools, the SDLC stage and the vulnerabilities can be seen as parameters. Selecting other data set for these parameters does not change the methodology, it changes the outcomes.
- It can also be extended. Web apps are not in the scope but connecting apps will extend the scope of inter-application vulnerabilities.
- This study is evolving by nature and be replayed periodically but also contributing to the Ghera repository.

The above threats show that outcome reproducibility is related to time and facts that are true now may change. However, the process itself is to be replayed periodically, just like vulnerability repositories have to be updated continuously.

IX. Conclusion and Future Work

In this paper, we presented a detailed survey of IDE plugins used for secure Android application development. Our study was motivated by the lack of existing research studies investigating their effectiveness in preventing security attacks. We conducted a rigorous and explicit evaluation process of 16 IDE plugins against 19 known vulnerabilities. We believe that the results of the survey would be useful in many cases including:

- Assisting developers in selecting the appropriate IDE plugin to use for secure Android application development,
- Guiding researchers and security tools manufacturers in identifying the existing limits in each tool, and in conducting further research related to Android vulnerabilities,
- In addition, it could be employed as an educational framework during security training sessions.

As the studied vulnerabilities are well known and already included in the list of Common Vulnerabilities Expose (CVE), we expected to have more true positives while conducting the analysis process. However, we were surprised by the obtained number of false negatives. There remains much effort to achieve the transition from Android software development life-cycle (SDLC) to Secure SDLC.

For future works, our study highlighted many observations that could benefit from improvements. First, we aim to enrich the Ghera repository with new vulnerabilities, as an example by implementing scenarios exploiting new vulnerabilities related to the manipulation of customer permissions [77]. Second, it is necessary to add more activities to embrace the full SSDLC, especially at early specification time -by defining threats, risks and scenarios that will be entry points for the application development- and lately by considering vulnerabilities at deployment time. Third, the current study focuses on native applications. To cover the Android ecosystem, it must be extended to analyze vulnerabilities related to hybrid applications, e.g., web applications. More generally, our methodology could apply to IOS applications by revisiting the identified vulnerabilities (*what* axis). Finally, to better detect the vulnerabilities identified in this survey, it is important to develop new IDE plugins that utilize effective analysis techniques.

REFERENCES

[1] M. E. A. Tebib, M. Graa, O.-E.-K. Aktouf, and P. Andre, "Ide plugins for secure android applications development: Analysis & classification study," in SECURWARE 2022: The Sixteenth International Conference on Emerging Security Information, Systems and Technologies, October 2022, pp. 48–53.

[2] Z. Ahmed and S. C. Francis, "Integrating security with devsecops: Techniques and challenges," in 2019 International Conference on Digitization (ICD). IEEE, 2019, pp. 178–182.

[3] "OWASP - Source Code Analysis Tools," Accessed: 2021-12-25. [Online]. Available: https://owasp.org/www-community/Source_Code_Analysis_Tools

[4] A. K. Jha, S. Lee, and W. J. Lee, "Developer mistakes in writing android manifests: An empirical study of configuration errors," in 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR). IEEE, 2017, pp. 25–36.

[5] T. Vidas, N. Christin, and L. Cranor, "Curbing android permission creep," in Proceedings of the Web, vol. 2, no. 0, 2011.

[6] W. Ahmad, C. Kästner, J. Sunshine, and J. Aldrich, "Inter-app communication in android: Developer challenges," in 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR). IEEE, 2016, pp. 177–188.

[7] J. Li, S. Beba, and M. M. Karlsen, "Evaluation of open-source ide plugins for detecting security vulnerabilities," in Proceedings of the Evaluation and Assessment on Software Engineering, 2019, pp. 200–209.

[8] A. Z. Baset and T. Denning, "Ide plugins for detecting input-validation vulnerabilities," in 2017 IEEE Security and Privacy Workshops (SPW). IEEE, 2017, pp. 143–146.

[9] J. Mejía, P. Maciel, M. Muñoz, and Y. Quiñonez, "Frameworks to develop secure mobile applications: A systematic literature review," in World Conference on Information Systems and Technologies. Springer, 2020, pp. 137–146.

[10] J. Senanayake, H. Kalutarage, M. O. Al-Kadri, A. Petrovski, and L. Piras, "Android source code vulnerability detection: a systematic literature review," ACM Computing Surveys (CSUR), 2022.

[11] A. Sadeghi, H. Bagheri, J. Garcia, and S. Malek, "A taxonomy and qualitative comparison of program analysis techniques for security assessment of android software," IEEE Transactions on Software Engineering, vol. 43, no. 6, pp. 492–530, 2016.

[12] B. Reaves, J. Bowers, S. A. Gorski III, O. Anise, R. Bobhate, R. Cho, H. Das, S. Hussain, H. Karachiwala, N. Scaife et al., "* droid: Assessment and evaluation of android application analysis tools," ACM Computing Surveys (CSUR), vol. 49, no. 3, pp. 1–30, 2016.

[13] V.-P. Ranganath and J. Mitra, "Are free android app security analysis tools effective in detecting known vulnerabilities?" Empirical Software Engineering, vol. 25, no. 1, pp. 178–219, 2020.

[14] "Ghera," Access 2021-12-25, https://bitbucket.org/secure-it-i/android-app-vulnerability-benchmarks/. [Online]. Available: https://bitbucket.org/secure-it-i/android-app-vulnerability-benchmarks/

[15] I. Ul Haq and T. A. Khan, "Penetration frameworks and development issues in secure mobile application development: A systematic literature review," IEEE Access, 2021.

[16] "Android references," Access 2021-12-25, https://github.com/impillar/AndroidReferences.

[17] "Android security assessment tools," Access 2021-12-25, https://bitbucket.org/secure-it-i/android-app-vulnerability-benchmarks/.

[18] "Code quality and code security," no date, accessed: 2022-03-05. [Online]. Available: https://www.sonarqube.org/

[19] "Findbugs-idea," no date, accessed: 2022-03-05. [Online]. Available: https://plugins.jetbrains.com/plugin/3847-findbugs-idea

[20] "Improve your code with lint checks," Accessed: 2022-03-05. [Online]. Available: https://developer.android.com/studio/write/lint

[21] "Developer loved, security trusted." no date, accessed: 2023-01-21. [Online]. Available: https://snyk.io/

[22] "Lintent: Towards security type-checking of android applications," no date, accessed: 2021-12-25. [Online]. Available: https://github.com/alvisespano/Lintent

[23] P. Gadient, M. Ghafari, P. Frischknecht, and O. Nierstrasz, "Security code smells in android icc," Empirical Software Engineering, vol. 24, no. 5, pp. 3046–3076, 2019.

[24] M. E. A. Tebib, P. André, O.-E.-K. Aktouf, and M. Graa, "Assisting developers in preventing permissions related security issues in android applications," in Dependable Computing-EDCC 2021 Workshops: DREAMS, DSOGRI, SERENE 2021, Munich, Germany, September 13, 2021, Proceedings 17. Springer, 2021, pp. 132–143.

[25] S. Liang, A. W. Keep, M. Might, S. Lyde, T. Gilray, P. Aldous, and D. Van Horn, "Sound and precise malware analysis for android via pushdown reachability and entry-point saturation," in Proceedings of the Third ACM workshop on Security and privacy in smartphones & mobile devices, 2013, pp. 21–32.

[26] K. Chen, P. Wang, Y. Lee, X. Wang, N. Zhang, H. Huang, W. Zou, and P. Liu, "Finding unknown malice in 10 seconds: Mass vetting for new threats at the {Google-Play} scale," in 24th USENIX Security Symposium (USENIX Security 15), 2015, pp. 659–674.

[27] E. Chin, A. P. Felt, K. Greenwood, and D. Wagner, "Analyzing inter-application communication in android," in Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, 2011, pp. 239–252.

[28] "Pmd-idea," no date, accessed: 2022-03-05. [Online]. Available: https://plugins.jetbrains.com/plugin/4596-qaplug--pmd

[29] "Checkstyle-idea," no date, accessed: 2022-03-05. [Online]. Available: https://plugins.jetbrains.com/plugin/1065-checkstyle-idea

[30] "Fortify on demand," no date, accessed: 2022-03-05. [Online]. Available: https://plugins.jetbrains.com/plugin/9943-fortify-on-demand

[31] "Checkmarx: Industry-leading application security testing," no date, accessed: 2022-03-05. [Online]. Available: https://checkmarx.com/

[32] D. C. Nguyen, D. Wermke, Y. Acar, M. Backes, C. Weir, and S. Fahl, "A stitch in time: Supporting android developers in writingsecure code," in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 1065–1077.

[33] G. Xu, S. Xu, C. Gao, B. Wang, and G. Xu, "Perhelper: Helping developers make better decisions on permission uses in android apps," Applied Sciences, vol. 9, no. 18, p. 3699, 2019.

[34] A.-D. Tran, M.-Q. Nguyen, G.-H. Phan, and M.-T. Tran, "Security issues in android application development and plug-in for android studio to support secure programming," in International Conference on Future Data and Security Engineering. Springer, 2021, pp. 105–122.

[35] A. Nirumand, B. Zamani, and B. T. Ladani, "Vandroid: A framework for vulnerability analysis of android applications using a model-driven reverse engineering technique," Softw. Pract. Exp., vol. 49, no. 1, pp. 70–99, 2019. [Online]. Available: https://doi.org/10.1002/spe.2643

[36] J. Mitra, V.-P. Ranganath, T. Amtoft, and M. Higgins, "Sema: Extending and analyzi storyboards to develop secure android apps," arXiv preprint arXiv:2001.10052, 2020.

[37] R. Slavin, X. Wang, M. B. Hosseini, J. Hester, R. Krishnan, J. Bhatia, T. D. Breaux, and J. Niu, "Toward a framework for detecting privacy policy violations in android application code," in Proceedings of the 38th International Conference on Software Engineering, 2016, pp. 25–36.

[38] M. Rowan and J. Dehlinger, "Encouraging privacy by design concepts with privacy policy auto-generation in eclipse (page)," in Proceedings of the 2014 Workshop on Eclipse Technology eXchange, 2014, pp. 9–14.

[39] "Secure software development life-cycle," accessed on 07. April 2022. [Online]. Available: https://codesigningstore.com/secure-software-development-life-cycle-sdlc

[40] R. Balebako, A. Marsh, J. Lin, J. I. Hong, and L. Cranor, "The privacy and security behaviors of smartphone app developers," Jun 2018. [Online]. Available: https://kilthub.cmu.edu/articles/journal_contribution/The_Privacy_and_Security_Behaviors_of_Smartphone_App_Developers/6470528/1

[41] G. L. Scoccia, A. Peruma, V. Pujols, I. Malavolta, and D. E. Krutz, "Permission issues in open-source android apps: An exploratory study," in 2019 19th International Working Conference on Source Code Analysis and Manipulation (SCAM). IEEE, 2019, pp. 238–249.

[42] R. Fujdiak, P. Mlynek, P. Mrnustik, M. Barabas, P. Blazek, F. Borcik, and J. Misurec, "Managing the secure software development," in 2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2019, pp. 1–4.

[43] A. Ramirez, A. Aiello, and S. J. Lincke, "A survey and comparison of secure software development standards," in 2020 13th CMI Conference on Cybersecurity and Privacy (CMI) - Digital Transformation - Potentials and Challenges(51275), 2020, pp. 1–6.

[44] R. N. Rajapakse, M. Zahedi, M. A. Babar, and H. Shen, "Challenges and solutions when adopting devsecops: A systematic review," Information and Software Technology, vol. 141, p. 106700, 2022.

[Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584921001543

[45] M. Tebib et al., "IDE plugins capabilities in detecting Android vulnerabilities," 2022, https://uncloud.univ-nantes.fr/index.php/s/mzwoC44xs5xiowN.

[46] M. Bugliesi, S. Calzavara, and A. Spanò, "Lintent: Towards security type-checking of android applications," in Formal techniques for distributed systems. Springer, 2013, pp. 289–304.

[47] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel, "Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps," Acm Sigplan Notices, vol. 49, no. 6, pp. 259–269, 2014.

[48] Y. Aafer, G. Tao, J. Huang, X. Zhang, and N. Li, "Precise android api protection mapping derivation and reasoning," in Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 1151–1164.

[49] C. Li, X. Chen, R. Sun, J. Xue, S. Wen, M. E. Ahmed, S. Camtepe, and Y. Xiang, "Natidroid: Cross-language android permission specification," arXiv preprint arXiv:2111.08217, 2021.

[50] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie, "Pscout: analyzing the android permission specification," in Proceedings of the 2012 ACM conference on Computer and communications security, 2012, pp. 217–228.

[51] R. Baldoni, E. Coppa, D. C. D'elia, C. Demetrescu, and I. Finocchi, "A survey of symbolic execution techniques," ACM Computing Surveys (CSUR), vol. 51, no. 3, pp. 1–39, 2018.

[52] C. Cadar and M. Nowack, "Klee symbolic execution engine in 2019," International Journal on Software Tools for Technology Transfer, pp. 1–4, 2020.

[53] N. Mirzaei, H. Bagheri, R. Mahmood, and S. Malek, "Sig-droid: Automated system input generation for android applications," in 2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE). IEEE, 2015, pp. 461–471.

[54] R. Spreitzer, G. Palfinger, and S. Mangard, "Scandroid: Automated side-channel analysis of android apis," in Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks, 2018, pp. 224–235.

[55] S. Lortz, H. Mantel, A. Starostin, and A. Weber, "A sound information-flow analysis for cassandra," TU Darmstadt, Tech. Rep., 2014.

[56] A. Souri, N. J. Navimipour, and A. M. Rahmani, "Formal verification approaches and standards in the cloud computing: a comprehensive and systematic review," Computer Standards & Interfaces, vol. 58, pp. 1–22, 2018.

[57] G. Iadarola, F. Martinelli, F. Mercaldo, and A. Santone, "Formal methods for android banking malware analysis and detection," in 2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS). IEEE, 2019, pp. 331–336.

[58] W. Landi, "Undecidability of static analysis," ACM Letters on Programming Languages and Systems (LOPLAS), vol. 1, no. 4, pp. 323–337, 1992.

[59] B. Chess and G. McGraw, "Static analysis for security," IEEE Security & Privacy, vol. 2, no. 6, pp. 76–79, 2004.

[60] J. Newsome and D. X. Song, "Dynamic taint analysis for automatic detection, analysis, and signaturegeneration of exploits on commodity software." in NDSS, vol. 5. Citeseer, 2005, pp. 3–4.

[61] F. Qin, C. Wang, Z. Li, H.-s. Kim, Y. Zhou, and Y. Wu, "Lift: A low-overhead practical information flow tracking system for detecting security attacks," in 2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06). IEEE, 2006, pp. 135–148.

[62] N. Nethercote and J. Seward, "Valgrind: a framework for heavyweight dynamic binary instrumentation," ACM Sigplan notices, vol. 42, no. 6, pp. 89–100, 2007.

[63] P. Oehlert, "Violating assumptions with fuzzing," IEEE Security & Privacy, vol. 3, no. 2, pp. 58–62, 2005.

[64] J. Viide, A. Helin, M. Laakso, P. Pietikäinen, M. Seppänen, K. Halunen, R. Puuperä, and J. Röning, "Experiences with model inference assisted fuzzing." WOOT, vol. 2, pp. 1–2, 2008.

[65] A. Dawoud and S. Bugiel, "Bringing balance to the force: Dynamic analysis of the android application framework," Bringing Balance to the Force: Dynamic Analysis of the Android Application Framework, 2021.

[66] O. A. V. Ravnås, "Frida: Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers," 2019.

[67] L. Dresel, M. Protsenko, and T. Müller, "Artist: the android runtime instrumentation toolkit," in 2016 11th International Conference on Availability, Reliability and Security (ARES). IEEE, 2016, pp. 107–116.

[68] A. Druffel and K. Heid, "Davinci: Android app analysis beyond frida via dynamic system call instrumentation," in International Conference on Applied Cryptography and Network Security. Springer, 2020, pp. 473–489.

[69] S. Zhao, X. Li, G. Xu, L. Zhang, and Z. Feng, "Attack tree based android malware detection with hybrid analysis," in 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications. IEEE, 2014, pp. 380–387.

[70] "Find security bugs," no date, accessed: 2023-02-11. [Online]. Available: https://plugins.jetbrains.com/plugin/3847-findbugs-idea

[71] E. Bello-Ogunu and M. Shehab, "Permitme: integrating android permissioning support in the ide," in Proceedings of the 2014 Workshop on Eclipse Technology eXchange, 2014, pp. 15–20.

[72] T. Li, Y. Agarwal, and J. I. Hong, "Coconut: An ide plugin for developing privacy-friendly apps," Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, vol. 2, no. 4, pp. 1–35, 2018.

[73] "Fixdroid: Usable security and privacy research," accessed: 2022-03-05. [Online]. Available: https://plugins.jetbrains.com/plugin/9497-fixdroid

[74] "Explore the microsoft security sdl practices," accessed on 07. April 2022. [Online]. Available: https://www.microsoft.com/en-us/securityengineering/sdl

[75] "Software assurance maturity model," accessed on 07. April 2022. [Online]. Available: https://owasp.org/www-project-samm/

[76] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in Proceedings of the 18th ACM conference on Computer and communications security, 2011, pp. 627–638.

[77] R. Li, W. Diao, Z. Li, S. Yang, S. Li, and S. Guo, "Android custom permissions demystified: A comprehensive security evaluation," IEEE Transactions on Software Engineering, 2021.

[78] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, and B. Regnell, Experimentation in Software Engineering - An Introduction, ser. The Kluwer International Series in Software Engineering. Kluwer, 2000, vol. 6. [Online]. Available: https://doi.org/10.1007/978-1-4615-4625-2

[79] R. Feldt and A. Magazinius, "Validity threats in empirical software engineering research - an initial survey," in Proceedings of the 22nd International Conference on Software Engineering & Knowledge Engineering (SEKE'2010), Redwood City, San Francisco Bay, CA, USA, July 1 - July 3, 2010. Knowledge Systems Institute Graduate School, 2010, pp. 374–379.

SPVExec and SPVLUExec - A Novel Realtime Defensive Tool for Stealthy Malware Infection

Nicholas Phillips
Department of Computer and Information Sciences
Towson University, Towson, MD, USA
nphill5@students.towson.edu

Aisha Ali-Gombe
Division of Computer Science and Engineering
Louisiana State University, Baton Rouge, LA, USA
aaligombe@lsu.edu

***Abstract*—The vicious cycle of malware attacks on infrastructures and systems has continued to escalate despite organizations' tremendous efforts and resources in preventing and detecting known threats. One reason is that standard reactionary practices such as defense-in-depth are not as adaptive as malware development. By utilizing zero-day system vulnerabilities, malware can successfully subvert preventive measures, infect its targets, establish a persistence strategy, and continue to propagate, thus rendering defensive mechanisms ineffective. In this paper, we propose sterilized persistence vectors (SPVs) - a proactive *Defense by Deception* strategy for mitigating malware infections that leverages a benign rootkit to detect changes in persistence areas. Our approach generates SPVs from infection-stripped malware code and utilizes them as persistent channel blockers for new malware infections. We performed an in-depth evaluation of our approach on Windows systems, versions 7 and 10, and Ubuntu Linux, Desktop, Server, and Core 22.0.04, by infecting them with 2000 different malware samples, 1000 per OS typing, after training the system with 2000 additional samples to fine-tune the hashing. Based on the memory analysis of pre-and post-SPV infections, our results indicate that the proposed approach can successfully defend systems against new infections by rendering the malicious code ineffective and inactive without persistence.**

Keywords*— *Malware; Rootkit; Reverse Engineering; Persistence; Defence by Deception.

I. Introduction

Malware is a continued threat against cyber systems. Characterized by stealthiness, persistence, and mutation, new-generation malware often utilizes various system vulnerabilities for infection and then leverages standard system functionality to maintain persistence. With a suitable persistence strategy, malware can remain active and prolong its existence on a host system. One of the strengths of modern malware development is its adaptability: methodologies mutate rapidly, targeting areas where security measures are weaker or nonexistent. This is true across all systems, but specifically against Windows and Linux platforms. Windows continues to hold the majority of new and unique malware samples due to its position as the most distributed OS in the marketplace, while Linux has been seeing exponential growth, growing 646 percent in samples from 2021 to 2022 [6], as shown in Figure 1. In both related literature and practice, many malware defensive techniques have been proposed - (1) antiviruses and host-based intrusion detection [33], [82], (2) integrity checking [49], [51], (2) integrity checking [31], [42], detection [10], [28], [40], [41], [49], [51], and (3) after-effect or post-mortem analysis [12], [14], [34], [44], [45], [80] of modern malware. However, as evidenced by the continued rise in stealthier attack scenarios, new samples, and variant development [19], these defensive approaches fall short of addressing a growing threat.

