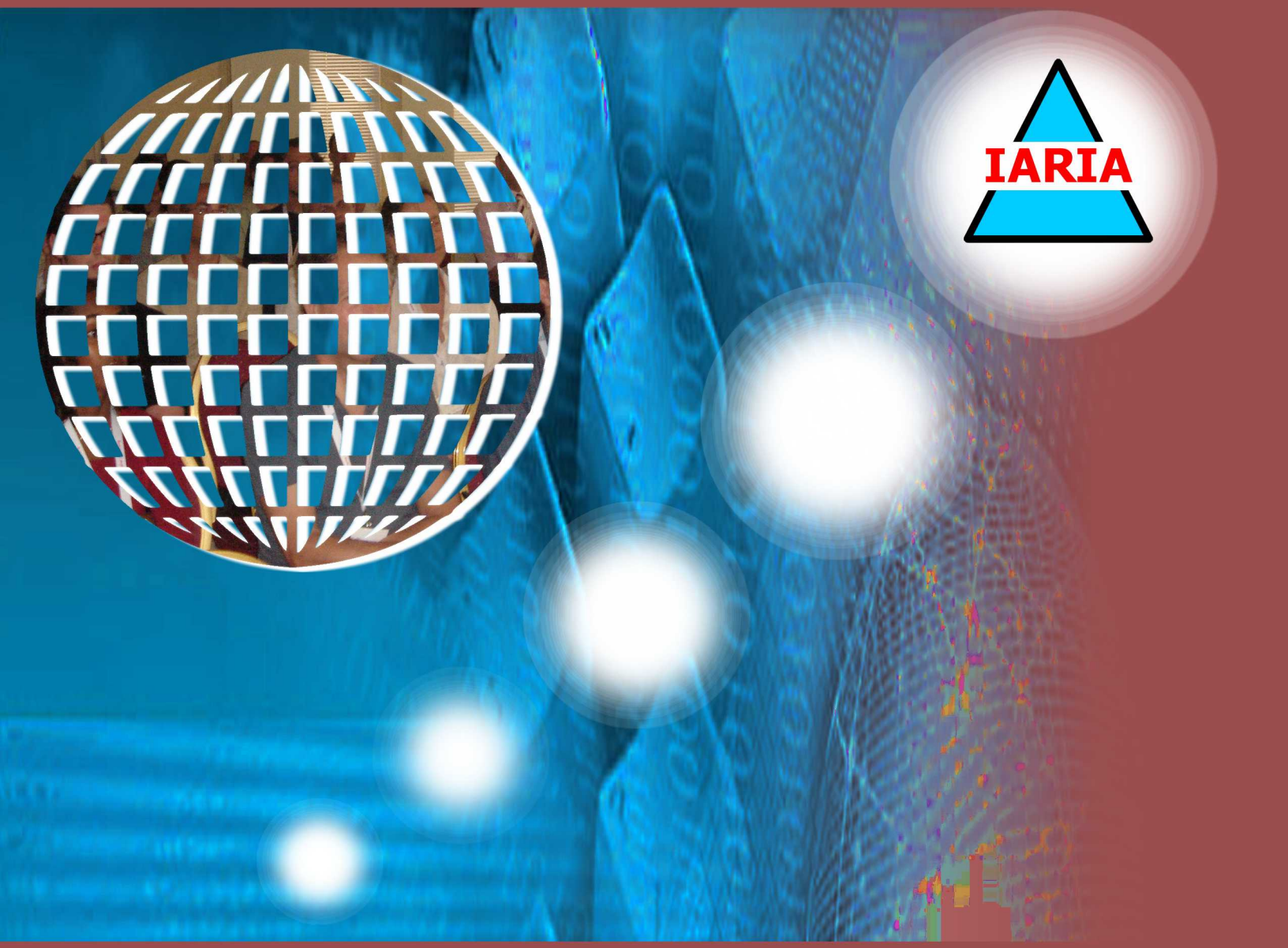
International Journal on Advances in Security



2025 vol. 18 nr. 3&4

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

Responsibility for the contents rests upon the authors and not upon IARIA, nor on IARIA volunteers, staff, or contractors.

IARIA is the owner of the publication and of editorial aspects. IARIA reserves the right to update the content for quality improvements.

Abstracting is permitted with credit to the source. Libraries are permitted to photocopy or print, providing the reference is mentioned and that the resulting material is made available at no cost.

Reference should mention:

International Journal on Advances in Security, issn 1942-2636
vol. 18, no. 3&4, year 2025, http://www.iariajournals.org/security/

The copyright for each included paper belongs to the authors. Republishing of same material, by authors or persons or organizations, is not allowed. Reprint rights can be granted by IARIA or by the authors, and must include proper reference.