The common theme of these techniques is identifying the problem either before infection through signature or anomaly detection or after infection through system scans. Neither provides a general means to stop malware due to its adaptability. These ideas of a responsive or reactionary approach to detecting and preventing malware infections, in many respects, play to malware's strengths. Because of the above mentioned limitations, we propose SPVs - a Defense by Deception approach. Our methodology aims to drastically reduce malware infections by reducing the available areas of persistence for a malicious actor's exploits, including zero-day attacks. Our approach employs malware code segments to defend a target system against future infection, thus serving as a defensive mechanism. This novel technique is a drastic shift from the conventional utilization of malware code for signature detection and fingerprinting. In our proposed approach, we place blockers called SPVs in critical areas of persistence on target systems. These SPVs are persistence and deployment elements stripped from the various malware samples analyzed. Essentially, SPVs prevent a new malware infection by blocking it from writing its own vector or overwriting the persistence vector associated with already established malware. With this approach, malware loses its ability to persist and is prevented from executing its payloads and consequently propagating further. Thus, in this extended version of our prior conference paper [1], we implemented the prototype of our SPV by manually building a library of 200 payload-stripped SPVs into the Defense by Deception code base, which is then compiled into a target system and deploying at system startup. The Defense by Deception code called the *SPVExec* on Windows

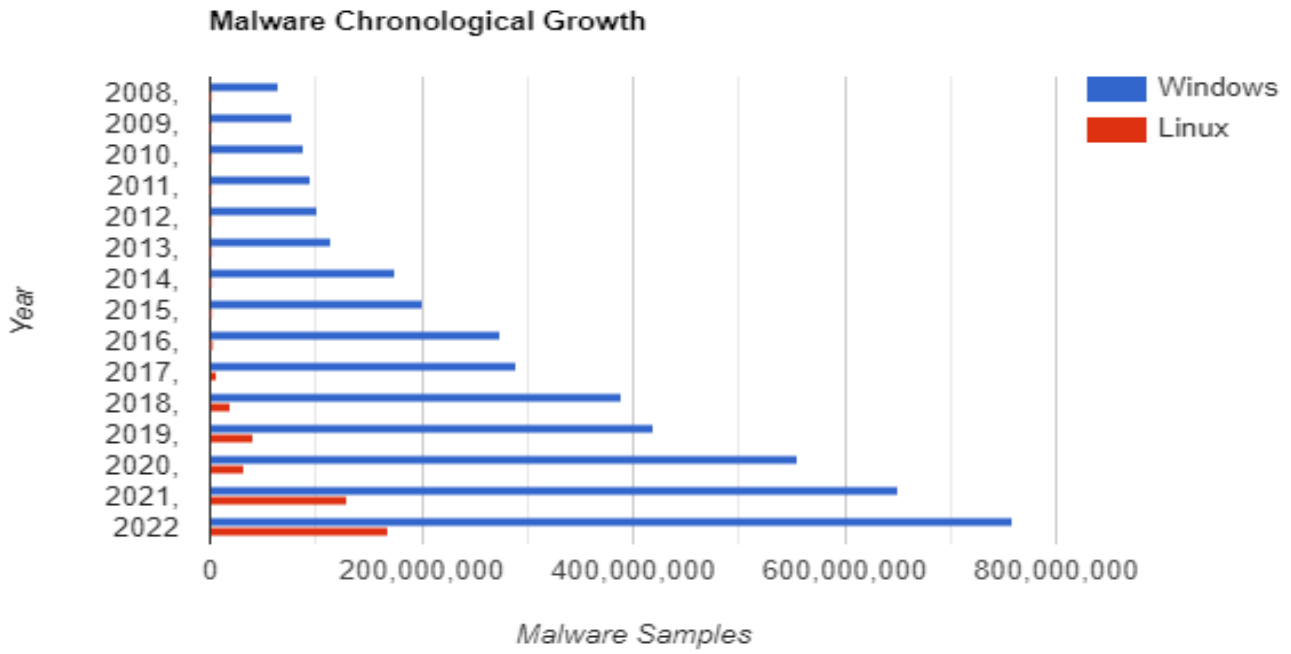


Fig. 1. Malware Growth By Year [6]

and *SPVLUEXEC* on Linux, then administered as a malware defensive apparatus on a need basis automatically at system runtime without user intervention. The empirical results of the evaluation on Windows 7 and 10, as well as Ubuntu Desktop, Server, and Core 22.0.04, for pre- and post-SPV deployment infected with 2000 malware samples, showed that the use of SPVs is a very effective strategy for malware defense. For 99% of the samples in the data set, the SPV Defense by Deception process rendered them inert - the malware sets could not execute their payloads, persist, or propagate. These blocked malware executables are also saved in a "quarantine" zone, allowing for collection and utilization in additional security tool development.

Contributions - Our proposed novel SPV strategy provides the following salient features:

- **Defense Against Malware**: Developing a practical approach to preventing new malware infections by simulating and inventing the perception that the system is already infected.
- **Fully Automated Deployment Process**: The deployment and rendering of the SPVs at runtime are done without human intervention.
- **Efficiency**: The SPV code incurs very minimal overhead on runtime system resources.
- **Usability**: The generated SPVs are reliable and seldom flagged as malware by system defense and antiviral tools. Furthermore, the proposed system allows legitimate programs to be installed without hindrance based on internal whitelisting.
- **Aided Defense Development**: Identified samples are saved and can be further analyzed for other security tool deployments.

The rest of the paper is organized as follows: Section 2 reviews the related literature; Section 3 presents the problem statement and an overview of rootkit infection; Sections 4 and 5 present the implementation of the SPV process and evaluation of our research, respectively; Section 6 details the future work; and Section 7 concludes the paper.

II. Related Work

With the rising threat of malware, the current field of work is constantly evolving, attempting to stem the problem and offer an effective form of Analysis and Defense against it. However, means of malware detection and analysis have grown more stagnant in the last ten years. Literature and current works can be divided into two main categories: Analysis and Detection/Defensive Measures.

A. Malware Analysis

Means of malware analysis have grown more stagnant in the last ten years. Windows malware analysis, in particular, has followed the main analysis structure since the early 2000s. As shown in Tahir, Alsmadi, and El Merabet [46], [47], [48], most of the improvements have been focused on implementing machine learning. This implementation is worked by classifying individual features within malware samples and rejecting

non-specific elements found within a large number of malware samples. While this is an improvement upon the standard malware detection means, there is the limitation that they are process intensive, both in the means of learning algorithms for detection and in scanning the multitude of files presented to the system. The remainder of the analysis techniques has dealt more with means of automation of the malware analysis. These are divided into two distinct areas of study, either generation of automated scripting to automate the analysis and the second is through continued utilization of machine learning to detect similar features within the malware samples.

Current literature in the analysis of malware adheres to a standard approach of malware analysis of a two-phased static and dynamic analysis approach [3]. However, others, such as Lee et al. [56], Dietz et al. [55], and Hwang et al. [57], have developed modifications to this by creating unique and independent analytical platforms. Mehdi et al. proposed Imad, an in-execution analysis platform for malware analysis. Another imported related technique proposed for malware detection is code or system-level instrumentation [4], [5]. Similar to the instrumentation is the development of Sandboxing environment. Mohino et al., in their work proposed MMALE - a Sandbox-like environment for automated execution and analysis of malware. Sandbox environments are also the focal point of Monnappa for attempts to automate malware analysis [49], [58], [59]. As with other Sandbox-based analysis techniques, there is the potential that malware authors can add code elements to detect these environments and not execute their code base. Additionally, malware authors have segmented their code bases and only contained some malicious elements in one download, as found in the research study across multiple malware samples presented in Kiachidis and Baltatzis [53]. This can cause the files in automated environments to flag them as non-malicious, even though they are the start of a malicious campaign.

In the area of machine-learning-based malware detection, the work of Jeon et al. and further developed by Kim et al., proposed deep learning for identifying similar elements of malware structure, such as similar function calls, domain addresses, string structures, etc., to try to determine the possibility of a newer sample being malicious [60], [54]. This is a tremendous step forward and can potentially speed up automated analysis. However, there is a shortcoming. These algorithms take time and a large sample base to learn the patterns in the malicious code. By the time they can identify the current trend of malicious code, malware authors can find new means to bypass these defensive measures, as shown in the evolution study presented by Cozzi et al. [61].

The SPV code does not have several limitations in the current analysis research elements. Instead of parsing through many code elements to determine maliciousness, the SPVs can target the smaller area of the malware persistence. This reduces thousands, if not millions, of code lines to a few major persistence areas. Additionally, as the SPV code is not attempting to stop the execution of the code through the current means, such as through the Sandbox analysis or process identification, several malware defensive measures are not utilized. The SPVs only need to worry about the raw malware code, not the identification of packers or encryption; it does not need to worry about antiVM and anti-RE capabilities. This leads to an extensive reduction in time and resources, which would be wasted in the analysis process. This paper presents a new SPV Defense by Deception strategy that leverages sterilized persistence vectors extracted from a real malware corpus to block potential malware infections. Our system utilizes code from malware samples, not as signatures but as defensive strategies that stop new infections from attempting to write into persistence regions. Compared to existing COTs and techniques described in the literature for malware detection and prevention, our approach is designed to be more robust and versatile, with the ability to block malware both on bare hardware and in virtualized environments. Additionally, our methodology does not require a signature or agnostic of the target malware behavior. Through an in-depth evaluation of 2000 malware samples with pre- and post-SPV infection, we demonstrate that our proposed SPV Defense by Deception mechanism can effectively defend systems against malware infections with 1-3 percent CPU and memory overhead while not limiting the ability to install legitimate programs properly.

B. Malware Detection

Malware detection methodologies can be broken down into Host-based, Hypervisor-based, and Post-mortem analysis.

1) Host-based Detection: The more traditional technique for rootkit detection is a host-based intrusion detection system that checks for anomalies or footprints of known malware. For example, the System Virginity Verifier verifies the validity of in-memory code for critical system DLLs and kernel modules; [39] checks the legitimacy of every kernel driver before it is loaded into the operating system; Panorama [40] is designed to perform behavioral runtime tracking, and SBCFI [26] detects threats by examining the control flow integrity of the kernel code. A smaller subset of methods, such as Autovac, utilizes forensics snapshot comparison engines to detect the execution of malware on the system to prevent it [38]. Other host-based rootkit detection systems include HookFinder [40] and HookMap [36]. These techniques use systematic approaches to detect and remove malware hooks in target operating systems. One offshoot of the pre-infection defensive measures proposed by Das et al. and further developed by Kedrowitsch et al. is the deployment of Docker containers as Honeypots and analyzing the behavior of the attacks to fine-tune defensive measures. By presenting these areas as more appealing targets for network attacks, security professionals can fine-tune the defensive measures on their main network components to avoid compromise [73], [75]. Shahzad et al. presented a means of detecting running malware on a Linux system by comparing the task structure of the Linux processes. By loading the kernel structures of a process, they have identified whether it is malicious or not with a minimal

impact on the system's overhead [64]. The shortcoming of this research is that once the specific modules have been removed, as shown in their evaluation, the accuracy begins to fail. Malware code has proven as it evolves to have the ability to start targeting elements that are preventing its infection, such as those proven in the study by Ngo et al. [81].

One major drawback of traditional host-based detection methodologies is the ability of the malicious entity to evade detection since it is running with the same level of privilege as the detection systems. Given that most of these tools are designed to probe for the rootkit signature and/or behavior, malware can easily subvert this effort as it evolves to have the ability to start targeting elements that are preventing its infection, such as those proven in the study by Ngo et al. [81]. The SPV code does not scan for malware footprint or traits; instead, it takes the more aggressive approach of hijacking the persistence area of a potential rootkit, leaving the malware with no place to hide. Furthermore, the SPV code is built so the malware cannot eject or terminate its process.

2) Hypervisor-based Detection: Integrity checking is a technique that requires continuous monitoring of the kernel code for changes to signatures, control flow, and kernel data structures. For kernel-level rootkits, the most practical approach for maintaining kernel integrity is hypervisor-based systems that leverage virtual machine introspection (VMI) [2], [17], [18], [28], [30], [31], [42], [43]. VMI systems and tools are built to introspect the virtual environment through the hypervisor. Since the hypervisor runs at a much lower level than the virtual OS, these mechanisms are often seen as effective for detecting rootkits and monitoring their behavior. However, their major limitation is that they target only virtualized environments and cloud infrastructures and cannot be applied to introspect real hardware-based systems. Moreover, most kernel integrity-check-based systems are susceptible to return-oriented rootkit attacks [17]. Asmitha and Vinod presented a means of malware classification based on eXtended-symmetric uncertainty. The work of [68] utilize entropy to rank features of different classes of malware and compare them against known features for malware identification. While promising research with the ability to detect nearly all cases with 99% accuracy, however, this work is limited to leveraging only the entropy. As shown in the cases of higher-level rootkits, such as those as part of the research in Raju et al. and Wang, newer malware samples can corrupt these algorithms that depend on static features [70]. Xu et al. proposed MIDAS, a real-time behavior auditing to detect malware across IoT devices. In their research, they developed a framework that analyzes the individual elements of executables as they are on the system. Once compared to the baseline, those that match the malware are flagged as malicious [71]. This research is vital because it checks for malicious elements as the sample run. However, this brings about its shortcoming: against elements of malicious code that do not match the pattern and benign software that does match the auditing requirements. Gomez et al., in their forensic analysis of IoT malware, found that several samples have become adapted to masquerading as benign programs, allowing the bypass of defensive measures [80]. The other central research element for pre-infection is the measurement of secure system installations. Sun et al. propose monitoring and protection elements during the installation phase of software deployment to minimize infection during the software deployments [76]. While this is an excellent idea, it comes to some of the same problems as other installation protection items, such as Antiviruses. Malware can compromise these checks and gain the access needed to complete their installation. Even some more robust defensive engines meant to stop improper software loading, such as SecureBoot, have been compromised, as proven in Alrawi et al. [85]. Methods used to detect the integrity of a system have been proven to be limited based on the existence of UEFI bootkits. These malicious code elements work by making the operating system accept that malicious code pieces are a legitimate portion of the system's code [11], [16], [27], [71]. With our proposed SPV Defense by Deception process, the system is designed to execute on both hardware and virtual systems, thus circumventing this limitation.

3) Post-mortem Analysis: The last category of rootkit detection methods is post-mortem analysis systems, designed to analyze the after-effects of rootkit execution. These forms of analysis are often passive and involve examining kernel memory snapshots looking for evidence of rootkit infection, persistence, and stealth. Disk forensics tools, such as [12], [14], [34], [44] are used for general system incident response. These tools can examine a target system for file modifications, running processes, network activities, and more. In much the same way as integrity checkers, disk forensics tools are limited by their coverage. If malicious code hides its elements in specific system files or structures, these will generally be missed by the post-mortem analysis [9]. With memory forensics, post-mortem analysis is carried out on a snapshot of volatile memory. The most widely used memory analysis framework is the volatility framework [45]. This methodology is restricted to current events and processes. Terminated malware behaviors cannot be retrieved. Furthermore, modern rootkits can evade detection from memory forensics tools by performing direct kernel object manipulations that hide their presence from registering in major kernel structures or by altering the memory collection or imaging process as a whole [21]. The SPV code does not scan for malware footprint or traits; instead, it takes the more aggressive approach of hijacking the persistence area of a potential rootkit, leaving the malware with no place to hide. Furthermore, the SPV code is built so the malware cannot eject or terminate its process. Compared to a more passive malware detection approach, our SPV process is an offensive approach that prevents malware infections in real-time. The SPVs are designed to block malware from executing, thus forcing the malware to terminate its process.

III. Problem Statement

Malware has always had the strength of its adaptability, which enables it to use multiple mechanisms to infect and

evade detection or bypass many of the elements of system defense [15]. Either using out-of-date signatures, exploiting unknown vulnerabilities, or targeting the weakest link - the human - malware will cause the defense to fail, even if only one falls short. Current detection and prevention tools are significantly disadvantaged because malware evolves faster than defense tools. Stealthy zero-day attacks are becoming increasingly common, and it takes only a single unknown offense or human error to bring down the whole gauntlet of defenses [20].

Thus, we present the SPV Defense by Deception process - a novel technique that attempts to hijack the areas in which malware, in general, and rootkits, in particular, can land their persistence vectors. Rootkit persistence vectors are specifically selected in this research because they are the most common persistence mechanism used by malware of all families [15].

The motivation to use persistence vectors stems from the fact that, in practice, infection vectors are unpredictable, meaning that exploits, especially zero-day exploits used to launch malware attacks, evolve with newly found vulnerabilities. However, the persistence vectors with which the malware maintains a presence on a victim's machine are often deterministic. As such, the most effective way to curtail rootkit infections and ultimately render them ineffective is to place blockers in the potentially persistent channels in the system. Long-term malware campaigns, specifically those utilized by Advanced Persistent Threats (APTs), do not wish to bring a targeted system down immediately. Instead, they want to complete target profiling against the network, exfiltrate sensitive data, and work further into the system. It can sometimes be months before the threat actors launch their final attack target. For this, they require a means to remain in the system. They require persistence. One of the longest of these types of campaigns was the Harkonnen Operation. Malicious actors could utilize their malware persistence and operate on a network for twelve years before they were finally detected. During this time, the malware implanted could assist with further target development, stealing essential data, such as corporate financial documentation, and pilfering money for the attackers [23]. Our approach injects the SPV code into the system startup process and can be rendered on bare hardware and virtualized environments. The SPV process blocks all malware by first detecting in real-time when the malware deploys its persistence vector. It then hijacks the malware area of persistence by automatically selecting and overwriting the malware code with certain SPVs. This process consistently blocks target malware from maintaining a presence on a defended system. Although our approach is currently limited to the categories of malware containing persistence vectors, "fileless" malware has only existed substantially since 2002. It is still not utilized as substantially as persistent malware [87], thus making this limitation minimal.

IV. The SPV - Defense by Deception Process

The SPV process is a code implementation of "sterilized" malware or malware with malicious content removed and injected via a common infection mechanism. It is a technique designed to prevent malware persistence on a system. SPV process involves injecting a malware persistence vector into a clean system to block potential malware from maintaining access. This process combines standing entries consisting of stripped malware persistence vectors and infection code fragments with filler code. With SPVs, the malicious payload code fragments are entirely stripped off while retaining the core elements of malware, such as API hooking, process manipulation, and service control in the SPV. Our proposed approach's workflow comprises the SPV development phase and SPVExec code deployment and integration.

A. Development Phase

This phase begins with identifying and extracting malware persistence vectors and then reprogramming the extracted persistence code fragments into one executable module.

1) Persistence Extraction: The mechanism in this stage requires manual extraction through detailed reverse engineering. We completed our reverse engineering via both static and dynamic malware analysis techniques. Malicious samples were collected from virus repositories: VirusShare [1] and Malshare [25]. One thousand samples per main OS platform were run through the two phases of reverse engineering. This was completed in a series of virtualized Windows and Linux environments. Two copies each were utilized, one for dynamic analysis and one for static analysis. These systems were identified as Testbed-1 for dynamic analysis and Testbed-2 for static analysis. Each machine had two 2.4 GHz cores and 4 GB RAM. For each target malware, we ran the sample against an unpacker for each target malware to remove any possible common packers and cryptors, leaving behind the bare-bones malware code that the analysis tools would evaluate. In this initial phase, the stripped malware code was executed in a custom-built dynamic analysis sandbox running ProcMon, CaptureBat, CFF Explorer, API Monitor, and RegShot for the Windows-based samples. The Linux samples were analyzed with X tools.

This static analysis identifies a specific part of the executable targeted during the dynamic analysis phase. Such code constructs include specific API invocation, non-normal network traffic, registry modification, and file creation. We executed the samples through a debugger and disassembler for the dynamic analysis, specifically IDAPro and OllyDbg (GDB for the Linux-based samples), targeting the identified elements in static analysis. Then, utilizing the HexRay program within IDAPro, the code section was removed and converted to a C program snippet.

2) SPV Generation: With the elements of persistence and infection identified and removed from the base malware code, we developed the SPVs. Since the identified persistence code was disassembled, we began this stage by converting the assembly code into C programming language. Upon extraction, PVs reflect specifically that individual sample of the malware, but additionally can be utilized against the majority of the

```
      // Installing the boot loader
      Status = BkSetupWithPayload(BootLoader, BootSize, Payload, PayloadSize);
      vFree(BootLoader);

      if (Status != NO_ERROR)
      {
           DbgPrint("BKSETUP: Installation failed because of unknown reason.\n");
           break;
      }

      // Creating program key to mark that we were installed
      if (RegCreateKey(HKEY_LOCAL_MACHINE, KeyName, &hKey) == NO_ERROR)
           RegCloseKey(hKey);

      Status = NO_ERROR;
      DbgPrint("BKSETUP: Successfully installed.\n");
} while(FALSE);

if (hMutex)
     CloseHandle(hMutex);

if (Payload)
     vFree(Payload);

if (KeyName)
     vFree(KeyName);

if (MutexName)
     vFree(MutexName);

if (IsExe)
     DoSelfDelete();
```

Fig. 2. Windows Extracted PV

samples of that specific malware family of that generation. For example, an extracted persistence vector from Zeus Botnet would identify that specific file and the different samples in that same generation of Zeus. Specific PVs could also be utilized against other families, dependent upon source code sampling used by the author upon its creation. Prior or future versions would require additional PV extractions depending on the evolution of the malware sample. Figure 2 show the PV extracted from Necurs Rootkit. The Necurs sample persists using multiple techniques, notably boot and registry modification implementation. These specific PVs were identified through our two-phased reverse engineering and exported for inclusion in the SPV library.

These 800 individual SPV extracted from the 1000 malware samples are loaded into the SPV Defense, including the deployment code elements. These were selected as they covered the range of persistence vectors and allowed for the broad defense of the SPVs when deployed on the system.

To build a more robust SPV defensive process, we developed an SPV library consisting of multiple SPVs.

B. SPVExec Implementation

The proposed SPV mechanism uses the extracted PVs to form a benign rootkit called SPVLUEXEC. Additional persistence scanning mechanisms were added to the code to overwrite non-whitelisted persistence modifications. Another functionality was implemented to deploy a FAT32 file system within the bootstrap code section of the system. This area was used for the SPV library, whitelisting, and the SPV Defense base code. The data remained encrypted, utilizing a 256-bit key to protect against registering on scans.

The SPVEXEC and SPVLUEXEC were implemented as single Windows EXE and Linux ELF executable programs loaded alongside the essential boot files at system startup. Each prototype is approximately 1800 lines of code in the C programming language. It is structured as follows:

- **SPV Database -** SPVs randomized for deployment across the system.
- **Defensive Measures -** Defensive elements to protect SPV code base from scans and identification of malicious code and legitimate defensive measures.
- **Dynamic White- and Blacklisting -** Included a listing of approved and disallowed changes that can be implemented on the system.
- **Analysis Mechanism -** Hash comparison against the deployed SPVs and values found in their areas.
- **SPV Launcher -** Mechanism that handles the deployment of the SPVs into their specific areas of persistence.
- **Quarantine Zone -** Area for tagged code samples for tool development.

After successfully loading the SPVExec, the persistence vectors employ two scanning techniques to validate and ensure that an intruder has not altered the injected SPVs at runtime. The first check utilizes time-based scans, similar to those employed by current protective tools. In the current implementation, this check runs a scan every second. Our

secondary scanning technique leverages API hooking to check for malware intrusion. The SPV instances are injected into kernel-level processes. Any attempts to access the protected area of persistence are redirected to one of the SPV Defended DLLs. Both scanning techniques utilize hash lookups. During SPV code deployment, a hashmap of the injected SPVs and the region of persistence are stored. The rewriters dynamically replace code elements within the SPVExec codebase and are designed to look up any changes to the injected SPVs. The dynamically computed hashes of the injected vectors are then compared against the SPVs expected to be in those regions. If no match is returned, the code rewrites those SPVs as expected.

V. Evaluation of the SPV Defense by Deception Process

We evaluate the effectiveness of our proposed SPV defense mechanism by performing four major experiments that answered the following questions:

- **Persistence of the SPV Defense process -** Can the SPV Defense survive and persist through system restarts and power removal?
- **Defense against malware -** Can the SPVs be used as an effective strategy to block potential malware from writing to protected areas of persistence?
- **Defense Through Deception -** Does the SPV Defense identify as malware to other malware and legitimate to legitimate programs?
- **System Performance -** Can the SPV Defense process be used as an efficient apparatus for system defense without depleting system resources?
- **Whitlisting Capability -** Does the SPV Defense allow legitimate programs to install without being replaced with SPV code?
- **Defense Development -** Does the SPV Defense aid with other defensive tool development?

A. Test Environment

To test SPVs across operating systems, we generated Testbed-3 and Testbed-4, utilizing Windows testbeds using the same baseline operating systems as in the persistence extraction phase, i.e., Windows for Windows and Ubuntu for Linux. They both contain sets of virtual machines and bare metal with two 2.4 GHz cores and 4 GB RAM. Testbed-1 remained at the same level of security as that of the persistence extraction environment; this removes the chance of malware failing to infect because of patching or security tools. Unlike in persistence extraction, however, this testbed has most of its nonsecurity functionality restored. This allows the system to act similarly to a standard user system that would be part of a normal network. Testbed-2, Testbed-3, and Testbed-4 are equipped with system security monitoring tools, such as operating system inbuilt Defense, i.e., Host-based Security System, and other commercial off-the-shelf antivirus products appropriate to the respective OS. For all the testbeds, user programs were installed to simulate a working system on a network, and typical applications were often targeted for compromise. To provide better containment during our analysis and testing, we implemented FakeDNS to resolve any network traffic.

B. Post-Mortem Analysis Environment

We leverage an in-depth analysis of the target systems' extracted memory snapshots to evaluate the overall SPV Defense process's accuracy, resilience, and performance. To perform forensic examinations of the memory dumps, we created a separate system equipped with FTK (Linux Memory Extractor (LIME) for Linux) and Volatility. The collection tools were also loaded on a USB to protect the data from being compromised after a malware infection. This allowed the acquisition to have a limited impact on the system while keeping the tools from being impacted by any potential built-in anti-analysis approach.

C. Experiments

1) Experiment I: Persistence: Vital to the functionality of the SPVExec benign rootkit is its ability to maintain persistence. We took the Testbed-2 system post-SPV deployment to test this functionality and saved it as "X-Security-TestingPost." We then performed a power cycle. A start-up alert was entered into the code to present a popup if the SPV remained intact. This alert displays the first SPV value and a "Hello World" message. Upon powering the system, a memory collection was completed utilizing FTK Imager. Volatility Memory Framework processed the memory image with the following plugins: psxview, malfind, ldrmodules, apihooks, dlldump, procdump, and threads. Processes and Dynamic Link Libraries (DLLs) of the SPVExec proved that it could maintain its persistence, and a popup was displayed.

For the Linux-based systems, a start-up alert was entered into the code to present a terminal displaying the first SPV value and a "Hello World" message if the SPV remained intact. Upon powering the system on, a memory collection was completed utilizing LIME. Volatility Memory Framework processed the memory image with the following plugins: Linux_psaux, Linux_malfind, Linux_pstree, Linux_kernel_opened_files, Linux_hidden_modules, Linux_procdump, and Linux_bash. Processes and shared objects of the SPVLUExec were found that proved that it could maintain its persistence and a terminal with the defined items was displayed.

Presented below in Figures 3 and 4 are the outputs from the Malfind upon the memory collection of the respective system, showing the SPV code still operating.

2) Experiment II-A: Defense Against Malware: The primary functionality of the SPVExec is its ability to stop malware attacks against the system. To provide a sufficient test of the defensive capabilities of our approach, we conducted this experiment with 1000 malware samples with diverse infection and persistence vectors and varying degrees of

```
Time Stamp: Tue Mar  7 20:22:23 2023
"C:\Users\nicho\OneDrive\Desktop\PhD\VolatilityWorkbench\vol.exe" -f "C:\Users\nicho\OneDrive\Desktop\PhD\memdump.mem" windows.malfind.Malfind
Please wait, this may take a few minutes.

Volatility 3 Framework 2.0.1
PID Process Start VPN End VPN Tag Protection CommitCharge PrivateMemory File output Hexdump Disasm
1692 dwm.exe 0x264729f0000 0x264729f0fff VadS PAGE_EXECUTE_READWRITE 1 1 Disabled
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
60 04 00 00 40 00 00 00 `...@...
00 00 9f 72 64 02 00 00 ...rd...
1692 dwm.exe 0x26478a80000 0x26478a80fff VadS PAGE_EXECUTE_READWRITE 1 1 Disabled
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
a1 03 00 00 40 00 00 00 ....@...
00 00 a8 78 64 02 00 00 ...xd...
4748 XtuService.exe 0x27ad6e00000 0x27ad6e0ffff VadS PAGE_EXECUTE_READWRITE 2 1 Disabled
00 00 00 00 00 00 00 00 ........
27 9e dc 12 6e 3a 00 01 '...n:..
ee ff ee ff 02 00 00 00 ........
20 01 e0 d6 7a 02 00 00  ...z...
20 01 e0 d6 7a 02 00 00  ...z...
00 00 e0 d6 7a 02 00 00  ...z...
00 00 e0 d6 7a 02 00 00  ...z...
0f 00 00 00 00 00 00 00 ........ 00 00 00 00 00 00 00 00 27 9e dc 12 6e 3a 00 01 ee ff ee ff 02 00 00 00 20 01 e0 d6 7a 02 00 00 20 01 e0 d6 7a 02 00 00 00 00 e0 d6 7a 02 00 00 00 00 e0 d6 7a 02 00 00 0f 00 00 00 00 00 00 00
4748 XtuService.exe 0x27ad6db0000 0x27ad6dbffff VadS PAGE_EXECUTE_READWRITE 2 1 Disabled
00 00 00 00 00 00 00 00 ........
97 cf 1e b2 fe 3b 00 01 .....;..
ee ff ee ff 02 00 00 00 ........
20 01 db d6 7a 02 00 00  ...z...
20 01 db d6 7a 02 00 00  ...z...
00 00 db d6 7a 02 00 00  ...z...
00 00 db d6 7a 02 00 00  ...z...
0f 00 00 00 00 00 00 00 ........ 00 00 00 00 00 00 00 00 97 cf 1e b2 fe 3b 00 01 ee ff ee ff 02 00 00 00 20 01 db d6 7a 02 00 00 20 01 db d6 7a 02 00 00 00 00 db d6 7a 02 00 00 00 00 db d6 7a 02 00 00 0f 00 00 00 00 00 00 00
4748 XtuService.exe 0x7ff494fd0000 0x7ff49506ffff VadS PAGE_EXECUTE_READWRITE 2 1 Disabled
d8 ff ff ff ff ff ff ff ........
08 00 00 00 00 00 00 00 ........
01 00 00 00 00 00 00 00 ........
00 02 0e 03 38 00 00 00 ....8...
08 41 d7 07 0c 00 00 00 .A......
10 68 10 3b f8 7f 00 00 .h.;....
20 40 f7 3a f8 7f 00 00  @.:.....
d0 d3 1c 3b f8 7f 00 00 ...;.... d8 ff ff ff ff ff ff ff 08 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 02 0e 03 38 00 00 00 08 41 d7 07 0c 00 00 00 10 68 10 3b f8 7f 00 00 20 40 f7 3a f8 7f 00 00 d0 d3 1c 3b f8 7f 00 00
4748 XtuService.exe 0x7ff494fc0000 0x7ff494fcffff VadS PAGE_EXECUTE_READWRITE 1 1 Disabled
00 00 00 00 00 00 00 00 ........
78 0d 00 00 00 00 00 00 x.......
0c 00 00 00 49 c7 c2 00 ....I...
00 00 00 48 b8 a0 49 58 ...H..IX
9a f8 7f 00 00 ff e0 49 .......I
c7 c2 01 00 00 00 48 b8 ......H.
a0 49 58 9a f8 7f 00 00 .IX.....
```

Fig. 3. Volatility Output Windows: Malfind

```
8852 PhoneExperienc 0x7ff81a170000 0x7ff81a18ffff Vad  PAGE_EXECUTE_READWRITE 0 0 Disabled
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
8852 PhoneExperienc 0x7ff81a160000 0x7ff81a16ffff Vad  PAGE_EXECUTE_READWRITE 0 0 Disabled
ff 25 fa 0f 00 00 4c 8b .%....L.
15 fb 0f 00 00 ff 25 fd ......%.
0f 00 00 90 66 66 66 66 ....ffff
ff 25 fa 0f 00 00 4c 8b .%....L.
15 fb 0f 00 00 ff 25 fd ......%.
0f 00 00 90 66 66 66 66 ....ffff
ff 25 fa 0f 00 00 4c 8b .%....L.
15 fb 0f 00 00 ff 25 fd ......%. ff 25 fa 0f 00 00 4c 8b 15 fb 0f 00 00 ff 25 fd 0f 00 00 90 66 66 66 66 ff 25 fa 0f 00 00 4c 8b 15 fb 0f 00 00 ff 25 fd 0f 00 00 90 66 66 66 66 ff 25 fa 0f 00 00 4c 8b 15 fb 0f 00 00 ff 25 fd
8852 PhoneExperienc 0x7ff81a150000 0x7ff81a15ffff Vad  PAGE_EXECUTE_READWRITE 0 0 Disabled
10 00 0c 20 00 00 00 00 ...  ....
09 08 28 02 00 00 00 00 ..(.....
b0 93 70 19 f8 7f 00 00 ..p.....
00 40 70 19 f8 7f 00 00 .@p.....
a0 ff 13 1a f8 7f 00 00 ........
38 49 14 1a f8 7f 00 00 8I......
48 00 15 1a f8 7f 00 00 H.......
60 00 15 1a f8 7f 00 00 `....... 10 00 0c 20 00 00 00 00 09 08 28 02 00 00 00 00 b0 93 70 19 f8 7f 00 00 00 40 70 19 f8 7f 00 00 a0 ff 13 1a f8 7f 00 00 38 49 14 1a f8 7f 00 00 48 00 15 1a f8 7f 00 00 60 00 15 1a f8 7f 00 00
8852 PhoneExperienc 0x7ff81a190000 0x7ff81a19ffff Vad  PAGE_EXECUTE_READWRITE 0 0 Disabled
10 01 0c 20 00 00 00 00 ...  ....
01 00 45 01 01 00 00 00 ..E.....
b0 93 70 19 f8 7f 00 00 ..p.....
00 40 70 19 f8 7f 00 00 .@p.....
c0 ff 15 1a f8 7f 00 00 ........
58 16 18 1a f8 7f 00 00 X.......
50 00 19 1a f8 7f 00 00 P.......
00 00 00 00 00 00 00 00 ........ 10 01 0c 20 00 00 00 00 01 00 45 01 01 00 00 00 b0 93 70 19 f8 7f 00 00 00 40 70 19 f8 7f 00 00 c0 ff 15 1a f8 7f 00 00 58 16 18 1a f8 7f 00 00 50 00 19 1a f8 7f 00 00 00 00 00 00 00 00
8852 PhoneExperienc 0x7ff81a210000 0x7ff81a21ffff Vad  PAGE_EXECUTE_READWRITE 0 0 Disabled
80 00 00 00 00 00 00 00 ........
00 00 00 00 00 00 00 00 ........
00 03 08 18 18 1f 80 dd ........
10 18 00 00 00 00 00 00 ........
28 ae 1f 1a f8 7f 00 00 (.......
10 d2 ba 19 f8 7f 00 00 ........
70 f5 1e 1a f8 7f 00 00 p.......
```

Fig. 4. Volatility Output Linux: Malfind

TABLE I: Regular Testing: Windows

Defense	TP	TN	FP	FN	Accuracy
Symantec	987	0	1	12	98.7%
Kaspersky	986	0	0	14	98.6%
Avast	984	0	1	15	98.4%
McAfee	987	0	0	13	98.7%
ESET	985	0	1	14	98.5%
SPV	999	0	0	1	99.9%

TABLE II: Regular Testing: Linux

Defense	TP	TN	FP	FN	Accuracy
Kaspersky	987	0	1	12	98.7%
BitDefender	986	0	0	14	98.6%
Avast	984	0	1	15	98.4%
McAfee	987	0	0	13	98.7%
ESET	985	0	1	14	98.5%
SPV	999	0	0	1	99.9%

TABLE III: Regression Testing: Windows

Defense	TP	TN	FP	FN	Accuracy
Symatec	500	0	25	475	50.0%
Kaspersky	475	0	90	435	47.5%
Avast	485	0	75	440	48.5%
McAfee	495	0	105	400	49.5%
ESET	480	0	120	400	48.0%
SPV	999	0	0	1	99.9%

TABLE IV: Regression Testing: Linux

Defense	TP	TN	FP	FN	Accuracy
Kaspersky	500	0	25	475	50.0%
BitDefender	475	0	90	435	47.5%
Avast	485	0	75	440	48.5%
McAfee	495	0	105	400	49.5%
ESET	480	0	120	400	48.0%
SPV	999	0	0	1	99.9%

stealthiness. We utilized Testbed-2, Testbed-3, and Testbed-4 and executed the SPVExec; the image was saved as "X-Post-SPV," with X representing the OS. Each malware sample was executed, and a snapshot and memory collection were taken. The system was then reset with the "Post-SPV" images and infected with the next malware sample. As each memory dump was analyzed with Volatility with the plugins mentioned above, the persistence elements of the SPV were found without the markers of the malware surviving. This proves that the SPV Defense prevented the malware from taking effect and rendered it inert, on the same level as other security tools. Comparisons of our process to standard antivirus software indicated that our proposed approach achieves the same level of accuracy as other COTs antiviruses as shown in Tables I and II.

D. Experiment II-B: Reversion Testing

An additional image of Testbed-2, Testbed-3, and Testbed-4 were generated for this experiment, titled "X-SecurityReversion-TestingPost." The commercial antivirus software signature libraries were downgraded by three versions lower, allowing newer malware to be tested as though it were a zero-day exploit. The sample repository listed above was run on both virtual machines. Compared to standard antivirus detection rates, SPV Defense was able to maintain consistent rates. However, during the zero-day detection experiment, it doubled the detection rates of standard antivirus software, as shown in Tables III and IV. This further proves that SPV Defense can perform far better than commercial malware detection tools against unknown threats because it only targets the persistence vectors.

1) Experiment III: Deceptive Capability: For this experiment, the SPVExec was run against two unique phases. One phase determined if malware identified SPVs as similar malware, avoiding infections. The second is if legitimate programs like Antivirus saw the SPVs as a benign code structure. The system was reverted to a save of the SPV-defended state presented in Testbed-1 for defense through deception testing. The Necurs malware sample was run against the Windows system, and The SpeakUp [84] malware sample was executed on this Linux OS. These particular samples were chosen because of a built-in function searching for already modified keys signaling an infected system. A total of ten instances of the malware were executed in attempts to infect the system; each time, memory collections were completed. Upon analysis of the memory samples via the Volatility analysis, no signs of the Necurs malware were present. Benign testing was conducted using a pool of fifteen antiviruses against the SPV code base. All tests returned negative, indicating that none of the antiviruses flagged the SPVs as malicious.

2) Experiment IV: System Performance: In this experiment, we evaluate the effectiveness of our approach on system resources, particularly the impact of the SPV Defense process on memory and CPU utilization.

(i) CPU Utilization: Utilization was recorded in two separate instances to obtain a baseline for the pre- and post-deployment system. Baseline scores for each of these system performances were recorded. Next, multiple applications were opened to simulate a typical user's desktop, including two Microsoft Word documents (LibreOffice Word documents on the Linux platforms), a single instance of Google Chrome, and one instance of the Windows or Linux file structure, depending on the system. The system was then left under these conditions for 10 minutes. In the same way, as most effective rootkits perform malicious activities without overloading the system, SPVs run in the background without exhausting CPU resources. The CPU usage overhead is on par with that of average antivirus software, or an IDS/IPS, which is approximately 2 percent on average [33].

(ii) Memory Utilization The amount of memory the SPVs

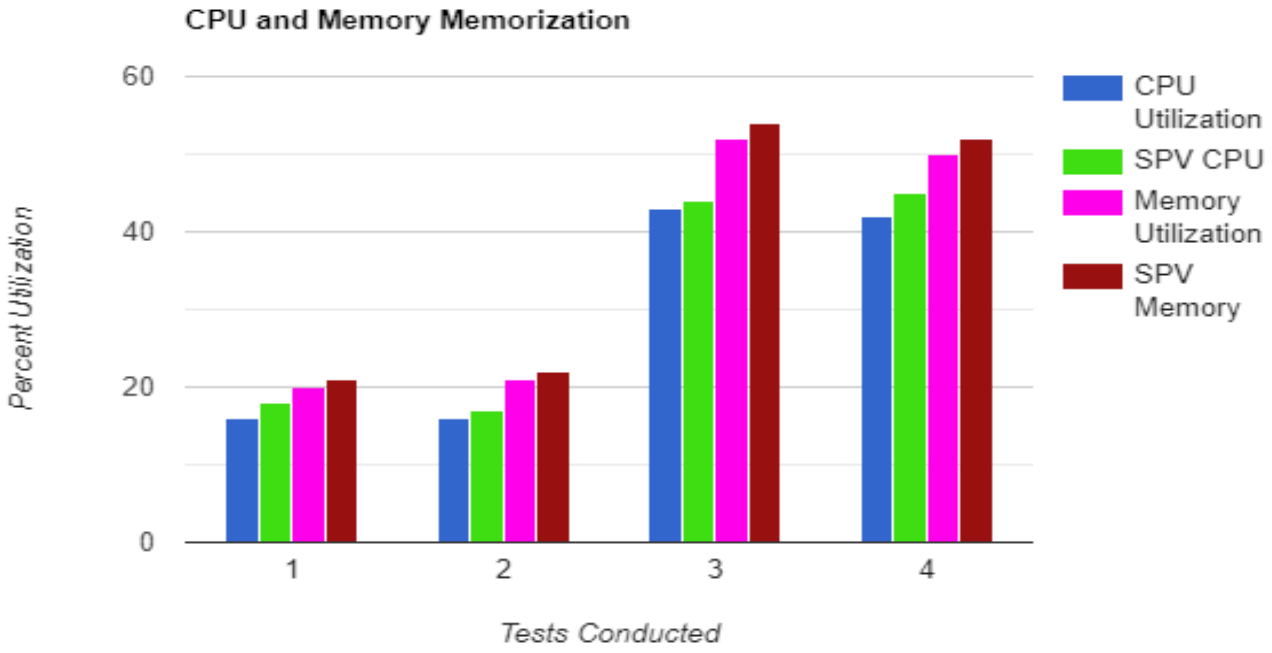


Fig. 5. Windows CPU and Memory System Comparison

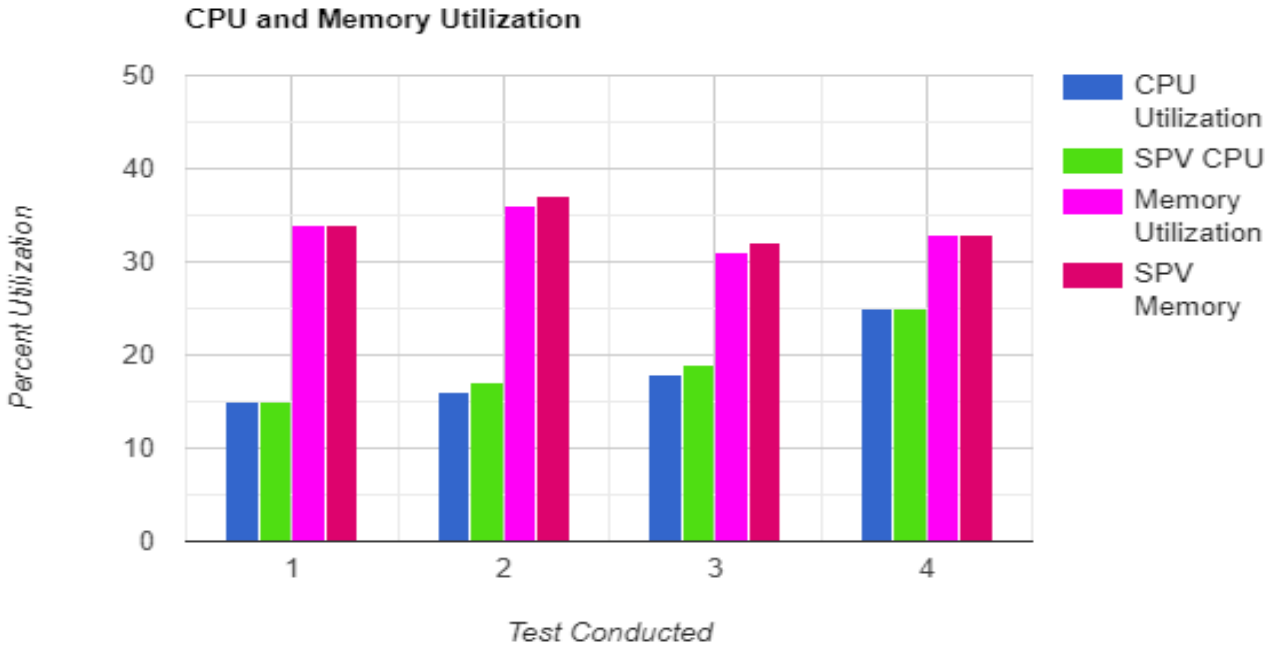


Fig. 6. Linux CPU and Memory System Comparison

utilize, specifically as they spawn processes, is also crucial. Too much memory utilization can cause an internal denial of service, making the method unusable. The baselines were again compared using the same parameters as in the CPU overhead test with the same software instances for the 10-minute implementation. This result also showed minimal impact on the system resources.

Figures 5 and 6 show a complete breakdown of these individual CPU and Memory utilization tests.