Reference to an article in the journal is as follows:

<Author list>, "<Article title>"
International Journal on Advances in Security, issn 1942-2636
vol. 18, no. 3&4, year 2025, <start page>:<end page> , http://www.iariajournals.org/security/

IARIA journals are made available for free, proving the appropriate references are made when their content is used.

Sponsored by IARIA
www.iaria.org

Copyright © 2025 IARIA

International Journal on Advances in Security
Volume 18, Number 3&4, 2025

Editors-in-Chief

Hans-Joachim Hof,
- Full Professor at Technische Hochschule Ingolstadt, Germany
- Lecturer at Munich University of Applied Sciences
- Group leader MuSe - Munich IT Security Research Group
- Group leader INSicherheit - Ingolstädter Forschungsgruppe angewandte IT-Sicherheit
- Chairman German Chapter of the ACM

Editorial Board

Oum-El-Kheir Aktouf, Univ. Grenoble Alpes | Grenoble INP, France
Eric Amankwa, Presbyterian University, Ghana
Ilija Basicevic, University of Novi Sad, Serbia
Cătălin Bîrjoveanu, "Al.I.Cuza" University of Iasi, Romania
Steve Chan, Decision Engineering Analysis Laboratory, USA
Abdullah S. Al-Alaj, Virginia Wesleyan University, USA
El-Sayed M. El-Alfy, King Fahd University of Petroleum and Minerals, Saudi Arabia
Rainer Falk, Siemens Technology, Germany
Steffen Fries, Siemens AG, Germany
Damjan Fujs, University of Ljubljana, Slovenia
Hans-Joachim Hof, Technische Hochschule Ingolstadt, Germany
Gahangir Hossain, University of North Texas, Denton, USA
Fu-Hau Hsu, National Central University, Taiwan
Sokratis Katsikas, Norwegian University of Science and Technology - NTNU, Norway
Hyunsung Kim, Kyungil University, Korea
Dragana Krstic, University of Nis, Serbia
Yosra Lakhdhar, Digital Research Center of Sfax (CRNS) / CN&S Research Lab at SUP'COM, Tunisia
Petra Leimich, Edinburgh Napier University, UK
Shimin Li, Winona State University, USA
Yi Liu, University of Massachusetts Dartmouth, USA
Giuseppe Loseto, LUM "Giuseppe Degennaro" University, Italy
Mohammadreza Mehrabian, South Dakota School of Mines and Technology, USA
Carla Merkle Westphall, Federal University of Santa Catarina (UFSC), Brazil
Aleksandra Mileva, Goce Delcev University, Republic of N. Macedonia
Vasudevan Nagendra, Sekyurity AI, USA
Brajendra Panda, University of Arkansas, USA
Paweł Rajba, University of Wroclaw, Poland
Danda B. Rawat, Howard University, USA
Claus-Peter Rückemann, Universität Münster / DIMF / Leibniz Universität Hannover, Germany
Antonio Ruiz Martínez, University of Murcia, Spain
Rocky Slavin, University of Texas at San Antonio, USA

Pedro Sousa, University of Minho, Braga, Portugal
Miroslav Velev, Aries Design Automation, USA
Cong-Cong Xing, Nicholls State University, USA

International Journal on Advances in Security
Volume 18, Numbers 3&4, 2025

CONTENTS

Extending SDN-ACL Automation with User Groups, Authentication Events, and Intrusion Detection System Integration

Florian Grießer*, Hirokazu Hasegawa‡, Hajime Shimada§
*School of Computation, Information and Technology, Technical University Munich,
Chair of Security in Information Technology, Germany
‡Center for Strategic Cyber Resilience R&D, National Institute of Informatics, Japan
§Information Technology Center, Nagoya University, Japan
florian.griesser@tum.de, hasegawa@nii.ac.jp, shimada@itc.nagoya-u.ac.jp

***Abstract*—As cyberattacks become more common and advanced, traditional networks fall behind because they depend on static settings and manual adjustments. Software-Defined Networking (SDN) provides more flexibility and can be used to solve these problems. In this work, we present a system that automatically creates Access Control Lists (ACLs) in SDN environments. The system links access control to the User Database and generates rules automatically, which reduces the effort for administrators. With Port Access Control, only authenticated devices are allowed to use network resources. In addition, the system integrates an Intrusion Detection System (IDS): suspicious clients are first monitored through mirrored traffic, and if violations go beyond a threshold, their traffic is redirected for inline inspection. We tested the system in three use cases: connecting new clients, adapting dynamically to authentication events, and redirecting malicious hosts through the IDS. The results show that our approach not only reduces manual work but also enforces role-based security and prevents IDS overload by escalating only persistent or severe attacks.**