3) Experiment V: Whitelisting Capability: All the experiments conducted above proved the proposed method's ability to block future malware infections. However, this would be moot if regular benign programs could not make low-level system modifications and maintain their persistence. For this experiment, we attempted to install 10 "legitimate" programs on an SPV Defended system and determined that all were still installed after the system restart. These programs were PyCharm, Visual Studio, BitRise, Atom, BlueFish, CodePen, Crimson Editor, Eclipse, Komodo Edit, and NetBeans. The same methodology was leveraged to examine these software programs as malware to determine the system changes made to ensure their persistence. Individual snapshots from the "X-Post-SPV" series had one of the above ten programs installed. Memory collection was completed, and a snapshot was taken, titled "XPost-SPVTool," with X being the software installed. Upon powering on, a second memory collection was completed. Finally, the application was tested for functionality by launching the program. In all instances, both the SPV Defense and the program were operational and maintained persistence.

4) Experiment VI: Forensic Analysis of SPV Quarantine: Per the SPV code base, blocked malware becomes flagged and added to the equivalent of an antivirus "quarantine" zone. SPVs can collect attacks and convert information from these executables into regular defensive measures. To test this functionality, we attempted the infection with ten malware not utilized in creating the SPVs. Individually, each sample was executed against the SPV-loaded image. After each malware was launched, we took a forensics image of the test machine utilizing FTK Imager loaded on a separate drive. This process was repeated for each of the malware samples. Upon loading the evidence files into FTK Forensic Suite, all ten files were found inside the created quarantine zone.

VI. Limitations and Future Works

Our proposed SPV provides a more robust defense against malware than the existing research. However, the current implementation is limited to only core Linux and Windows OS. Additional work can be conducted into the persistence vectors that are different and unique to other OSes, which could prove beneficial. Other major operating systems, specifically Mobile OSs such as MacOS and Android, can benefit from the defense-by-deception strategy of SPVs. Thus, as part of future work, we plan to extend the current SPV to these platforms, along with improvements to the automation of the SPV generation. Additionally, research can be conducted into merging the SPVs into a universal executable, which is platform agnostic, and deploys SPVs based on an OS scan upon execution.

VII. Conclusion

This paper presents a new SPV Defense by Deception strategy that leverages sterilized persistence vectors extracted from a real malware corpus to block potential malware infections. Our system utilizes code from malware samples, not as signatures but as defensive strategies that stop new infections from attempting to write into persistence regions. Compared to existing COTs and techniques described in the literature for malware detection and prevention, our approach is designed to be more robust and versatile, with the ability to block malware both on bare hardware and in virtualized environments. Additionally, our methodology does not require a signature or agnostic of the target malware behavior. Through an in-depth evaluation of 2000 malware samples with pre- and post-SPV infection, we demonstrate that our proposed SPV Defense by Deception mechanism can effectively defend systems against malware infections with 1-3 percent CPU and memory overhead while not limiting the ability to install legitimate programs properly.

References

[1] N. Phillips and A. Ali Gombe, "Sterilized Persistence Vectors (SPVs): Defense Through Deception on Windows Systems," in Proc. CYBERWARE, 2022, pp. 56-61.

[2] I. Ahmed, A. Zoranic, S. Javaid, and G.G. Richard III, "Modchecker: Kernel module integrity checking in the cloud environment," In 2012 41st International Conference on Parallel Processing Workshops, Sep. 2012, pp. 306-313, IEEE.

[3] A. Ali-Gombe, I. Ahmed, G.G. Richard III, and V. Roussev, "AspectDroid: Android app analysis system," In Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, pp. 145-147, 2016.

[4] A. Ali-Gombe, B. Saltaformaggio, D. Xu, and G.G. Richard III, "Toward a more dependable hybrid analysis of Android malware using aspect-oriented programming," Computers & Security, vol. 73, pp. 235-248, 2018.

[5] A. Ali-Gombe, I. Ahmed, G.G. Richard III, and V. Roussev, "Opseq: Android malware fingerprinting." In Proceedings of the 5th Program Protection and Reverse Engineering Workshop, Dec. 2015, pp. 1-12.

[6] Z. Gittins and M. Soltys, "Malware persistence mechanisms," Procedia Computer Science, vol. 176, pp. 88-97. Jan. 2020.

[7] M.U. Rana, M.A. Shaha, and O. Ellahi, "Malware Persistence and Obfuscation: An Analysis on Concealed Strategies," In 2021 26th International Conference on Automation and Computing (ICAC), Sep. 2021, pp. 1-6, IEEE.

[8] B.V. Prasanthi, "Cyber forensic tools: a review," International Journal of Engineering Trends and Technology (IJETT), vol. 41(5), pp. 266-271, 2016.

[9] M. Carbone, W. Cui, L. Lu, W. Lee, M. Peinado, and X. Jiang, "Mapping kernel objects to enable systematic integrity checking," In Proceedings of the 16th ACM conference on Computer and communications security, Nov. 2009, pp. 555-565.

[10] E. Chan, S. Venkataraman, F. David, A Chaugule, and R. Campbell, "Forenscope: A framework for live forensics," In Proceedings of the 26th Annual Computer Security Applications Conference, Dec. 2010, pp. 307-316.

[11] B.N. Flatley, "Rootkit Detection Using a Cross-View Clean Boot Method," AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH GRADUATE SCHOOL OF ENGINEERING AND MANAGEMENT, Mar. 2013.

[12] S.L. Garfinkel, "Automating disk forensic processing with SleuthKit, XML, and Python," In 2009 Fourth International IEEE Workshop on Systematic Approaches to Digital Forensic Engineering, May 2009, pp. 73-84, IEEE.

[13] Z. Gu, B. Saltaformaggio, X. Zhang, and D. Xu, "Face-change: Application-driven dynamic kernel view switching in a virtual machine," In 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Jun. 2014, pp. 491-502, IEEE.

[14] D. Byers and N. Shahmehri, "A systematic evaluation of disk imaging in EnCase® 6.8 and LinEn 6.1," Digital Investigation 6.1-2, 2009, pp. 61-70.

[15] I.U. Haq, S. Chica, J. Caballero, and S. Jha, "Malware lineage in the wild," Computers & Security, vol. 78. pp. 347-363, Sep. 2018.

[16] O.S. Hofmann, A.M. Dunn, S. Kim, I. Roy, and E. Witchel, "Ensuring operating system kernel integrity with OSck," ACM SIGARCH Computer Architecture News, vol. 39(1), pp. 279-290, 2021.

[17] R. Hund, T. Holz, and F.C. Freiling, "Return-oriented rootkits: Bypassing kernel code integrity protection mechanisms," InUSENIX security symposium, Aug. 2009, pp. 383-398.

[18] X. Jiang, X. Wang, and D. Xu, "Stealthy malware detection through VMM-based 'out-of-the-box'semantic view," In 14th ACM Conference on Computer and Communications Security (CCS), Alexandria, VA, Nov. 2007, pp. 128-138.

[19] A. Kapoor and R. Mathur, "Predicting the future of stealth attacks," In Virus Bulletin Conference, Oct. 2011, pp. 1-9.

[20] J.D. Kornblum and C.F. ManTech, "Exploiting the rootkit paradox with Windows memory analysis," International Journal of Digital Evidence, vol. 5(1), pp. 1-5, 2006.

[21] T.K. Lengyel, S. Maresca, B.D. Payne, G.D. Webster, S. Vogl, and A. Kiayias, "Scalability, fidelity, and stealth in the DRAKVUF dynamic malware analysis system," In Proceedings of the 30th annual computer security applications conference, Dec. 2014, pp. 386-395.

[22] L. Litty, H.A. Lagar-Cavilla, and D. Lie, "Hypervisor Support for Identifying Covertly Executing Binaries," In USENIX Security Symposium, Jul. 2008, vol. 22, p. 70.

[23] R. Luh, S. Schrittwieser, and S. Marschalek, "TAON: An ontology-based approach to mitigating targeted attacks," In Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services, Nov. 2016, pp. 303-312.

[24] D. Patten, The evolution to fileless malware, 2017.

[25] Malshare, www.malshare.com, Oct. 2019.

[26] N.L. Petroni Jr. and M. Hicks, "Automated detection of persistent kernel control-flow attacks," In Proceedings of the 14th ACM conference on Computer and communications security, Oct. 2007, pp. 103-115.

[27] F. Raynal, Y. Berthier, P. Biondi, and D. Kaminsky, "Honeypot forensics," In Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, Jun. 2004, pp. 22-29. IEEE.

[28] R. Riley, X. Jiang, and D. Xu, "Guest-transparent prevention of kernel rootkits with VMM-based memory shadowing," In International Workshop on Recent Advances in Intrusion Detection, Sep. 2008, pp. 1-20, Springer, Berlin, Heidelberg.

[29] J. Rutkowska, "System virginity verifier: Defining the roadmap for malware detection on Windows systems," In Hack in the Box security conference, Sep. 2005.

[30] M. Schmidt, L. Baumgartner, P. Graubner, D. Bock, and B. Freisleben, "Malware detection and kernel rootkit prevention in cloud computing environments," In 2011 19th International Euromicro Conference on Parallel, Distributed, and Network-Based Processing, Feb. 2011, pp. 603-610. IEEE.

[31] A. Seshadri, M. Luk, N. Qu, and A. Perrig, "SecVisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity OSes," In Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles, Oct. 2007, pp. 335-350.

[32] M.I. Sharif, W. Lee, W. Cui, and A. Lanzi, "Secure in-VM monitoring using hardware virtualization," In Proceedings of the 16th ACM conference on Computer and communications security, Nov. 2009, pp. 477-487.

[33] O. Sukwong, H. Kim, and J. Hoe, "Commercial antivirus software effectiveness: an empirical study," Computer, vol. 44(03), pp. 63-70, Mar. 2011.

[34] S. Vömel and H. Lenz, "Visualizing indicators of Rootkit infections in memory forensics," In 2013 Seventh International Conference on IT Security Incident Management and IT Forensics, Mar. 2013, pp. 122-139, IEEE.

[35] J. Wang, A. Stavrou, and A. Ghosh, "Hypercheck: A hardware-assisted integrity monitor," In International Workshop on Recent Advances in Intrusion Detection, Sep. 2010, pp. 158-177, Springer, Berlin, Heidelberg.

[36] Z. Wang, X. Jiang, W Cui, and X. Wang, "Countering persistent kernel rootkits through systematic hook discovery," In International Workshop on Recent Advances in Intrusion Detection, Sep. 2008, pp. 21-38, Springer, Berlin, Heidelberg.

[37] M. Xu, X. Jiang, R. Sandhu, and X. Zhang, "Towards a VMM-based usage control framework for OS kernel integrity protection," In Proceedings of the 12th ACM symposium on Access control models and technologies, Jun. 2007, pp. 71-80.

[38] Z. Xu, J. Zhang, G. Gu, and Z. Lin, "Autovac: Automatically extracting system resource constraints and generating vaccines for malware immunization," In 2013 IEEE 33rd International Conference on Distributed Computing Systems, Jul. 2013, pp. 112-123, IEEE.

[39] J. Rutkowska, "System virginity verifier: Defining the roadmap for malware detection on Windows systems," In Hack in the Box security conference, Sep. 2005.

[40] H. Yin, Z. Liang, and D. Song, "HookFinder: Identifying and understanding malware hooking behaviors," Proceedings of the Network and Distributed System Security Symposium, NDSS 2008, San Diego, Feb. 2008

[41] H. Yin, D. Song, M. Egele, C. Kruegel, and E. Kirda, "Panorama: capturing system-wide information flow for malware detection and analysis," In Proceedings of the 14th ACM conference on Computer and communications security, Oct. 2007, pp. 116-127.

[42] H.A. Lagar-Cavilla and L. Litty, "Patagonix: Dynamically Neutralizing Malware with a Hypervisor," 2008.

[43] Y. Oyama, T.T. Giang, Y. Chubachi, T. Shinagawa, and K. Kato, "Detecting malware signatures in a thin hypervisor," In Proceedings of the 27th Annual ACM Symposium on Applied Computing, Mar. 2012, pp. 1807-1814.

[44] O. Vermaas, J. Simons, and R. Meijer, "Open computer forensic architecture a way to process terabytes of forensic disk images,"

In Open Source Software for Digital Forensics, 2010, pp. 45-67, Springer, Boston, MA.

[45] A. Mohanta and A. Saldanha, "Memory Forensics with Volatility," In Malware Analysis and Detection Engineering, 2020, pp. 433-476, Apress, Berkeley, CA.

[46] R. Tahir, "A study on malware and malware detection techniques," International Journal of Education and Management Engineering, vol. 8(2). p. 20, Mar. 2018.

[47] N. Idika and A.P. Mathur, "A survey of malware detection techniques," Purdue University, vol. 48(2). pp. 32-46, Feb. 2007.

[48] H. El Merabet and A. Hajraoui, "A survey of malware detection techniques based on machine learning," International Journal of Advanced Computer Science and Applications, vol. 10(1), 2019.

[49] K. Monnappa, "Automating Linux malware analysis using limon sandbox," Black Hat Europe, 2015, IV-A.

[50] M. Alrammal, M. Naveed, S. Sallam, and G. Tsaramirsis, "Malware analysis: Reverse engineering tools using santuko Linux," Materials Today: Proceedings, vol. 60, pp. 1367-1378, 2022.

[51] A. Ravi and V. Chaturvedi, "Static Malware Analysis using ELF features for Linux-based IoT devices," In 2022 35th International Conference on VLSI Design and 2022 21st International Conference on Embedded Systems (VLSID), Feb. 2022, pp. 114-119, IEEE.

[52] D. Serpanos, P. Michalopoulos, G. Xenos, and V. Ieronymakis, "Sisyfos: A modular and extendable open malware analysis platform," Applied Sciences, Vol. 11(7), p. 2980, 2021.

[53] I.G. Kiachidis and D.A. Baltatzis, "Comparative Review of Malware Analysis Methodologies," arXiv preprint arXiv:2112.04006, 2021.

[54] J. Kim, Y. Ban, G. Jeon, Y.G. Kim, and H. Cho, "LiDAR: A Light-Weight Deep Learning-Based Malware Classifier for Edge Devices," Wireless Communications and Mobile Computing, 2022.

[55] C. Dietz, M. Antzek, G. Dreo, A. Sperotto, and A. Pras, "DMEF: Dynamic Malware Evaluation Framework," In NOMS 2022 - IEEE/IFIP Network Operations and Management Symposium, Apr. 2022, pp. 1-7, IEEE.

[56] S. Lee, H. Jeon, G. Park, J. Kim, and J.M. Youn, "IoT Malware Static and Dynamic Analysis System," Journal of Human-centric Science and Technology Innovation, Vol. 1(1), pp. 43-48, 2021.

[57] C. Hwang, J. Hwang, J. Kwak, and T. Lee, "Platform-independent malware analysis applicable to Windows and Linux environments," Electronics, vol. 9(5), p. 793, 2020.

[58] J.J. De Vicente Mohino, J. Bermejo-Higuera, J.R. Bermejo Higuera, J.A. Sicilia, M. Sánchez Rubio, and J.J. Martínez Herraiz, "MMALE a methodology for malware analysis in Linux environments," 2021.

[59] S.B. Mehdi, A.K. Tanwani, and M. Farooq, "Imad: in-execution malware analysis and detection," In Proceedings of the 11th Annual Conference on Genetic and evolutionary computation, pp. 1553-1560, 2009.

[60] J. Jeon, J.H. Park, and Y.S. Jeong, "Dynamic analysis for IoT malware detection with convolution neural network model," IEEE Access, vol. 8, pp. 96899-96911, 2020.

[61] E. Cozzi, "Binary Analysis for Linux and IoT Malware," Doctoral dissertation, Sorbonne Université, 2020.

[62] K.A. Asmitha and P. Vinod, "A machine learning approach for Linux malware detection," In 2014 International conference on issues and challenges in intelligent computing techniques (ICICT), 2014, pp. 825-830, IEEE.

[63] M. Kumar, "Scalable malware detection system using big data and distributed machine learning approach," Soft Computing, vol. 26(8), pp. 3987-4003, 2022.

[64] F. Shahzad, S. Bhatti, M. Shahzad, and M. Farooq, "In-execution malware detection using task structures of Linux processes," In 2011 IEEE International Conference on Communications (ICC), 2011, pp. 1-6. IEEE.

[65] I. Vurdelja, I. Blažić, D. Drašković, and B. Nikolić, "Detection of Linux Malware Using System Tracers-An Overview of Solutions," IcEtran, 2020.

[66] S.M.P. Dinakarrao, H. Sayadi, H.M. Makrani, C. Nowzari, S. Rafatirad, and H. Homayoun, "Lightweight node-level malware detection and network-level malware confinement in IOT networks," In 2019 Design, Automation and Test in Europe Conference and Exhibition, 2019, pp. 776-781, IEEE.

[67] T. Landman, and N. Nissim, "Deep-Hook: A trusted deep learning-based framework for unknown malware detection and classification in Linux cloud environments," Neural Networks, vol. 144, pp. 648-685, 2021.

[68] K.A. Asmitha, and P. Vinod, "Linux malware detection using eXtended–symmetric uncertainty," In International Conference on Security, Privacy, and Applied Cryptography Engineering, 2014, pp. 319-332, Springer, Cham.

[69] K.A. Asmitha and P. Vinod, "Linux malware detection using non-parametric statistical methods," In 2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2014, pp. 356-361, IEEE.

[70] A. D. Raju, I. Y. Abualhaol, R.S. Giagone, Y. Zhou, and S. Huang, "A survey on cross-architectural IOT malware threat hunting," IEEE Access, vol. 9, pp. 91686-91709, 2021.

[71] Y. Xu, Z. Yin, Y. Hou, J. Liu, and Y. Jiang, "MIDAS: Safeguarding IoT Devices Against Malware via Real-Time Behavior Auditing," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 41(11), pp. 4373-4384, 2022.

[72] A. Mishra, A. Roy, and M.K. Hanawal, "Evading Malware Analysis Using Reverse Execution," In 2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS), 2022, pp. 1-6. IEEE.

[73] S. Das, H. Xiao, Y. Liu, and W. Zhang, "Online malware defense using attack behavior model," In 2016 IEEE International Symposium on Circuits and Systems (ISCAS), 2016, pp. 1322-1325. IEEE.

[74] F. Shahzad, M. Shahzad, and M. Farooq, "In-execution dynamic malware analysis and detection by mining information in process control blocks of Linux OS," Information Sciences, vol. 231, pp. 45-63, 2013.

[75] A. Kedrowitsch, D. Yao, G. Wang, and K. Cameron, "A first look: Using Linux containers for deceptive honeypots," In Proceedings of the 2017 Workshop on Automated Decision Making for Active Cyber Defense, 2017, pp. 15-22.

[76] W. Sun, R. Sekar, Z. Liang, and V.N. Venkatakrishnan, "Expanding malware defense by securing software installations," In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 164-185, Springer, Berlin, Heidelberg, 2008.

[77] S.M.P. Dinakarrao et al., "Adversarial attack on microarchitectural events based malware detectors," In Proceedings of the 56th Annual Design Automation Conference, 2019, pp. 1-6.

[78] M. Zhang, A. Raghunathan, and N.K. Jha, "A defense framework against malware and vulnerability exploits," International Journal of information security, vol. 13(5), pp. 439-452, 2014.

[79] V. Chierzi and F. Mercês, "Evolution of IoT Linux Malware: A MITRE ATTandCK TTP Based Approach," In 2021 APWG

Symposium on Electronic Crime Research (eCrime), 2021, pp. 1-11, IEEE.

[80] J.M.C. Gómez, J.R. Gómez, J.L.M. Martínez, and A. del Amo Mínguez, "Forensic Analysis of the IoT Operating System Ubuntu Core," In Journal of Physics: Conference Series, vol. 2224, no. 1, p. 012082, IOP Publishing, 2022.

[81] Q.D. Ngo, H.T. Nguyen, V.H. Le, and D.H. Nguyen, "A survey of IoT malware and detection methods based on static features," ICT Express, vol. 6(4), pp. 280-286, 2020.

[82] H. Wang, W. Zhang, H. He, P. Liu, D.X. Luo, Y. Liu, and X. Lan, "An evolutionary study of IoT malware," IEEE Internet of Things Journal, vol. 8(20), pp. 15422-15440, 2021.

[83] C, R. (2022, July 27), "Linux malware trends 2022 H1." Infogram, Retrieved January 5, 2023, from https://infogram.com/linux-malware-trends-2022-h1-1ho16vowkmwm84n

[84] S. Popoveniuc, "Speakup: remote unsupervised voting," Industrial Track ACNS, 2010.

[85] O. Alrawi, C. Lever, K. Valakuzhy, R. Court, K.Z. Snow, F. Monrose, and M. Antonakakis, "The Circle Of Life: A Large-Scale Study of The IoT Malware Lifecycle," In USENIX Security Symposium, 2021, pp. 3505-3522.

[86] A.D. Raju, I.Y. Abualhaol, R.S. Giagone, Y. Zhou, and S. Huang, "A survey on cross-architectural IOT malware threat hunting," IEEE Access, vol. 9, pp. 91686-91709, 2021.

[87] D. Patten, "The evolution to fileless malware," 2017.

[88] T. Steffens, "Attribution of Advanced Persistent Threats," Springer Berlin Heidelberg, pp. 153-164, 2020.

[89] L.E.S. Jaramillo, "Malware detection and mitigation techniques: Lessons learned from Mirai DDOS attack," Journal of Information Systems Engineering & Management, vol. 3(3), p. 19, 2018.

Runtime Trustworthiness Evaluation of Evolving Cyber Physical Systems

Rainer Falk and Steffen Fries
Siemens AG
Technology
Munich, Germany
e-mail: {rainer.falk|steffen.fries}@siemens.com

Abstract—**The integrity of Cyber Physical Systems (CPS) needs to be protected to ensure a reliable, trustworthy operation. The hardware and software components of a CPS must be in a well-defined, approved configuration state. However, such system integrity protection becomes increasingly challenging with CPSs that are flexibly reconfigured to address evolving demands. An approach for integrity monitoring for such dynamic CPSs is described. Instead of preventing changes to a CPS, the focus is on detecting changes and on analyzing and checking whether the detected changes are in-line with a policy defining permitted changes. A key element is a reliable device lifecycle state attestation, so that a CPS integrity monitoring system can determine the current configuration state of CPS components and the way in which it was changed.**

Keywords–system integrity; trustworthiness; device integrity; attestation; lifecycle; resilience; cyber physical systems; Internet of Things; cyber security.

I. INTRODUCTION

The integrity and resilience of Cyber Physical Systems (CPS), e.g., technical automation and control systems, are highly relevant security objectives [1]. Unauthorized changes the configuration of a CPS have to be prevented as well as detected. Related security requirements are defined by the industrial security standard IEC62443 [2]. Such security objectives could even be pushed by related regulative requirements, as can be seen, e.g., in the proposed update to the EU Network Information Security (NIS) directive [3].

A concept for enhanced integrity monitoring of overall industrial automation and control systems, combining integrity monitoring from physical processes up to its control and support systems, has been described in [4]. Enhanced attack resilience allows an operator to keep the CPS operational, possibly with some limitations, even during an ongoing attack [5]. Particularly challenging are CPSs with a dynamically changing configuration as driven by the flexibility of IIoT and Industry 4.0. Cyber systems will become more open and dynamic to support flexible production down to "lot size 1" by supporting plug-and-work reconfiguration of manufacturing equipment and flexible adaptation of production systems to changing needs, and by increasingly adopting software-based automation and control functions. This implies that also security has to support such dynamically CPSs that are evolving over time in a practical way.

In the past, CPS have been often rather static. After being put into operation, changes to the configuration happen only rarely, e.g., to replace a defect component, or to install smaller upgrades during a planned maintenance window. To cope with increasing demands for flexible production and increased productivity, CPS will also increasingly become more dynamic, allowing for reconfiguration during regular operation. Such scenarios for highly adaptive production system that can be adapted flexibly to changing production needs have been described in the context of Industry 4.0 [6]. The flexibility starts at the device level, where smart devices allow for upgrading and enhancing the device functionality by user-downloadable apps, and by the increasing software-based realization of automation and control functions. Besides the device level, also the system of interconnected machines is reconfigured according to changing needs. Examples are Software Defined Networks (SDN) enabling a fast reconfiguration of the communication infrastructure to adapt flexibly to the communication needs and the use of wireless communications as wireless LAN of private 5G networks. Another example relates to manufacturing systems (e.g., robots) in industrial automation systems, where smart tools are attached to a robot that in turn feature also a local communication network connecting to the robot's network.

The focus of cyber security is protection against cyber attacks, their detection, and the recovery from successful cyber attacks. An increasingly important further aspect is trustworthiness, where automated checks verify whether the overall systems and the used components meet the explicitly defined trustworthiness criteria. However, the concept of trustworthiness is subjective. The presented approach checks for changes within a CPS to determine whether the CPS configuration is in a permitted, trustworthy state.

Section II gives an overview on related work. After describing shortly industrial CPS in Section III, previous work on protecting integrity of cyber physical systems and their components is summarized in in Section IV. The monitoring of reliable device lifecycle information based on lifecycle state attestations is described in Sections V and VI, extending CPS integrity monitoring information. Approaches for analyzing detected lifecycle state attestations are described Section VII. Section VIII evaluates the presented approach. Section IX concludes the paper and gives an outlook towards future research.

II. Related Work

The objective of CPS system integrity and CPS resilience is to support the trustworthiness of CPS. While not new, the concept of trustworthiness is gaining increasing interest in ongoing research and standardization: The standard ISO/IEC TS 5723 [7] published in 2022 defines trustworthiness of systems and the characteristics of trustworthiness, addressing products, services, technologies as well as the trustworthiness of organizations that are providing these. A common understanding and description of trustworthiness characteristics allows stakeholders to judge whether their trustworthiness expectations are met. Mohammadi describes in [8] a trustworthiness framework for CPS that covers development phases like requirements engineering and system design, but also run-time maintenance, and evidence-based assurance. Also, evaluation of trustworthiness during CPS runtime is covered by monitoring its trustworthiness properties. Jiang proposed a data-driven vulnerability analysis for CPS using machine learning [9]. Northern, Burks, Hatcher, Rogers, and Ulybyshev described a methodology to determine a hardened CPS configuration by analyzing cyber vulnerabilities [10]. Cyber risk scores for different CPS configurations are compared, and vulnerable CPS components are replaced or reconfigured. Malik and Tosh described a framework for the dynamic risk assessment and analysis of CPS using multi-formatted knowledge bases derived from open-source vulnerability databases [11]. M. Tapia, P. Thier, S. Gößling-Reisemann performed an empirical study on the vulnerability and resilience of cyber-physical power systems [12]. A resilience management approach is proposed that targets a better handling of CPS failures. The proposed resiliency measures address the categories technology, organizational security policies and procedures, human factor, and regulations. Akbarzadeh and Katsikas described a cybersecurity risk assessment method that addresses the interactions and interdependencies between the cyber and the physical components using a model of the CPS and its components [13].

Requirements related to resilience on device level have been addressed in different standards. The Trusted Computing Group (TCG) specified requirements for cyber resilient modules and building blocks [14]. It describes architectural elements on device level for resilience (resilience target, resilience engine, resilience authority), as well as building blocks as, e.g., storage protection and attention signal generators. Recommendations for resiliency of platform firmware and data have been described by [15], supporting a rapid and secure recovery from attacks on platform firmware of computer devices. Also, the standard ETSI EN303 645 on baseline security requirements for consumer IoT includes resilience-related requirements [23].

Segovia, Rubio-Hernan, Cavalli and Garcia-Alfarometrics define a metric based on control theory to quantify the cyber-resilience level of a CPS based on the design, structure, stability, and performance under attack [16]. The metric is related to the mathematically modelled control function of a CPS. Khazraei. Hallyburton, Gao, Wang and Pajic describe how deep learning can be applied for vulnerability analysis of CPS control mechanisms [17].

III. Industrial Cyber Physical Systems

A CPS, e.g., an industrial automation and control system, monitors and controls a technical system. Examples are process automation, machine control, energy automation, and cloud robotics. Figure 1 shows an example of an industrial automation and control system, comprising

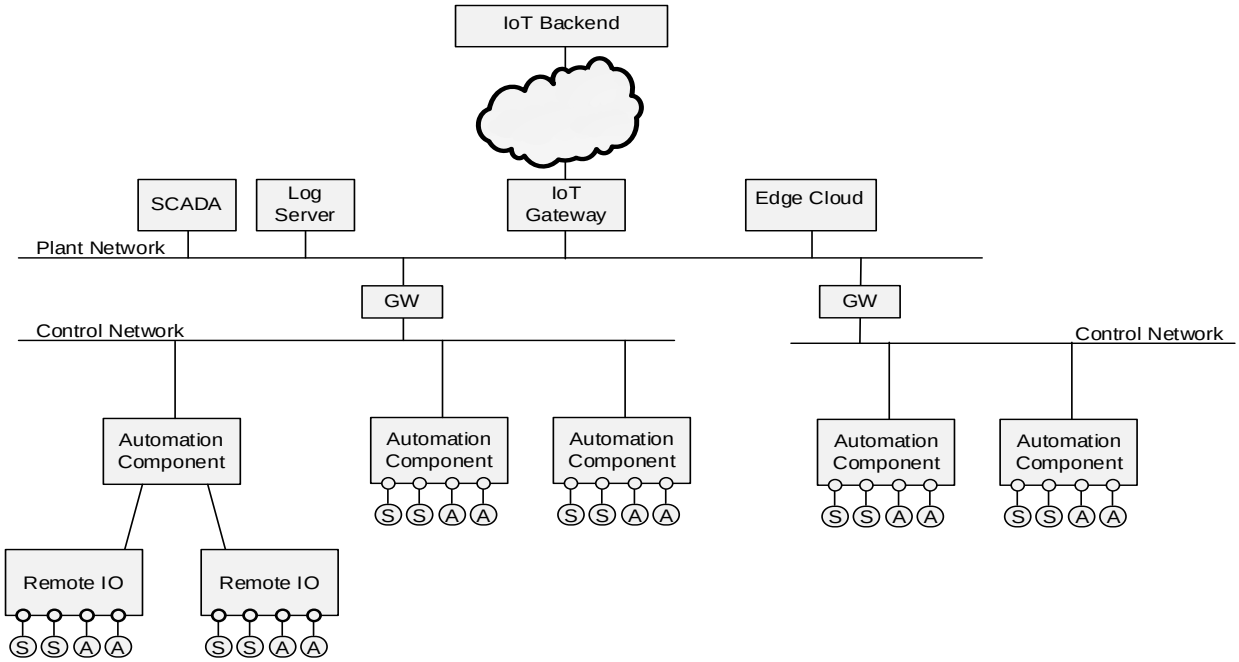


Figure 1. Example CPS System

different control networks connected to a plant network and a cloud backend system. Automation control equipment with sensors (S) and actuators (A) is connected directly with automation components, or via remote input/output modules. The technical process is controlled by measuring its current state using the sensors, and by determining the corresponding actuator signals. Separation of the network is typically used to realize distinct control networks with strict real-time requirements for the interaction between sensors and actuators of a production cell, or to enforce a specific security policy within a production cell. Such an industrial automation and control system is an example of a CPS. Industrial automation and control systems are utilized in various automation domains, including discrete automation (factory automation), process automation, railway automation, energy automation, and building automation.

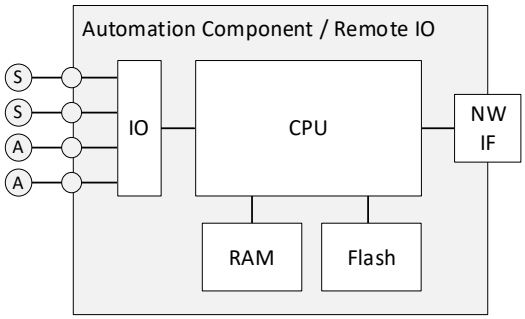


Figure 2. Automation Component

Figure 2 shows the typical structure of automation components of a CPS that monitor and control the physical world using sensors (S) and actuators (A). The monitoring and control functionality is defined by its firmware/software that is executed on a central processing unit (CPU) and the corresponding configuration data, both stored in non-volatile memory (Flash). A network interface (NW IF) allows communication with other devices, e.g., via Ethernet or via wireless communications as wireless local area network (WLAN) or a private 5th generation (5G) mobile communication system.

In cyber physical systems, the impact of a vulnerability in the OT system may not only affect data and data processing as in classical IT, but it may have an effect also on the physical world. For example, production equipment could be damaged, or the physical process may operate outside the designed physical boundaries, so that the produced goods may not have the expected quality, or even safety-related requirements could be affected.

IV. CPS System Integrity protection

Information Technology (IT) security mechanisms have been known for many years and are applied in smart devices (Internet of Things, Cyber Physical Systems, industrial and energy automation systems, operation technology). Such mechanisms target source authentication, system and communication integrity, and confidentiality of data in transit or at rest. System integrity takes a broader approach where not only the integrity of individual components (device integrity) and of network communications are addressed, but where integrity shall be ensured at the overall system level of multiple interconnected devices.

A. Industrial Security

Protecting industrial automation and control systems against intentional attacks is increasingly demanded by operators to ensure a reliable operation, and also by regulation. The main relevant industrial security standard that describes security from a holistic view is IEC 62443 [2]. Security requirements defined by the industrial security standard IEC 62443 range from security processes during development and operation of devices and systems, personal and physical security, device security, network security, and application security, addressing the device manufacturer, the integrator as well as the operator of the industrial automation and control system.

Industrial security is also called Operation Technology (OT) security, to distinguish it from general IT security. Industrial systems have different security priorities and requirements compared to common IT systems. Typically, availability and integrity of an automation system have higher priority than confidentiality.

Specific requirements and side conditions of industrial automation systems like high availability, planned configuration (engineering info), scheduled maintenance windows, long life cycles, unattended operation, real-time operation, and communication, as well as safety requirements have to be considered when designing an OT security solution.

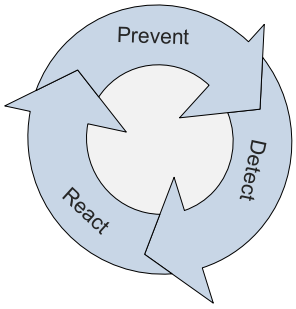


Figure 3. Prevent Detect React Cycle

Overall, security has to address the areas prevent, detect, and react, see Figure 3. It is not sufficient to only define security measures to protect against attacks. The cycle shows also the need for detecting attacks, and to define measures to react adequately once an attack has been detected. The approach describes in this paper puts more effort on the "detect" and "react" phases than on the "prevent" phase with the intention to supported increased CPS productivity by allowing for high flexibility of CPS reconfigurations.

B. Device Integrity

The objective of device integrity is to ensure that a single device is not manipulated in an unauthorized way, ensuring that it operates as genuine device. Integrity protection includes the integrity of the device firmware, the integrity of

the device configuration, but also its physical integrity. The main technologies to protect device integrity are:

- Secure boot: A device loads at start-up only unmodified, authorized firmware.
- Measured boot: The loaded software modules are checked at the time they are loaded. Usually, a cryptographic hash value is recorded in a platform configuration register of a hardware of firmware Trusted Platform Module (TPM). The configuration information can be used to grant access to keys, or it can be attested towards third parties.
- Protected firmware update: When the firmware of a device is updated, the integrity and authenticity of the firmware update is checked. The firmware update image can be digitally signed.
- Application whitelisting: Only allowed, known applications can be started on a device. A whitelist defines which application binaries can be started.
- Runtime integrity checks: During operation, the device performs a self-test of security functionality and integrity checks to verify whether it is operating as expected. Integrity checks can verify the integrity of files, configuration data, software modules, and runtime data as the process list, i.e., the list of currently executed processes.
- Process isolation, kernel-based Mandatory Access Control (MAC): Hypervisors or kernel-based MAC systems can be used to isolate different classes of software (security domains). An attack or malfunction of one security domain does not affect other security domains on the same device.
- Tamper evidence, tamper protection: The physical integrity of a device can be protected, e.g., by security seals or by tamper sensors that detect opening or manipulation of the housing.
- Device integrity self-test: A device performs a self-test to detect failures. The self-test is performed typically during startup and is repeated regularly during operation.
- Operation integrity checks: measurements on the device can be compared with the expected behavior in the operative environment. An example is the measurement of connection attempts to/from the device, based on parameters of a Management Information Base (MIB).

The established approaches to protect device integrity focus on its IT-related functionality of a device. The main protection objective for device integrity is to ensure that the device's control functionality operates as designed. However, the integrity of input/output interfaces, sensors, and actuators are typically out of scope. In typical industrial environments, applying a strong tamper protection to each control device, sensor, and actuator would not be economically feasible. A strong physical tamper protection is not common at device level, as it would complicate not only the production of a devices, but also the test, service, and repair. Therefore, protecting device integrity of used devices alone would be too limited to achieve the goal of protection the integrity of an overall CPS.

C. Cyber Physical System Integrity Monitoring

Classical approaches for protecting device and system integrity target at preventing any changes and compare the current configuration to a fixed reference policy. More flexible approaches are needed to protect integrity for flexibly reconfigurable and self-adapting CPSs. In previous work [4], we described an integrated, holistic approach for ensuring CPS integrity as an extensible framework to include integrity information from IT-based functions and the physical world of a CPS. This allows integrating integrity information from the digital and the physical world. Trusted physical integrity sensors can be installed as add-on to existing automation and control systems. One-way gateways can be used to extract integrity monitoring information from closed control networks, while ensuring freedom from interference for the control function.

Integrity does not only affect single devices, but also the overall system level comprising a set of interconnected devices. The main approaches to protect system integrity are collecting and analyzing information at system level [4]:

- Device inventory: Complete and up-to-date list of installed devices (including manufacturer, model, serial number version, firmware version, current configuration, installed software components, location)
- Centralized Logging: Devices provide log data, e.g., using Open Platform Communication Unified Architecture (OPC UA) protocol, Simple Network Management Protocol (SNMP), or syslog protocol, to a centralized logging system for further analysis. This may be done in a Security Information and Event Management (SIEM) System and lead to reactions on identified cybersecurity events.
- Runtime device integrity measurements: A device integrity agent provides information gathered during the operation of the device (see also subsection B above). It collects integrity information on the device and provides it for further analysis. Basic integrity information includes the results of a device self-test, and information on the current device configuration (firmware version, patches, installed applications, configuration). Furthermore, runtime information can be gathered and provided for analysis (e.g., process list, file system integrity check values, partial copy of memory).
- Network monitoring: The network communication is intercepted, e.g., using a network tap or a mirror port of a network switch. A challenge is the fact that network communication is increasingly encrypted.
- Physical Automation process monitoring: Trusted sensors provide information on the physical world that can be used to cross-check the view of the control

system on the physical world. Adding trusted sensors to existing installation allows for a smooth migration from legacy systems to systems providing integrated sensors as they can be used for plausibility checks.

– Physical world integrity: Trusted sensors (of physical world), integrated monitoring of embedded devices and IT-based control systems, and of the technical process allow now quality of integrity monitoring as physical world and IT world are checked together.

The captured integrity information can be used for system runtime integrity monitoring to detect integrity violations in real-time. Operators can be informed, or actions can be triggered automatically. Furthermore, the information is archived for later investigations. This allows that integrity violations can be detected also later with a high probability, so that corresponding countermeasures can be initiated (e.g., plan for an additional quality check of produced goods). The integrity information can be integrated in or linked to data of a production management system, so that it can be investigated under which integrity conditions certain production steps have been performed. Product data is enhanced with integrity monitoring data related to the production of the product. Moreover, the data may also be used in the context of supply chain security to support trustworthiness claims.

An intelligent analysis platform performs data analysis (e.g., statistical analysis, big data analysis, artificial intelligence) and triggers suitable respondence actions (e.g., alarm, remote wipe of a device, revocation of a device, stop of a production site, planning for additional test of manufactured goods). The analysis can combine monitoring information originating from IT-related control functions, from physical security systems, as well as from the operation of the actual technical process.

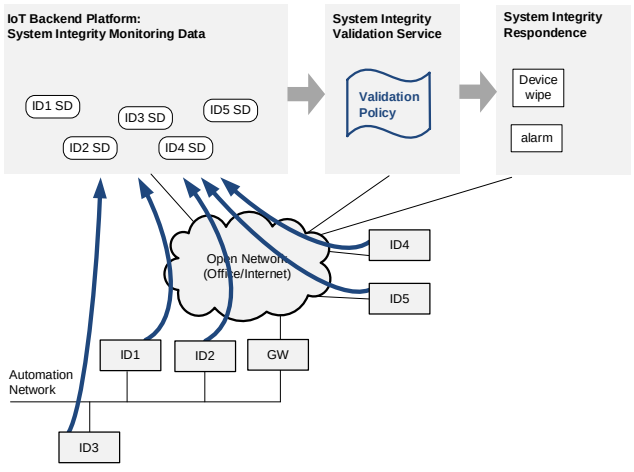


Figure 4. CPS Integrity Monitoring System [4]

Figure 4 shows an example for an IoT system with IoT devices (ID1, ID2, etc.) that communicate with an IoT backend platform. The devices provide current integrity monitoring information to the backend platform. The devices can be automation devices that include integrity measurement functionality, or dedicated integrity sensor devices. The device monitoring system itself has to be protected against attacks, following the industrial security standard IEC 62443.

An integrity data validation service checks the obtained integrity measurement data for validity using a configurable validation policy. If a policy violation is detected, a corrective action is triggered. For example, an alarm message can be displayed on a dashboard. Furthermore, an alarm message can be sent to the IoT backend platform to terminate the communication session of the affected IoT device. Moreover, the device security service can be informed so that it can revoke the devices access permissions or revoke the device authentication credential.

The integrity monitoring events are analyzed using known data analysis tools. As stated before, in industrial environments, it is also important to have reliable information about the system integrity of a production system for the time period during which a certain production batch was performed. This allows performing the verification also afterwards to check whether during a past production batch integrity-violations occurred.

The final decision whether a certain configuration is accepted as correct is up to human operators. After reconfiguration, or for a production step, the configuration is to be approved. The approval decision can be automated according to previously accepted decisions, or preconfigured good configurations.

D. Resilience Under Attack

Being resilient means to be able to withstand or recover quickly from difficult conditions [18]. It shifts the focus of "classical" IT and OT security, which put the focus on preventing, detecting, and reacting to cyber-security attacks, to the aspect to continue to deliver an intended outcome despite an adverse cyber attack taking place, and to recover quickly back to regular operation. More specifically, resilience of a system is the property to be resistant to a range of threats and withstand the effects of a partial loss of capability, and to recover and resume its provision of service with the minimum reasonable loss of performance [19].

Risk management, the established approach to cyber security, identifies threats and determines the risk depending on probability and impact of a potential attack. The objective is to put the focus of defined security measures on the most relevant risks, reducing the probability that a successful attack takes place, and reducing the impact of successful attacks, e.g., by detect successful attacks by security monitoring allowing to react, e.g., by shutting down a CPS. Resilience, however, puts the focus on a reduction of the impact of successful attacks, where the system can stay operational with a degraded performance or functionality, and to recover quickly from a successful attack. Robustness is a further related approach that tries to keep the system operational *without* a reduction of the system performance, i.e., to withstand attacks.

Figure 5 illustrates the concept of cyber resilience: Even if an attack is carried out, the impact on the system operation, i.e., the performance or functionality of the

system, is limited [5]. The effects of an attack are "absorbed", so that the system stays operational, but with limited performance or functionality. A recovery takes place to bring the system up to the regular operation.

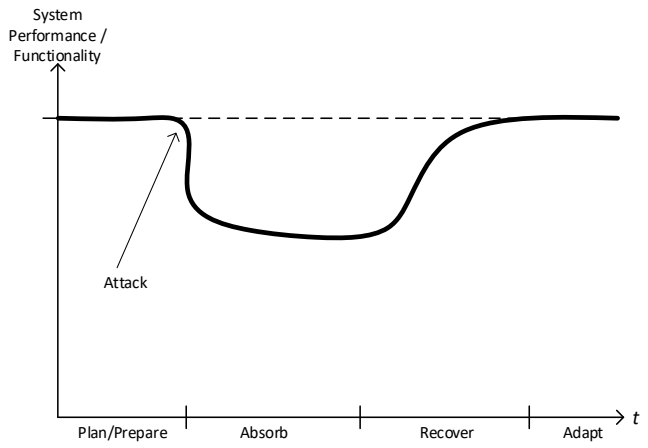


Figure 5. Concept of Cyber Resilience [5]

In adaptation of resilience, the system might be enhanced to better prepare for future attacks leading to a sort of self-healing functionality. In a cyber physical environment, a main objective is that the CPS stays operational and that its integrity is ensured. In the context of an industrial automation and control system, that means that intended actions of the system in the physical world continue to take place even when the automation and control system of the CPS should be attacked successfully.