***Keywords*-Software-Defined Networking; Authentication; Access Control Lists; Intrusion Detection Systems**

I. Introduction

This article is an extended version of our previous work presented at the International Conference on Networks (ICN) [1], where we introduced a system for automatically generating Access Control Lists (ACLs) within Software-Defined Networking (SDN) environments. While the conference paper focused on the system architecture and initial evaluation, this extended version expands the analysis in several directions. In particular, we introduce a more comprehensive *Formal Security Model and Threat Analysis* (Section V) and extend the system with an *Intrusion Detection System (IDS)* for dynamic policy adaptation. We further examine how identity and authentication events can drive the automated creation of fine-grained access control rules in SDN environments and how such mechanisms adapt to user behavior while maintaining scalability, security, and low administrative overhead. The core challenge we address is how to integrate authentication-driven, identity-aware access control into SDN in a way that adapts dynamically to user behavior while preserving scalability and security. The primary aim of this work is to demonstrate the feasibility and practical applicability of the proposed system in realistic scenarios, with a focus on validating the architectural concept and its security properties rather than conducting an exhaustive performance study.

Digital transformation has exponentially increased the complexity of network architectures, presenting significant challenges in maintaining robust security frameworks [2]. In this ever-evolving digital landscape, cybersecurity threats have become more sophisticated, leveraging the linkage of modern infrastructures to exploit vulnerabilities at an alarming rate. Traditional network security mechanisms, which mainly rely on static configurations and manual oversight, are increasingly proving inadequate against this backdrop of dynamic and evolving threats [2]. The inherent limitations of these conventional approaches, characterized by their inflexibility and slow response times, underline the urgent need for more adaptable, responsive security measures.

Software-Defined Networking (SDN) is a paradigm that promises to redefine network management and security [3]. At its core, SDN separates the network's control logic from the underlying hardware, facilitating a centralized and programmable framework that transcends traditional hardware limitations [4]. This separation enhances network flexibility and management and introduces agility and adaptability that were unachievable with conventional network architectures until now. According to a report by Global Market Insights, the SDN market, valued at USD 28.2 billion in 2023, is expected to experience significant growth, with a projected expansion rate exceeding 17% annually from 2024 to 2032 [5]. Through SDN's capabilities, networks gain the flexibility to adapt swiftly to evolving security demands. This flexibility enables the immediate implementation of tailored security measures and configurations to counter new threats effectively, as illustrated in the study by Ali et al. [6].

Furthermore, our contribution is complemented by the work of Yakasai et al. in FlowIdentity, which advances virtualized network access control within SDN through a role-based firewall [7]. We also build on the architectural insights of Casado et al. in Ethane, demonstrating the power of centralized policy enforcement [8], and the approach of Mattos et al. in AuthFlow, focusing on authentication and access control mechanisms in SDN environments [9].

Additionally, this approach was refined by incorporating a structured analysis of authentication logs, drawing upon the work of Xing et al. in SnortFlow, which explores an OpenFlow-based intrusion prevention system in cloud environments [10], and the study by An Le et al. on a flexible

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

network-based intrusion detection and prevention system on Software-Defined Networks [11].

The paper progresses from reviewing related SDN security work in Section II to foundational concepts in Section III. Section IV describes our system for automating ACLs, followed by Section V, which presents the formal security model and analyzes potential threats. Section VI details the implementation, Section VII evaluates the system's performance and demonstrates the benefits of the IDS integration, and Section VIII encapsulates concluding thoughts and future directions.

II. Related Work

Network security and access control advancements are crucial in the evolving landscape of SDN. The following studies demonstrate that emerging technologies and frameworks are pivotal in addressing these challenges.

A. Intrusion Detection with Authentication Events

In the study by Chu et al. [12], "ALERT-ID," an intrusion detection system for large-scale network infrastructures, is presented. The system distinguishes between normal operations and potential security threats through real-time analysis of authentication, authorization, and accounting (AAA) system logs. It employs behavioral models built on historical access patterns and user profiles, efficiently identifying potential intrusions and misuse. Notably, ALERT-ID balances the need for thorough security monitoring with a manageable false alarm rate, demonstrating the importance of dynamic security measures in complex network environments.

Building on this, Janabi et al. provide a comprehensive survey of intrusion detection systems in Software-Defined Networking [13]. Their work categorizes existing approaches into anomaly-based, signature-based, and hybrid detection mechanisms, and highlights key challenges such as scalability, controller bottlenecks, and false alarm reduction. This survey underscores the urgency of developing IDS solutions that are tailored to the dynamic characteristics of SDN environments.

A more specialized perspective is given by Susilo et al., who present an SDN-based intrusion detection system that leverages deep learning techniques [14]. By training models on traffic data, they achieve high detection rates for a variety of attack types, demonstrating the potential of machine learning to significantly enhance IDS performance in SDN settings.