V. LIFECYCLE CONFIGURATION CHANGE MONITORING

A main concept presented in this paper is an enhancement to the system-level integrity monitoring system, described in Section IV.C. Instead of comparing integrity measurements describing the current configuration status to a fixed reference policy, the changes to the CPS components and to their configuration are validated during CPS operation. An integrity violation is detected if changes are detected that are not in-line with a policy on what and how changes are applied and when. The changing configuration of CPS components along their lifecycle in the operation of a dynamically evolving CPS is validated to determine whether the CPS is in a trustworthy, authorized state (CPS system integrity).

Lifecycle state agents on the CPS components act as integrity sensors that collect lifecycle state information of a device and provide it in the form of a lifecycle state attestation to the system integrity monitoring system.

Figure 6 shows the basic concept of a CPS lifecycle-change integrity monitoring system. Devices (D) provide Life Cycle State Attestations (LCSA) to a CPS lifecycle-change integrity monitoring system. The CPS lifecycle-change integrity monitoring system determines changes on device lifecycle states based on the provided LCSA attestations, and it validates whether the detected changes are in-line with a lifecycle change validation policy.

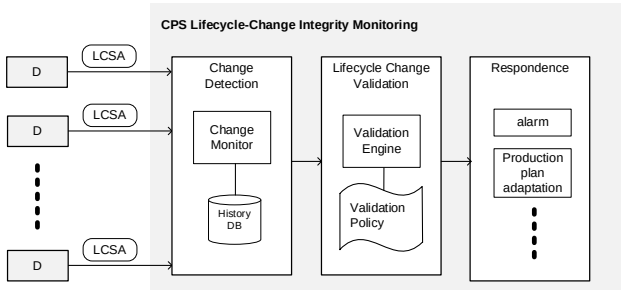


Figure 6. CPS Lifecycle Change Monitoring

The lifecycle change validation policy defines which changes are permitted so that the CPS is considered to be still in a trustworthy configuration state. If the lifecycle change validation policy is violated, e.g., an alarm can be generated, or the CPS operation of the production plan can be adapted accordingly.

VI. DEVICE LIFECYCLE STATE ATTESTATION

Different lifecycle states of industrial IoT devices can be distinguished, including factory default state, commissioned, operational, failure, network connected, provisioned, repair, service, or being put out of service. The current lifecycle state of a device can be determined based on its current configuration data. Some security standards, e.g., ETSI EN 303645 on Consumer IoT Security includes an example of a device life cycle model [23]. Besides the life cycle phase information, also the parts of the specific configuration can be provided as part of the life cycle attestation and analyzed. It is not assumed that a common life-cycle model is explicitly supported by the devices, as in a real-world CPS, different device types originating from various manufacturers are used. Instead, the available information of the device configuration is taken as basis to derive/estimate the related life-cycle phase, at least if it is not provided explicitly.

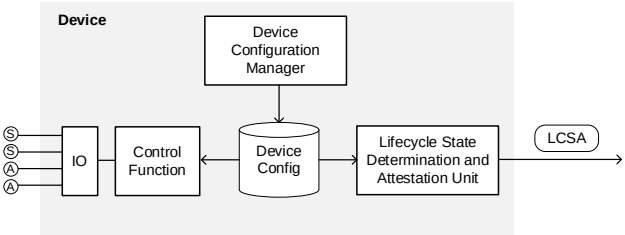


Figure 7. Control Device with Lifecycle State Attestation

A device can determine its own lifecycle state and confirm it externally by a device lifecycle state attestation. Figure 7 shows a device, e.g., a control device for monitoring and controlling a technical process via sensors (S) and actuators (A) by a control function that interacts via an input-output unit (IO) with the sensors and actuators, according to the device configuration established by a device configuration manager. The lifecycle state attestation unit determines the device lifecycle state based on the current

device configuration and creates a cryptographically protected LCSA. Besides the current lifecycle state, also previous lifecycle states can be kept and attested, providing a more comprehensive information on the device lifecycle history. Alternatively, the lifecycle state may be determined and attested by an external add-on component, allowing that a LCSA can be provided also for legacy devices that do not have an integrated functionality for determining and attesting the device lifecycle state.

The LCSA can be provided in a dedicated attestation data structure, i.e., a data structure that describes the current lifecycle state of the device, and that is protected by a cryptographic checksum, i.e., a digital signature or a message authentication code. However, it is also possible to encode the life cycle information in a device credential, e.g., a device authentication certificate, a device attribute certificate, a device authentication token, or a verifiable credential.

VII. Device Lifecycle State Analysis

A simple approach for validating CPS configuration changes would be the manual analysis of detected configuration changes and a manual approval of detected changes by OT personnel. Manual checking and approval would however not scale well for larger CPS that are frequently reconfigured. Therefore, an automatic validation of detected configuration changes is needed. The CPS Lifecycle-change Integrity Monitoring system determines the changes to the CPS configuration based on the obtained device lifecycle state attestations. It validates whether changes are in-line with a lifecycle change validation policy that defines the permitted types of changes to the CPS configuration. If the lifecycle change validation policy is violated, e.g., an alarm can be generated, or the CPS operation or a production plan can be adapted accordingly.

The lifecycle change validation policy defining permitted changes of lifecycle states can be preconfigured. However, this would require significant effort for explicitly defining rules for permitted configuration changes. Therefore, an automated learning system, based on artificial intelligence, is proposed that learns from good examples of permitted changes. In an initial introduction phase, good changes (allowed changes from a system operation level) have to be marked by the OT personnel. Over time, the system learns from these good examples. This approach is conceptually similar to a network firewall for which the filter policy is determined automatically during a learning phase.

Such a self-learning of permitted changes leads to an automated learning of what changes lead to a trustworthy CPS. It is in real-world practice often not easy to determine explicit rules on which specific properties make a component or a change being considered as trustworthy. By learning from good and bad examples, the attributes that are relevant for the trustworthiness evaluation can also be determined over time automatically. The system learns which attributes of a lifecycle state attestation are relevant for determining which changes are permitted. This self-learning approach allows also for subjective trust policies: Different users, i.e., operators of similar CPSs, can give examples of what they consider to be trustworthy or not so trustworthy. Depending on these examples, a trustworthiness evaluation policy is derived. In contrast to conceptually similar approaches like the example of firewalls in learning mode, this approach is more open as even the attributes (criteria) that are relevant for making trust decisions do not have to be predefined. It allows also to distinguish varying operational concepts for CPSs that are operated by different OT operators.

VIII. Evaluation

From the perspective of a real-world CPS, the approach presented in Sections V, VI and VII is not self-contained, but is an extension to other, well-established security measures to protect a CPS. The main advantage comes by the support for increasingly dynamic, evolving CPS. To ensure that a CPS and its components are in a trustworthy state, it is not ensured that the configuration corresponds to a fixed reference, but to check whether the detected changes are acceptable. This approach can compensate when classical, rather strict security controls preventing heavy changes to a CPS cannot be applied anymore in the same way as for static CPS deployments.

The security of a cyber system can be evaluated in practice in various approaches and stages of the system's lifecycle:

- Threat and Risk Analysis (TRA) of cyber system
- Checks during operation to determine key performance indicators (e.g., check for compliance of device configurations).
- Security testing (penetration testing)

During the design phase of a cyber system, the security demand is determined, and the appropriateness of a security design is validated using a TRA. Assets to be protected and possible threats are identified, and the risk is evaluated in a qualitative way depending on probability and impact of threats. The effectiveness of the proposed enhanced device authentication means can be reflected in a system TRA.

The main evaluation using security tools is performed during secure operation, when as part of an overall operational security management appropriate technologies are deployed that, in combination, reduce the risk to an acceptable level. The new approach presented in this paper provides an additional element, integrated into the overall system security architecture that is used to reduce the risk of integrity violations, despite a dynamically changing CPS configuration.

For the applicability to real-world CPS environments, the approach allows for:

- Flexibility for updates: The device life cycle integrity monitoring system can be updated independently from the actual CPS. Therefore, updates can be installed also outside the scheduled maintenance windows of the CPS.
- It can be installed as add-on to existing automation systems (brownfield). It can be introduced stepwise, starting with lifecycle monitoring for most relevant devices.

– It can be installed as an add-on system that does not endanger the reliable operation of a CPS or invalidate its certifications.

Such non-technical properties simplify the adoption in real-world CPS, and they are often important factors for acceptance by OT operators.

As long as the technology proposed in the paper has not been proven in a real-world operational setting, it can be evaluated conceptually by analyzing the impact that the additional security measure would have on the identified residual risks as determined by a TRA, and on key performance indicators (KPI) of automation and production systems like uptime, availability, output. Actually, the approach of evaluation the impact of different approaches to handle security on KPIs that are not directly security-related is typically not done systematically in the security community. The motivation of the lifecycle security monitoring intends to give high flexibility to reconfigure industrial CPS to changing needs, while still ensuring the required level of security.

It is also an open point how to balance security controls addressing different phases (prevent, detect, react) in an optimized way. The approach described in this paper puts less emphasis on restrictive security measures on the "protect" phase but rather intends to compensate that by automated monitoring of configuration changes ("detect") and to use the high flexibility for CPS reconfiguration to flexibly react also to detected security problems ("react"), improving thereby also the resiliency. Putting these considerations in the context of a TRA, means shifting the focus for reducing identified risks to an acceptable level from reducing the likelihood of a threat occurrence ("prevent") to reducing its impact ("detect" and "react").

Threat	Likelihood	Impact	Risk
Device communication intercepted	unlikely	moderate	minor
Device communication manipulated	unlikely	critical	moderate
Vulnerability in unpatched device exploited	likely	critical	major
Device replaced by fake device	possible	moderate	moderate
⋮	⋮	⋮	⋮

Figure 8. Example Threats of a Threat and Risk Analysis

Figure 8 shows a simplified table as used typically in a threat and risk analysis to collect and evaluate relevant treats to a technical system or component. Some threats are shown as examples. Actual TRAs for real-world systems and components include usually a much longer list of threats. The likelihood and the impact of the threat is determined by judgement of competent personal, usually in a team including technical experts, developers, and people responsible for the product or system. The corresponding risk is determined based on likelihood and impact. It has shown to be useful to define and document explicitly the criteria leading to the categorization of likelihood and impact, including also the assumptions made on the operational environment. The TRA with prioritized risks is the basis for security design decisions, focusing on the most critical risks. It is the basis to define a security concept that defines suitable measures for reducing the risk to an acceptable level.

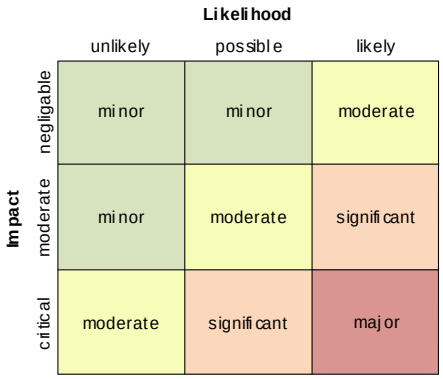


Figure 9. Risk Mapping

Figure 9 shows how the mapping of likelihood and impact to the corresponding risk value. In the example, the three categories unlikely, possible, and likely are used to describe the likelihood. For the impact, the three categories negligible, moderate, and critical are used. In practice, also more fine-granular rankings can be used, distinguishing, e.g., four or five different categories. Also, the risk evaluation can in general include further categories, e.g., disastrous. It can be seen that a reduction of the risk can be achieved by both, by reducing the likelihood as well as by reducing the impact.

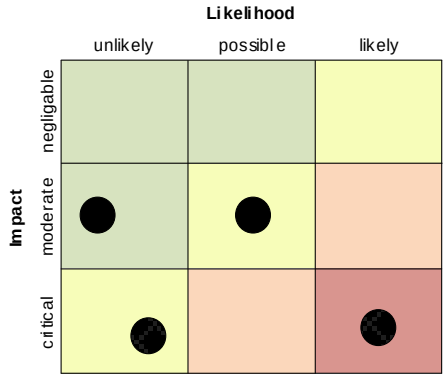


Figure 10. Risk Reporting for the Example Threats

An overview on the determined risks can be shown in a risk reporting as shown in Figure 10. It gives an easily understandable graphical representation on the distribution of risks. This representation can be useful if many risks have been identified. In particular, the example also shows one major threat as well as a moderate threat with critical impact.

The effect of reducing the risk by limiting the impact is illustrated in Figure 11. As, shown in the example, the impact of the two risks with critical impact reduces from critical to moderate, the risk is reduced correspondingly. Thereby, also the overall risk situation of the overall CPS in which the considered device is used, is improved.

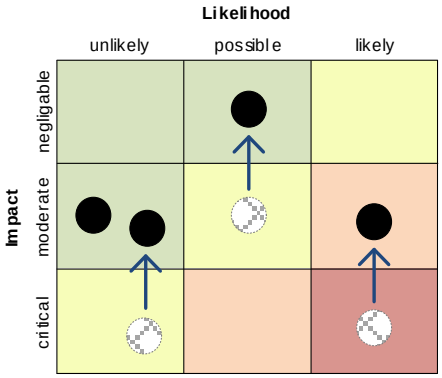


Figure 11. Reducing risk be limiting the impact

As the evaluation in a real-world CPS requires significant effort, and as attack scenarios cannot be tested that could really have a (severe) impact on the physical world, a simulation-based approach or using specific testbeds are possible approaches, allowing to simulate or evaluate in a protected testbed the effect on the physical world of certain attack scenarios with compromised components. The simulation would have to include not only the IT-based control function, but also the physical world impact of an attack. Using physical-world simulation and test beds to evaluate the impact of attacks have been described by Urbina, Giraldo et al. [24]. However, we are not aware of research work that analyzes systematically the impact of different security approaches on operational KPIs of a CPS, e.g., based on simulations or by analyzing data or operational real-world CPSs.

IX. CONCLUSION

Ensuring device and system integrity is an essential security feature for cyber physical systems and the (industrial) Internet of Things. This must be ensured from the beginning using the security design principle of "defense in depth". It allows to support system integrity based on the information provided from single components or devices that build the CPS.

This paper proposed a framework for ensuring system integrity in flexibly adaptable cyber physical systems. With new concepts for flexible automation systems coming with Industrial IoT / Industry 4.0, the focus of system integrity clearly has to move from preventing changes to device and system configuration to having transparency on the device and system configuration and checking it for compliance.

The approaches for integrity monitoring in industrial automation and control systems described in this paper focuses on the operational phase by relying on lifecycle attestations for single components building a CPS. This approach enhances the existing systems, with an attestation about a specific state in the lifecycle, which allows an industrial monitoring system to evaluate the current life cycle state with the expected one. This can be done in addition to classical system monitoring, which verifies configuration and system behavior against expected patterns.

Integrity in a broader sense has to cover the whole life cycle, from development, secure procurement, secure manufacturing, and supply chain security up to the commissioning phase in the operational environment. This lifecycle information can then be used to enhance the current system state information. Due to the life cycle information available on the device or its associated management system, feedback to manufacturer can be provided in case of failure, in which the problem may be traced back to a specific production step. This also allows the manufacturer to better react in future versions of a device. It also allows for informing other users of the same component or systems about potential failure scenarios or situations.

Security-critical operations of a device, e.g., use for control operations, provisioning operational keys, or providing sensitive commissioning data is performed only for devices being in an expected state. A device can be used for regular operational purposes only if, according to its lifecycle, it is in a valid lifecycle state, and if this lifecycle state has been established in a permitted way.

A main objective of the described approach is to support the increase of CPS productivity that is coming with the flexible production of industry 4.0, supporting "lot size 1". The described security approach supports a high flexibility of CPS reconfigurations while still ensuring an appropriate security level. Integrity of the CPS is not ensured by preventing changes to the CPS configuration, but by reliably determining performed configuration changes and by validating whether they are permitted.

Possible future research could analyze systematically the impact of different security approaches on operational KPIs as productivity, defective goods, or whether tight production schedules are met by simulating complete CPS systems under different usage situations and under different attack scenarios. A further approach may be the integration of simulation into existing production environments using a digital twin. This digital twin would then be operated under the same conditions as the physical devices with the option to virtually manipulate parameters of the operational environment to stipulate extreme cases and thus better prepare for timely reactions to potential real events. While such analysis is considered to require some effort, it could provide the bases to come up with security designs for complex CPS that optimize operational KPIs while still reliably ensuring the targeted level of security.

REFERENCES

[1] R. Falk and S. Fries, "Dynamic Trust Evaluation of Evolving Cyber Physical Systems", CYBER 2022, The Seventh International Conference on Cyber-Technologies and Cyber-Systems, pp.19-24, 2022, [Online]. Available from http://thinkmind.org/index.php?view=article&articleid=cyber_2022_1_30_80022 [retrieved January, 2023]

[2] IEC 62443, "Industrial Automation and Control System Security" (formerly ISA99), available from: http://isa99.isa.org/Documents/Forms/AllItems.aspx [retrieved January, 2023]

[3] European Commission, "Proposal for a directive of the European parliament and of the council on measures for a high common level of cybersecurity across the Union, repealing Directive (EU) 2016/1148", COM(2020) 823 final. 2020/0359(COD), Dec. 2020, [Online]. Available from https://eur-lex.europa.eu/legal-

content/EN/TXT/?uri=COM:2020: \ 823:FIN [retrieved January, 2023]

[4] R. Falk and S. Fries, “System Integrity Monitoring for Industrial Cyber Physical Systems”, International Journal On Advances in Security, volume 11, numbers 1&2, pp. 170-179, 2018, [Online]. Available from https://www.thinkmind.org/index.php?view=article&articleid=sec_v11_n12_2018_14 [retrieved January, 2023]

[5] R. Falk and S. Fries, “Enhancing the Resilience of Cyber-Physical Systems by Protecting the Physical-World Interface”, International Journal On Advances in Security, volume 13, numbers 1 and 2, pp. 54-65, 2020, [Online]. Available from: http://www.thinkmind.org/index.php?view=article&articleid=sec_v13_n12_2020_5 [retrieved January, 2023]

[6] Plattform Industrie 4.0, “Industrie 4.0 Plug-and-produce for adaptable factories: example use case definition, models, and implementation”, Plattform Industrie 4.0 working paper, June 2017, [Online]. Available from: https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/Industrie-40-Plug-and-Produce.pdf [retrieved January, 2023]

[7] ISO/IEC TS 5723:2022 “Trustworthiness – vocabulary”, July 2022. Available from: https://www.iso.org/standard/81608.html [retrieved January, 2023]

[8] N. G. Mohammadi, “Trustworthy cyber-physical systems: a systematic framework towards design and evaluation of trust and trustworthiness”, Springer, January 2019.

[9] Y. Jiang, “Vulnerability analysis of critical infrastructures”, PhD thesis, University of Skövde, 2022, [Online]. Available from: https://www.researchgate.net/profile/Yuning-Jiang-7/publication/363174252_PhD_Thesis_-_Vulnerability_Analysis_for_Critical_Infrastructures/links/631467bd61e4553b9564e7ff/PhD-Thesis-Vulnerability-Analysis-for-Critical-Infrastructures.pdf [retrieved January, 2023]

[10] B. Northern, T. Burks, M. Hatcher, M. Rogers, and D. Ulybyshev, “VERCASM-CPS: vulnerability analysis and cyber risk assessment for cyber-physical systems”, Information 2021, 12(10), 408, MDPI, 2021 [Online]. Available from: https://www.mdpi.com/2078-2489/12/10/408 [retrieved January, 2023]

[11] A. A. Malik and D. K. Tosh, “Dynamic risk assessment and analysis framework for large-scale cyber-physical systems”, SESA 22(30):1, EAI, 2022, [Online]. Available from: https://eudl.eu/doi/10.4108/eai.25-1-2022.172997 [retrieved January, 2023]

[12] M. Tapia, P. Thier, and S. Gößling-Reisemann, “Vulnerability and resilience of cyberphysical power systems – results from an empirical-based study”, artex paper 222, Univ. of Bremen, April 2020, [Online]. Available from: https://www.uni-bremen.de/fileadmin/user_upload/sites/artec/Publikationen/artec_Paper/222_paper.pdf [retrieved January, 2023]

[13] A. Akbarzadeh and S. K. Katsikas, “Dependency-based security risk assessment for cyber-physical systems”, International Journal of Information Security, Springer, August 2022, [Online]. Available from: https://link.springer.com/article/10.1007/s10207-022-00608-4 [retrieved January, 2023]

[14] Trusted Computing Group, “Cyber resilient module and building block requirements”, Version 1.0 Revision 0.2, June 2022, [Online]. Available from: https://trustedcomputinggroup.org/resource/cyber-resilient-module-and-building-block-requirements/ [retrieved January, 2023]

[15] A. R. Regenscheid, "Platform firmware resiliency guidelines", SP800-193, NIST, May 2018, [Online]. Available from: https://www.nist.gov/publications/platform-firmware-resiliency-guidelines [retrieved January, 2023]

[16] M. Segovia, J. Rubio-Hernan, A. R. Cavalli, and J. Garcia-Alfaro, "Cyber-resilience evaluation of cyber-physical systems," 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA), 2020, pp. 1-8. Available from https://ieeexplore.ieee.org/document/9306741 [retrieved January, 2023]

[17] A. Khazraei. S. Hallyburton, Q. Gao, Y. Wang, and M. Pajic, "Learning-based vulnerability analysis of cyber-physical systems", ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS), 2022. Available from: https://cpsl.pratt.duke.edu/sites/cpsl.pratt.duke.edu/files/docs/khazraei_iccps22.pdf [retrieved January, 2023]

[18] P. England, R. Aigner, A. Marochko, D. Mattoon, R. Spiger, and S. Thom, “Cyber resilient platforms”, Microsoft Technical Report MSR-TR-2017-40, Sep. 2017, [Online]. Available from: https://www.microsoft.com/en-us/research/publication/cyber-resilient-platforms-overview/ [retrieved January, 2023]

[19] Electronic Communications Resilience&Response Group, “EC-RRG resilience guidelines for providers of critical national telecommunications infrastructure”, version 0.7, March 2008, available from: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/62281/telecoms-ecrrg-resilience-guidelines.pdf [retrieved January, 2023]

[20] ISO/IEC 27001, “Information technology – Security techniques – Information security management systems – Requirements”, October 2013, available from: https://www.iso.org/standard/54534.html [retrieved January, 2023]

[21] IEC 62443-3-3:2013, “Industrial communication networks – Network and system security – Part 3-3: System security requirements and security levels”, Edition 1.0, August 2013. Available from: https://webstore.iec.ch/publication/7033 [retrieved January, 2023]

[22] IEC 62443-4.2:2019, “Security for industrial automation and control systems - Part 4-2: Technical security requirements for IACS components”, Feb. 2019. Available from: https://webstore.iec.ch/publication/34421 [retrieved January, 2023]

[23] EN 303 645, “Cyber Security for Consumer Internet of Things: Baseline Requirements”, ETSI, V2.1.1 (2020-06), June 2020. Available from: https://www.etsi.org/deliver/etsi_en/303600_303699/303645/02.01.01_60/en_303645v020101p.pdf [retrieved January, 2023]

[24] D. Urbina, J. Giraldo, N. O. Tippenhauer, and A. Cardenas, “Attacking fieldbus communications in ICS: applications to the SWaT testbed”, Singapore Cyber-Security Conference (SG-CRC), IOS press, pp. 75–89, 2016, [Online]. Available from: http://ebooks.iospress.nl/volumearticle/42054 [retrieved January, 2023]

Protected Establishment of a Secondary Network Access Channel

Steffen Fries, Rainer Falk
Corporate Technology
Siemens AG
Munich, Germany
e-mail: {steffen.fries|rainer.falk}@siemens.com

***Abstract*—Several use cases demand for the setup of a separate, dedicated communication channel that provides a specific quality of service or that separates communications of different criticality. Different properties of communication channels are, for instance, performance, latency, but may be also security related. In several cases, a reliable association to an already established communication channel is required. Specifically, if a first communication channel has been securely established, a cryptographic binding of a second communication channel to this first communication channel is needed. One example use case is the charging of electric vehicles. Besides the charging control, also value-added services like software updates for the infotainment system or other parts of the electric vehicle or entertainment services may be provided. To avoid interfering with the charging-related control communications, a second, separate communication channel is established. The two communication channels may require different quality of service. The cryptographic binding allows to perform authorization checks to access value-added services and maybe also to associate the billing of consumed value-added services to the user that has been authenticated in the setup of the first communication channel. The paper provides an overview about existing solutions and proposes an alternative solution that allows establishing arbitrary communication channels of different nature and on different communication layers of the OSI protocol stack. The main example used is the interaction between an electric vehicle and a charging station, but the proposed solution is open to different applications.**

Keywords–communication security; cryptographic channel binding; quality of service; industrial automation and control system; Internet of Things.

I. INTRODUCTION

In network communications, it is typically required to have distinct relations between communicating endpoints, which are defined by several parameters, like the addresses of the communicating endpoints, security credentials connected with the endpoints, but also by certain quality-of-service related features. Quality-of-service (QoS) features may relate to a specific throughput expected by the communication channel or a specific response time or latency of the communication, but also to specific security properties of the communication like integrity protection or combined integrity and confidentiality protection. These properties may be provided on different levels of the Open System Interconnection (OSI) protocol stack, like the utilized transport protocol or application protocol. Another option is to verify and enforce required properties already during network access. Network access may be achieved using different communication technologies such as wired access using a classic cable installation, but also using wireless access via wireless LAN (WLAN), 4G, or 5G mobile communications.

Specific QoS features are required by a variety of applications. In [1], the emphasis was placed on electric vehicle charging as prominent example. Further applications with specific QoS requirements are also known from real-time applications like real-time control in industrial automation, voice-and-video conferences, or video streaming. Specific security applications may leverage a separate communication channel like the provisioning of credentials using a connection with limited access to the operational service providing network. If the setup of a communication channel with certain QoS features is based on a previously established communication relation, a binding of the two communication sessions can be leveraged in multiple ways.

The aim of this paper, as extended version of [1], is to further elaborate the setup of a new secure secondary communication channel that utilizes properties of a previously established communication channel. Specifically, it will provide further insights into the electric vehicle charging use case and the correlation of charging related communication and value-added related communication. As outlined, value-added services may relate to updates of the firmware, software, or map material for the infotainment system of an electric vehicle.

This paper is structured in the following way. Section II provides an enhanced overview about electric vehicle charging as a potential target scenario. Section III investigates existing approaches to provide distinct communication channels with distinct properties. Section IV describes a new approach, and section V analyzes its advantages. Section VI concludes the paper and provides and outlook to future work.

II. ELECTRIC VEHICLE CHARGING

The number of electric vehicles as bicycles, motorcycles, and cars has increased in the recent years significantly. They are connected to the Digital Smart Grid for charging. Besides basic charging, also the development of bidirectional charging

is ongoing, which allows to utilize electric vehicles as energy storage system and to feedback energy to the power grid when necessary. Depending on the charging interface between the electric vehicle and the infrastructure, the charging may be accomplished within minutes, or it may need up to several hours. While connected to a charging station, the vehicle exchanges constantly control data with the charging station to provide data like locally measured energy consumption on the vehicle side or charging commands with parameter adaptations from the charging station. This connection time may also be used to provide value-added services by utilizing the connection already established between the electric vehicle and the charging station.

As depicted in Figure 1, a multitude of potential communication options exists involving different actors within the charging system. The communication channel established between the electric vehicle and the charging station may be setup using different standards like ISO/IEC 15118 [2] or CHaDemo [3]. The focus in this paper is placed on ISO/IEC 15118.

The communication is accomplished using power line communication when the vehicle is connected via a wired interface. Alternatively, a wireless interface like WLAN is typically employed when inductive charging is performed. In this case, the charging station provides a WLAN access point to facilitate the communication. According to ISO/IEC 15118, access to the charging station is not protected on the WLAN access layer, but on higher communication layers. This avoids a specific WLAN access configuration of electrical vehicles for a specific charging station. The communication performed in the context of ISO/IEC 15118 allows to provide charging parameter information, billing relevant information. To do this in a secure manner, mutual authentication of the electric vehicle and the charging station can be performed at the beginning of a charging session. The security of ISO/IEC 15118-2 has been studied from the early beginning of standardization (cf. for example [4]) of the vehicle to grid interface. Meanwhile, the standard has been completed, and a revision will be published soon as Edition 2.

The communication channel between the electric vehicle and the charging station is part of the bigger Digital Grid picture as shown in Figure 1. Besides the pure charging relevant communication. Value-added service providers may utilize the communication channel as well but may be independent of the charging point operator.

The energy distribution network as critical infrastructure relies on the availability of the information infrastructure. Therefore, the information infrastructure must be managed and operated according to the same level of reliability as required for the stability of the power system infrastructure to prevent any type of outage or disturbance. The immediately apparent security needs target the reliable operation of the power grid and prevention of financial fraud. Especially the interaction between new market participants and value-added services has been investigated and is also addressed in ISO/IEC 15118.

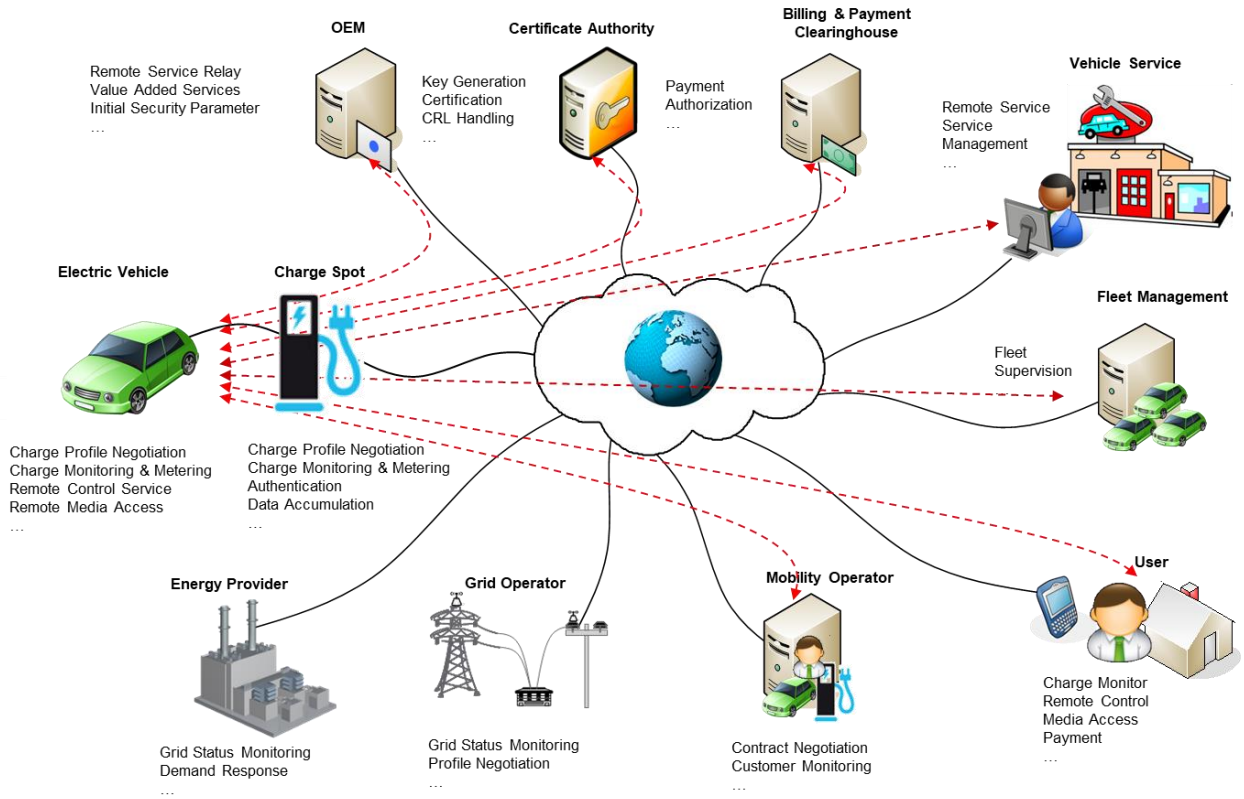


Figure 1. Electric Vehicle Communication Connections

Common to both editions, 1 and 2, of the standard ISO/IEC 15118 is the security approach and specifically the security setup between the electric vehicle and the charging infrastructure. It relies on the establishment of a secured communication channel based on Transport Layer Security (TLS). TLS, version 1.2 specified in IETF RFC 5246 [5] is used in Edition 1 of ISO/IEC 15118-2, while TLS version 1.3 specified as IETF RFC 8446 [6] is used in edition 2. During the communication establishment, it requires that the charging station authenticates towards the electric vehicle using an X.509 certificate as part of the TLS handshake. In turn, if the electric vehicle uses plug-and-charge, or if it wants to consume value-added services, it authenticates with an own X.509 certificate that is bound to the charging contract. This charging contract has been established between the electric vehicle owner and the mobility operator of his choice. It allows for a seamless charging experience for the vehicle owner, similar to roaming in the mobile communication domain. In addition, it allows to access value-added services after connecting to the charging station.

The value-added service communication is performed separately from the control and measurement communication channel. This is to avoid any interference with the charging related control communication. ISO/IEC 15118 facilitates this by establishing a separate communication channel that is bound to the initial authentication of both peers and outlined in section III.D below.

The following section Investigates different options of providing an authenticated channel that is bound to a mutual authentication between the electric vehicle and the charging station.

III. Existing Approaches For Communication Channel Protection and Channel Binding

Communication channel protection can be performed in different ways. The probably simplest way is the usage of a virtual private network (VPN) to protect arbitrary higher layer communication protocols in a transparent way. Transparent means here that the protected application (layer protocol) is not affected by the VPN. This may be achieved by approaches like a cryptographic VPN (see subsection III.A), SOCKS (see sub section III.B), or a virtual local area network (VLAN) (see subsection III.C). In contrast, an application may also be aware of the underlying security channel and signal it to the user as in case of, for instance, browser-based communication using the Hypertext Transfer Protocol (HTTP) over TLS resulting in HTTPS. The “S” indicates the establishment of a secure tunnel towards the user. The approach in TLS is described in subsections III.D and III.E. The latter also provides insights into the cryptographic binding of TLS sessions or TLS channels.

The concept of cryptographic channel binding is described in IETF RFC 5056 [7] and relates to the binding of a lower layer communication to a higher layer communication. In the context of this document, channel binding is also used to refer to a binding between two network access sessions to ensure that they involve the same peers and also that they have been established in a specific order.

The following subsections investigate into approaches for setting up a communication channel, which can be bound to another communication channel supporting certain security properties like the authentication of a single peer or of both peers.

A. Cryptographic Virtual Private Networks–- VPN

A cryptographic virtual private network is a communication connection that can be setup between a client (endpoint) and a service providing network, or between two networks. To ensure security properties like peer authentication, communication integrity and confidentiality (to ensure privacy of the communication), cryptographic protocols are used to setup and protect this connection.

Cryptographic VPNs may be built on different layers of the OSI protocol stack. Typical are OSI Layer 3 VPNs. A prominent protocol supporting the establishment of such a VPN is the IP Security Protocol (IPSec, IETF RFC 4301 [9]. Alternatives are for instance OpenVPN [10] or the much younger WireGuard [11]. The latter bases on the noise protocol framework [12] and focuses on simplifications in the configuration and in a more restrictive definition of utilized cryptographic algorithms to offer a streamlined protection.

Cryptographic VPNs may also be established on OSI layer 2 and support the security of Virtual LANs (see subsection III.C below). Moreover, they may also use OSI layer 4 by utilizing TLS as security protocol. They are referred to as SSL VPNs or TLS-VPNs. In contrast to IPSec, the cryptographic protection is established per application connection and not resulting in a network coupling as in the case of IPSec.

B. Socket Secure – SOCKS

SOCKS [13] is an internet protocol that allows applications (client or server) to connect through proxies in an application layer independent way. This is done by using a SOCKS proxy that creates a TCP connection to the target server on behalf of the client. As SOCKS operates on layer 5, it can handle different application protocols like HTTP, the Simple Mail Transfer Protocol (SMTP), or the File Transfer Protocol (FTP). It allows a client to open a connection from behind a firewall to an external server in an authenticated and authorized way. SOCKS5 allows for different authentication methods, in which the client authenticates towards the SOCKS server. It may also be used in conjunction with TLS. After authentication and authorization check by the SOCKS server, the application protocol is tunneled over the established connection and forwarded to the external target server.

The authentication is done between the requesting client and the SOCKS server, and the tunneling of the application protocol binds to this authentication. However, the server is not aware of this authentication and needs to authenticate the client by other means. As the tunnel is provided on an application base, multiple tunnels for different applications are necessary, all with an own, independent security setup.

C. Virtual LAN – VLAN

VLAN or virtual local area networks are defined in IEEE 802.1Q [14]. The standard defines a logical network and

allows the separation of different communication channels on layer 2. Different properties may be assigned in addition to this virtual LAN like performance or throughput. To achieve this, infrastructure components like managed switches are used, supporting the differentiation of traffic according to VLANs. A peer sending information in this VLAN (unicast or multicast) will only reach other peers that are part of the same VLAN.

Two basic approaches exist for VLANs. The first approach is a port-based VLAN in which the association to a logical LAN is done by attaching the client to a dedicated physical port of a managed switch. The second approach is a tagged VLAN, in which the Ethernet frames are tagged with a specific VLAN identifier (VLAN ID). Based on this VLAN tag, a switch can forward the Ethernet frame according to its configuration.

With this, VLANs themselves provide a way to separate traffic, which is also a step towards improved security. The definition of this separation is not done on cryptographic means, as stated before. Therefore, it is recommended to provide additional protection of the communication. Examples are IEEE 802.1X [15], providing port-based access control. With this, a client authenticates to the infrastructure (typically a RADIUS or DIAMETER server) via the infrastructure access network switch using different means, e.g., based on the Extensible Authentication Protocol (EAP) [16]. EAP allows for authentication with username and password, but also for a certificate-based authentication employing a client's X.509 certificate. In addition, MAC security (MACSec), specified in IEEE 802.1AE [17], can be used to provide integrity and/or confidentiality protection for the traffic between the device and the network switch in a hop-by-hop fashion.

Security for VLAN can be provided using additional security means like IEEE 802.1X as outlined on OSI Layer 2. Another prominent protocol for securing VLANs is IPSec as described in subsection III.A. If associated to a dedicated VLAN, quality of service parameter may be assigned.

D. Transport Layer Security Features

Transport Layer Security (TLS) is a protocol defined in IETF RFC 5246 as version 1.2 [5]. Meanwhile, it evolved to version 1.3 in IETF RFC 8446 [6]. While version 1.3 is being increasingly adopted [21], version 1.2 is still widely used. TLS is probably the most commonly used security protocol to protect TCP-based communications. The most prominent application is the protection of web-based communication over http. Also, other TCP-based protocols leverage the bump in the wire properties of TLS, like ISO/IEC 15118. ISO/IEC 15118-20 mandates the support of TLS v1.3, while TLSv1.2 may still be used.

TLSv1.3 features a re-designed handshake, which is not backward compatible to TLSv1.2. The version handling in TLS allows to fall back to TLSv1.2, if TLSv1.3 is not supported yet. The handshake is encrypted, except for the very first message, to better protect the privacy of client certificate information that is thereby already send encrypted. Moreover, the handshake may already transmit application data, which can accelerate the communication setup. This feature is called 0-RTT (zero round-trip time), but the use requires careful review.

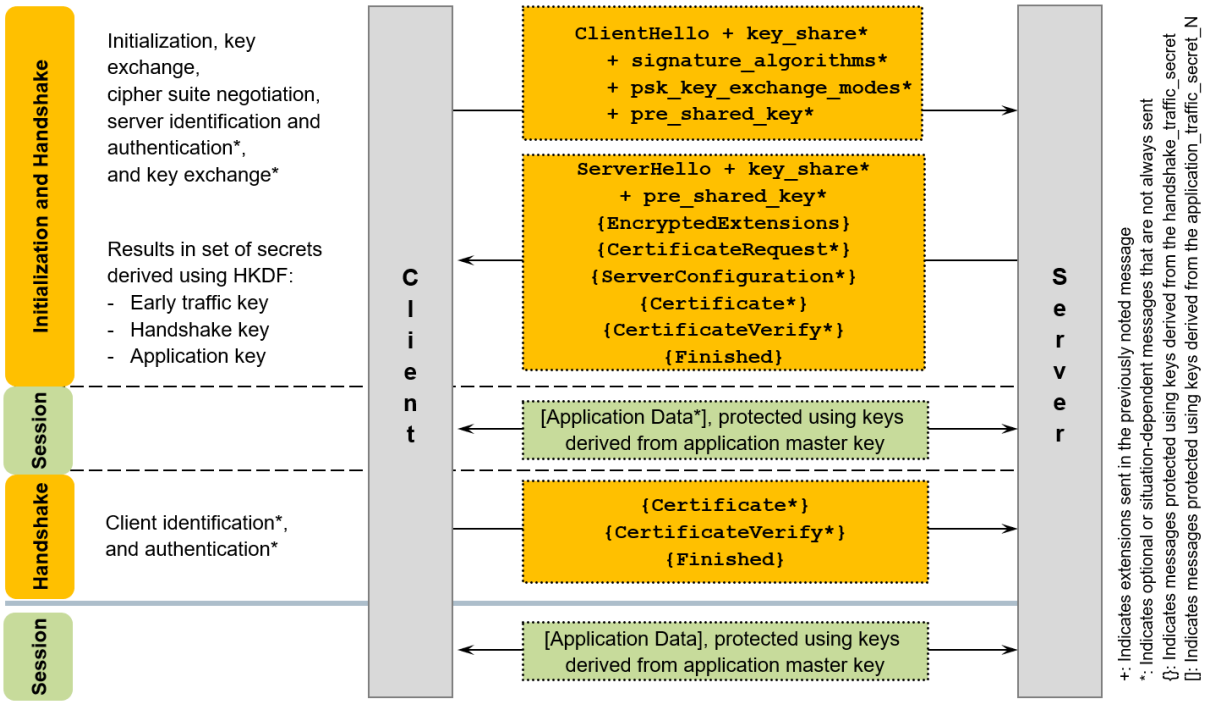


Figure 2. TLS v1.3 Session Establishment with full handshake

The full handshake of TLSv1.3 is depicted in Figure 2. TLS supports different authentication options:

- server-side authentication (mainly used in web traffic) using X.509 certificates;
- mutual authentication involves the client to authenticate using an X.509 certificate in addition to server authentication;
- authentication based on a pre-shared key, which is applied also within TLS as described below;
- authentication based on raw public keys.

Besides the peer authentication, the TLS handshake is used to negotiate further session parameters like the cipher suite for protecting communication integrity and confidentiality. A cipher suite is a statement regarding the utilized algorithms for the protection of the communication session.

TLS with mutual authentication is applied in ISO/IEC 15118-20 for plug-and-charge and for access to value-added services. This ensures that billing-relevant charging and service consumption can be associated with a dedicated account.

Besides the establishment of a protected channel, TLS defines further operations for the management of this secured channel, beyond them the update of session parameters during an ongoing session, like the utilized cryptographic key. One important functionality is the so-called session resumption. Session resumption allows a previously established and closed session to be resumed, based on the security parameters negotiated in the initial session. This saves the asymmetric cryptographic operations during the TLS handshake, and it utilizes a pre-shared key included in a ticket from the initial handshake. Note that there is a timely limitation how long a closed session may be resumed, depending on the TLS version. While TLSv1.2 recommends 24 hours, TLSv1.3 limits the validity time in the tickets used for resumption to seven days.

Besides the re-establishment of a closed connection, TLS session resumption may also be used to "clone" an existing session. This can be achieved by opening a TLS connection to a different port on the target host than the original one used and referencing the existing session. Using this, a separate TLS-protected TCP communication channel is established. As the second communication channel relies on the security parameters of the first one and thus is cryptographically bound to it, it implicitly provides the assurance of mutual authentication to both participants.

ISO/IEC 15118 utilizes this feature to allow the establishment of value-added service communication channels. Note that these are currently restricted to TCP-based communications.

In general. there also exists a protocol similar to TLS to protect UDP/IP traffic by the Datagram Transport Layer Security protocol (DTLS, IETF RFC 9147, [18]). It could be used to protect, e.g., media traffic, which is often transmitted via UDP. Note that interactions between both (TLS and DTLS) are not considered in ISO/IEC 15118, as the protection of the actual value-added service data is left to the value-added service itself. ISO/IEC 15118 mainly handles the secure establishment of a further communication session between the electric vehicle and the charging station to facilitate the value-added service communication.

As shown in Figure 3, a second session is opened between the electric vehicle and the charging station using TLS session resumption. This saves communication overhead and provides a cryptographic binding to the initial, authenticated TLS channel, which protects the charging control session. Note that while TLSv1.3 has specific optimizations like sending application data already in the resumed handshake (called 0-RTT), this feature is not allowed in ISO/IEC 15118-20 to avoid replay attacks of application data.

Port forwarding is used at the charging station to forward the traffic to the intended value-added service provider. The security of the communication channel to the value-added service provider is out of scope of ISO/IEC 15118 and needs to be defined and setup by the value-added service separately. For protecting UDP-based traffic between the electric vehicle and the charging station, OpenVPN is mentioned but nothing is specified. Note that this missing definition may have an influence on real-time services (like voice and video over IP or streaming services).

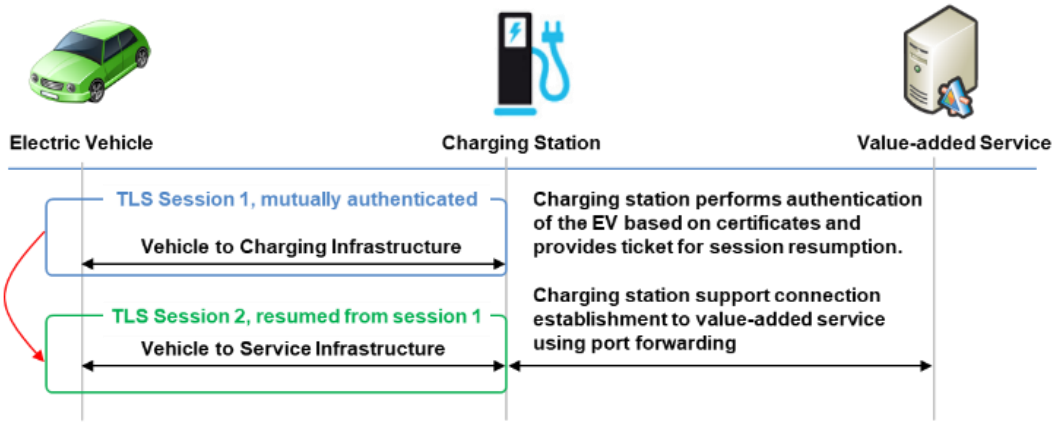


Figure 3. TLS Session Resumption to establish second communication channel

E. TLS Channel Binding

IETF RFC 5929 [19] describes a binding of a higher layer communication protocol to a negotiated TLS channel. Different approaches are specified. The most versatile is the definition of the *tls-unique* value. The *tls-unique* value is essentially the first "Finish Message" sent in the latest TLS handshake. The finish message contains a hash over all messages exchanged during the handshake phase.

This definition makes this parameter specific to a negotiated session. When a session is resumed or renegotiated (only for TLS 1.2), the *tls-unique* value will change accordingly. This has to be obeyed by the applying application. Using *tls-unique* in an application provides a direct linkage to the properties negotiated during the TLS handshake and applied in the ongoing TLS session.

An example is the application in the context of Enrollment over Secure Transport (EST, IETF RFC 7030, [20]), a certificate enrollment protocol executed over TLS. In this protocol, the client sends a certification request object (here, a PKCS#10 request) to enroll a new X.509 client certificate. The certification request is signed with the private key of the freshly generated key pair. This provides a proof-of-possession to the receiver, that the sender, i.e., the client, knows the private key corresponding to the contained public key. Part of the certification request can be a *tls-unique* value. As in case of EST, the TLS handshake may be performed with mutual authentication. Therefore, the receiver in addition gets the proof-of-identity of the client, due to the link to the utilized client certificate in the TLS handshake. This linkage is provided through the inclusion of the *tls-unique* value.

IV. Dependent Secure Communication Channels

As discussed in section I, the goal of this paper is to propose a solution for setting up an additional (wireless) communication channel or more specific an additional access path that utilizes security properties of a previously established communication channel to specifically gain and apply derivations of the original security parameters. The existing solutions discussed in section III provide elements that are partly used in the approach. The solution is described in the context of the initially provided example of electric vehicle charging and additionally as option in the context of dependent monitoring in the following subsections.

A. Alternative Handling of Value-Added Services communication during electric vehicle charging

The following description proceeds with the example of electric vehicle charging as in section II and provides an alternative solution. This described solution specifically allows for multiple connections between a value-added service provider and an electric vehicle, which are all bound to an existing charging session. These multiple channels may be of different nature like TCP/IP or UDP/IP traffic. This enhances the currently provided functionality for securing TCP/IP based value-added service communication. Therefore, the alternative may also be used in situation in which, e.g., video streaming or conferencing is provided, typically relying on UDP/IP for the real-time part.

Figure 4 provides an overview of the solution. According to ISO/IEC 15118-20, a TLS connection is established between the electric vehicle EV1 and the charging station CS1 via a well-known service-set identifier (SSID) of the charging station. The well-known SSID may be either preconfigured, or it may be broadcasted using Bluetooth beacons in the vicinity of the charging station. The connection is established based on the authentication of CS1 as server towards EV1. The EV1 authentication can be carried out over the already TLS protected link to protect the identity information of EV1.

The client-side authentication in the first communication channel may be done based on an X.509 certificate but also using other methods on application layer like HTTP digest authentication or based on a token, like for instance when using the OAuth 2.0 framework [22]. Specific for the electric vehicle charging, the owner of the EV may also authenticate directly towards the charging station, avoiding any information to be transmitted over the communication link. In each case, a binding to the originally established TLS connection is required.

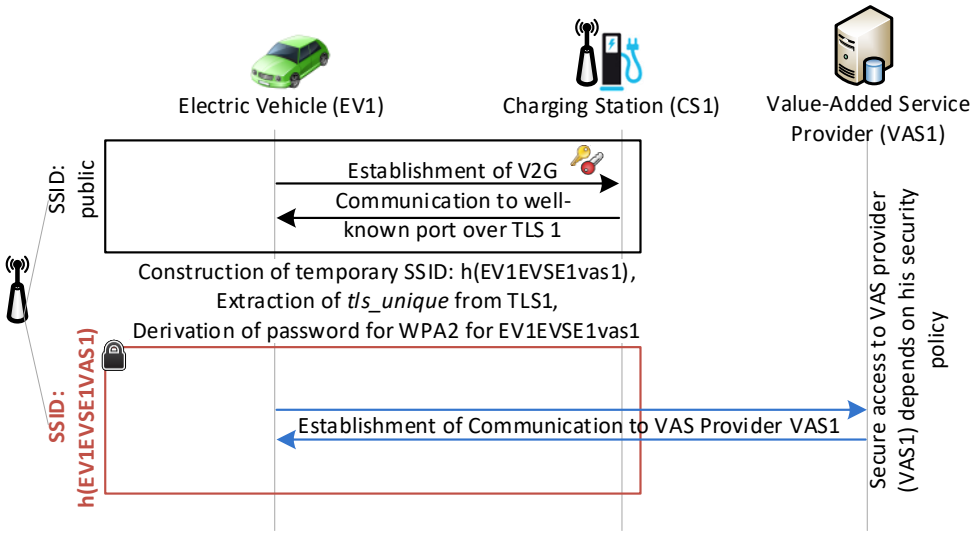


Figure 4. Application of *tls-unique* to protect second WLAN establishment

To achieve this, the *tls-unique* value is extracted, which is intended as means to provide the binding to the originally established TLS channel for further connections to be opened. This extraction equals to the TLS channel binding described in section III.E.

Over the established TLS channel, an information is provided to the electric vehicle regarding available value-added services offered by the charging station, which can be consumed during the charging period. These value-added services may be software updates for the infotainment system, normal web access, gaming, or videos to bridge the charging time.

While in section II, the additional communication channel for value-added services in ISO/IEC 15118 is opened using TLS session resumption on a different port than the one for the charging communication, the following describes an alternative, which can be used more versatile for different types of data exchange. It essentially provides a second access point thus allowing to perform further connections on OSI layer 2.

When the EV selects a value-added service, it will receive the additional configuration information for setting up a second, temporary WLAN access to the charging station for the electric vehicle. The configuration information shall be specific to the charging session between *EV1* and *CS1* and a specific value-added service provider *VAS1*. This allows for correct billing of consumed services (if necessary), based on the association.

For setting up a temporary access point, a second network access policy needs to be provided, which may comprise information regarding protection means or quality of service parameter. In case of WLAN, a temporary network name (SSID) and a pre-shared key for access protection to the temporary WLAN are also required to utilize WPA2 or WPA3 for access protection to the temporary WLAN.

Instead of providing this information directly, it can be derived locally by the communication peers based on the already existing charging control communication session as following:

Temporary SSID = Hash (EV ID | CS ID | VAS ID)

In the example in Figure 4, this will result in the hashed value of "*EV1CS1VAS1"*. Depending on the utilized hash function, e.g., SHA-256, the result can be truncated to, e.g., 20 Bytes. With the goal to bind the temporary WLAN to the already existing charging session, the temporary WLAN access credentials in terms of a shared secret are derived incorporating the *tls-unique* value of the initial TLS session as following:

Temp. SSID PW = Hash (tls-unique | EV ID | VAS ID)

The derivation may consist of further parameter besides the EV identifier and the VAS identifier. In addition to the VAS ID, the name of the value-added service may be provided, e.g., as fully-qualified domain name (FQDN).

Depending on the security policy of the charging station operator, the temporary WLAN access for the value-added services may be terminated as soon as the charging session ends. There may be cases for leaving the session open for a grace period, e.g., for ending a specific transaction. This option may also be part of the contract a customer has with a specific charging station operator.

As described, the approach can be generalized to provide the binding also to other network access methods like 4G or 5G. It may also be leveraged to setup VLANs for separating the communication, utilizing derived parameters for VLAN name and access credentials.

B. Utilizing dependent security channels for device monitoring

In the previous section, the existence of an established secure session was used to open a second network path to provide value-added services during electric vehicle charging as example use case.

The approach to have a specific network access for a function allows for better maintaining quality of service characteristics for this communication channel or for a dependent communication channel. This can also be leveraged and applied in other types of communication. A further example is provided in this subsection based on the monitoring of industrial device operation.

In industrial use cases like process automation, factory automation, or also energy automation, there are often strict boundary conditions with respect to the expected real-time behavior. This relates to strict processing and transmission time specification in applications like protection in electrical networks. As the communication between protection devices has been changed from a per wire communication to Ethernet based communication, the timing of the direct wiring needs to be ensured also for the Ethernet-based communication to ensure reliability for the power provisioning but most importantly for safety. The protection device essentially resembles the same functionality as an electrical fuse, known from typical household applications. Protocols used in this context are defined for instance in IEC 61850 with Generic Object-Oriented Substation Events (GOOSE), and Sample Values (SV) using plain Ethernet-based communication in substation environments.

Following mechanisms are used to ensure the specified transmission speed and reliability:

- GOOSE data is directly embedded into Ethernet data packets and works on publisher-subscriber mechanism on multicast or broadcast MAC (Media Access Control) addresses
- GOOSE uses VLAN and priority tagging as per IEEE 802.1Q to have a separate virtual network within the same physical network and to set an appropriate message priority level
- Enhanced retransmission mechanisms – the same GOOSE message is retransmitted with varying and increasing re-transmission intervals.

IEC 61850-5 [24] defines message types and their performance classes as:

- P1 applies typically to a distribution bay (or where low requirements can be accepted),
- P2 applies typically to a transmission bay (or if not otherwise specified by the customer),

– P3 applies typically to a top performance transmission bay.

The following table shows the different message types and their timing requirements based on IEC 61850-5 [24].

TABLE I. GOOSE TRANSFER TIMES

Type	Definition	Timing Requirements
1	**Fast messages** contain a simple binary code containing data, command or simple message, examples are: "Trip", "Close", etc.	See Type 1a and 1 b below
1A	**TRIP** – most important message	– P1: Transfer time shall be in the order of half a cycle. → 10 ms – P2/3: Transfer time shall be below the order of a quarter of a cycle. → 3 ms
1B	**OTHER** – Important for the interaction of the automation system with the process but have less demanding requirements than trip.	– P1: Transfer time < 100ms – P2/3: Transfer time shall be below the order of one cycle. → 20 ms
2	**Medium speed messages** are messages where the time at which the message originated is important but where the transmission time is less critical.	– Transfer time < 100ms
3	**Low speed messages** are used for slow speed auto-control functions, transmission of event records, reading or changing set-point values and general presentation of system data.	– Transfer time < 500ms

The definition of transfer time, according to IEC 61850-5, is shown in Figure 5 below. The transfer time includes the complete transmission of a message including necessary handling at both ends.

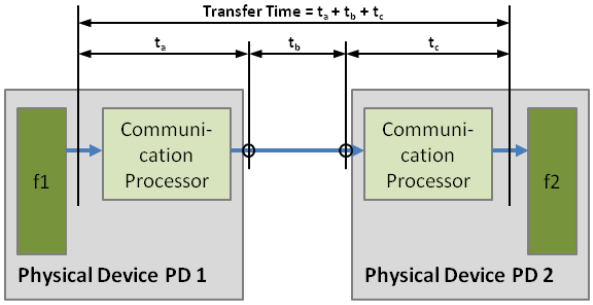


Figure 5. Transfer Time [24]

The security for GOOSE and also SV communication is specified in IEC 62351-6 [25] and defines an extension applicable to both protocols to carry integrity information for the exchanged information based on independently negotiated security parameters.

An alternative approach to protocol inherent security may be the setup of a dependent security association between the GOOSE and the SV communication. This on one hand supports the direct assignment of QoS parameters to the communication channel. In addition, if the communication is performed using different network technologies, a related security attack on one (physical or logical) network can be directly related to the communication on the other and appropriate measures can be derived.

As described, the approach can be generalized to provide the binding also to other network access methods like 4G or 5G. It may also be leveraged to setup VLANs for separating the communication, utilizing derived parameters for VLAN name and access credentials.

V. EVALUATION

The evaluation of the proposed solution is done based on the concept only, as it has not been implemented, yet. In general, the security of an industrial system can be evaluated in practice in various approaches and stages of the system's lifecycle:

– A Threat and Risk Analysis (TRA, also abbreviated as TARA) is typically conducted at the beginning of the concept definition, as for ISO/IEC 15118, product design or system development, and updated after major design changes, or to address a changed threat landscape. In a TRA, possible attacks (threats) on the system are identified. The impact that would be caused by a successful attack and the probability that the attack happens are evaluated to determine the risk of the identified threats. The risk evaluation allows to prioritize the threats, focusing on the most relevant risks and to define corresponding security measures. Security measures can target to reduce the probability of an attack by preventing it, or by reducing the impact.

– Security checks can be performed during operation or during maintenance windows to determine key performance indicators (e.g., check compliance of device configurations) and to verified that the defined security measures are in fact in place.

– Security testing (penetration testing, also called pentesting for short) can be performed for a system that has been built, but that is currently not in operation. A pentest can usually not be performed on an operational automation and control system, as the pentest could affect the reliable operation auf the system. Pentesting can be performed during a maintenance window when the physical system is in a safe state or using a separate test system.

As long as the solution proposed in the paper has not been proven in a real-world operational setting, it can be evaluated conceptually by analyzing the impact that the additional security measure would have on the identified residual risks as determined by a TRA. The main objective is to determine the specific benefits that are relevant for the selection of a suitable protection approach. The main aspects relevant for the evaluation of the proposed solution are:

a. the level of isolation of different types of communications (charging control communication; value added services communication);
b. the scope of protection, i.e., what exactly is protected concerning integrity and or confidentiality, and
c. the flexibility to use it for various protocols used by different value-added services.

These aspects can be evaluated qualitatively as follows:

a. The control communication for charging control and the communication of value-added services are taking place on separate layer 1 / layer 2 communication links. While a reliable traffic isolation can be implemented also on a logical level, the isolation realized by having separate layer 1 / layer 2 communication links ensures by design a strong isolation, avoiding logical interference between these different types of communications. Moreover, this separation offers the option to not only provide different protection options for the communication links, but also to assign different quality of services classes to ensure for instance a dedicated throughput or latency.
b. The proposed solution protects all communications, including, e.g., dynamic host configuration by DHCP or iPv6 auto configuration, or DNS requests. Thereby, also user privacy protection is increased, as meta-data of communication as, e.g., network addresses, cannot be intercepted as all communication is protected on layer 2. Also, active manipulations by 3rd parties, e.g., injected false DNS responses, can be avoided.
c. The solution can be used with any types of communication, including UDP datagram communication. So, it can be flexibly applied also for value-added services using UDP-based communications (e.g., multi-media communications based on RTP).

VI. Conclusion

This paper provides a new generic approach for setting up a temporary network access channel allowing to assign specific quality of service parameter to the new network access, which is cryptographically bound to an already established communication channel. The approach is discussed in the context of electric vehicle charging combined with value-added services. A further example from the power automation domain is described. The approach as such is not limited to these examples and may be applied also in other domains.

The advantage of the proposed approach is the ability to be applied in an application layer protocol independent way. It preserves the privacy of user credentials for observers of the network applied on higher layers. This is especially important for wireless communication as the exchanged communication can be easily accessed. Note that in the design of TLS 1.3, the privacy requirements have already been considered in the redesign phase. In TLS 1.3, only the initial message is sent in clear, while the remaining part of the handshake, including the client-side authentication, is encrypted. Note that there are ongoing discussions in the IETF standardization on securing the initial TLS handshake message to further protect the privacy of the communication [26].

The proposed approach in this paper is available as concept and needs to be implemented a proof of concept, which would be a future intended step. Such a proof of concept can leverage already specified base mechanisms like *tls-unique* extraction. Moreover, in the context of ISO/IEC15118 it would align with the approach not sending the client authentication (here the client is the electric vehicle) in clear over the network.

References

[1] S. Fries and R. Falk, "Secure and Flexible Establishment of Temporary WLAN Access", Securware 2022, July 2022, https://www.thinkmind.org/articles/securware_2022_1_40_30013.pdf, [retrieved: May, 2023]

[2] ISO/IEC 15118-20: Road vehicles — Vehicle-to-Grid Communication Interface — Part 20: Network and application protocol requirements, Work in Progress

[3] CHAdeMO, https://www.chademo.com/, [retrieved: May, 2023]

[4] R. Falk and S. Fries, "Electric Vehicle Charging Infrastructure – Security Considerations and Approaches", Internet 2012, June 2012, ISBN: 978-1-61208-204-2, pp.58-64

[5] T. Dierks and E. Rescorla, IETF RFC 5246, "Transport Layer Security (TLS) Protocol v1.2, 08/2008, https://tools.ietf.org/html/rfc5246, [retrieved: May, 2023]

[6] E. Rescorla, IETF RFC 8446, "Transport Layer Security (TLS) Protocol v1.3", 08/2018, https://tools.ietf.org/html/rfc8446, [retrieved: May, 2023]

[7] N. Williams, IETF RFC 5056, "On the Use of Channel Bindings to Secure Channels", 11/2007, https://tools.ietf.org/html/rfc5056, [retrieved: May, 2023]

[8] ISO/IEC 7498-1:1994, Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model, November 1994, https://www.iso.org/standard/20269.html, [retrieved: May, 2023]

[9] S. Kent and K. Seo, IETF RFC 4301, "Security Architecture for the Internet Protocol", https://tools.ietf.org/html/rfc4301, December 2005, [retrieved: May, 2023]

[10] OpenVPN, https://openvpn.net/, [retrieved: May, 2023]

[11] Wireguard, https://www.wireguard.com/, [retrieved: May, 2023]

[12] Noise Protocol Framework, http://www.noiseprotocol.org/, [retrieved: May, 2023]

[13] M. Leech et al., IETF RFC 1928, „SOCKS Protocol Version 5, 03/1996, https://tools.ietf.org/html/rfc1928, [retrieved: May, 2023]

[14] IEEE 802.1Q, "IEEE Standard for Local and Metropolitan Area Networks – Bridges and Bridged Networks", 2018, https://standards.ieee.org/ieee/802.1Q/6844/, [retrieved: May, 2023]

[15] IEEE 802.1X, "IEEE Standard for Local and Metropolitan Area Networks – Port-Based Access Control", 2020, https://ieeexplore.ieee.org/document/9018454, [retrieved: May, 2023]

[16] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H.Levkowetz., IETF RFC 3748, "Extensible Authentication protocol (EAP)", 06/2004, https://tools.ietf.org/html/rfc3748, [retrieved: May, 2023]

[17] IEEE 802.1AE "IEEE Standard for Local and Metropolitan Area Networks – Media Access Control (MAC) Security",

2018, https://ieeexplore.ieee.org/document/8585421, [retrieved: May, 2023]

[18] E. Rescorla, H. Tschofenig, and N. Modadugu, IETF RFC 9147, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", April 2022 https://datatracker.ietf.org/doc/html/rfc9147, [retrieved: May, 2023]

[19] J. Altman and N. Williams, IETF RFC 5929, TLS channel binding, July 2010, https://tools.ietf.org/html/rfc5929, [retrieved: May, 2023]

[20] M. Pritikin, P. Yee, and D. Harkins, IETF RFC 7030, "Enrollment over Secure Transport ", 10/2013, https://tools.ietf.org/html/rfc7030, [retrieved: May, 2023]

[21] SSL Puls: TLS Dashboard, continuously updated, https://www.ssllabs.com/ssl-pulse/, [retrieved: May, 2023]

[22] D. Hardt, IETF RFC 6749, "The OAuth 2.0 Authorization Framework", October 2012, https://tools.ietf.org/html/rfc6749, [retrieved: May, 2023]

[23] D. Hardt, A.Parecki, and T. Lodderstedt, IETF Draft, "The OAuth 2.1 Authorization Framework", March 2023, https://datatracker.ietf.org/doc/draft-ietf-oauth-v2-1/, [retrieved: May, 2023]

[24] IEC 61850-5 – "Communication networks and systems for power utility automation - Part 5: Communication requirements for functions and device models", March 2022, https://webstore.iec.ch/publication/64584, [retrieved: May, 2023]

[25] IEC 62351-6 – "Power systems management and associated information exchange - Data and communications security - Part 6: Security for IEC 61850", October 2020, https://webstore.iec.ch/publication/63742, [retrieved: May, 2023]

[26] E. Rescorla, K. Oku, N. Sullivan, and C. A. Wood, IETF Draft "TLS Encrypted Client Hello", April 2023, https://datatracker.ietf.org/doc/draft-ietf-tls-esni/, [retrieved: May, 2023]