In a related direction, Wang et al. propose an AI-powered network threat detection system [15]. Their approach integrates artificial intelligence methods such as Random Forests and neural networks into SDN Controllers to enable real-time threat detection. The study highlights the effectiveness of AI in reducing false positives and improving scalability, though it also raises concerns regarding explainability and computational overhead.

B. Dynamic Access Control in SDN

Transitioning to dynamic access control, the work by Nayak et al. introduces "Resonance: Dynamic Access Control for Enterprise Networks" [16]. Resonance implements dynamic security policies with a registration phase, complemented by real-time monitoring and inference mechanisms specified by administrator rules.

A more recent study by Shah and Yadav integrates IEEE 802.1X authentication into SDN to control port-level access [17]. Their approach provides admission control based on authentication status, yet it does not support dynamic ACL updates or identity-based rule generation. This gap is central to our contribution, which leverages authentication events and user group information to automatically derive and update fine-grained, identity-aware ACLs.

Further extending the concept of network security, the study by Martins et al. [18] introduces an access control architecture for SDN leveraging the ITU X.812 standard. This framework incorporates Role-Based Access Control (RBAC) with traffic prioritization rules, advancing towards more granular access control based on predefined role mappings. While powerful, this approach depends on extensive manual configuration efforts to establish complete rule sets, limiting its ability to adapt dynamically to user behavior or evolving authentication contexts.

C. Formal Security Models & Threat Modeling in SDN

Beyond practical implementations, several works emphasize the necessity of formal reasoning in SDN security. Sharma and Tyagi present a structured threat model for SDN environments, covering controller protection, inter-controller communication, and malicious switch scenarios [19]. Their taxonomy illustrates how different layers of the SDN stack are exposed to unique attack vectors, motivating the need for systematic threat analysis.

Pradeep et al. propose EnsureS, an SDN security model that validates service paths based on efficient hashing and tag verification mechanisms [20]. Their design provides both efficiency and security guarantees, reducing the risk of path manipulation and enhancing packet integrity.

Finally, Meng et al. introduce a policy model transformation and verification framework [21]. By automatically converting high-level security policies into flow-level configurations and applying formal verification techniques, they ensure consistency between intended policies and deployed rules. This approach demonstrates the potential of combining formal methods with SDN programmability to achieve provable security properties.

D. Policy Conflict Detection and Resolution in SDN

In parallel with dynamic access control, substantial work has addressed the challenge of policy conflict detection and resolution in SDN environments. Systems like FortNOX [22] and FlowGuard [23] provide real-time enforcement mechanisms that prevent new flow rules from violating established security policies by checking for conflicts during rule insertion. Verification tools such as VeriFlow [24] and NetPlumber [25] take a verification-oriented approach, intercepting flow updates to ensure they do not violate global invariants such as isolation

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

or reachability. More recent frameworks, including PGA [26] and Brew [27], focus on composing and reconciling policies from different modules or tenants, producing a consistent, conflict-free rule set. These efforts highlight the importance of handling interactions between dynamic ACLs, legacy firewall configurations, and other network policies, which is particularly relevant in hybrid and large-scale deployments.

These studies illustrate a significant progression in SDN security research, ranging from intrusion detection systems to dynamic access control and formal threat modeling. Building on these developments, our work reduces administrative burden while enabling adaptive, security-driven policy updates.

III. Preliminary Concepts

This section introduces foundational concepts relevant to network management and security, including OpenFlow switches in SDN, 802.1X Port-Based Network Access Control, access management with Active Directory and LDAP-based directory services, and the Extensible Authentication Protocol over LAN (EAPOL).

A. The Role of OpenFlow Switches in SDN

Software-Defined Networking (SDN) represents a paradigm shift in network management by decoupling the control plane from the data plane. OpenFlow, one of the earliest and most widely adopted SDN protocols, provides a standardized interface for communication between the centralized controller and the network devices. Within this architecture, OpenFlow switches serve as the essential data plane components, responsible for forwarding packets based on flow rules received from the controller [28].

These programmable switches enable dynamic network control, granular traffic engineering, and the implementation of advanced security policies. Because the control logic resides in a centralized controller, network behavior can be reconfigured on the fly without the need for manual reprogramming of individual devices. This centralized programmability simplifies policy enforcement and enables rapid adaptation to changing security requirements.

While OpenFlow switches provide strong flexibility, they also introduce security challenges. Because switches rely on the controller for all flow decisions, a compromised controller could install malicious rules or bypass policies. In addition, limited flow-table capacity makes them vulnerable to exhaustion attacks such as flow flooding [29].

The communication channel between switches and the controller, typically protected by TLS, can also be a point of failure if improperly configured [3]. Authentication, certificate management, and key distribution therefore become crucial aspects of securing the SDN infrastructure. Finally, the lack of default policies or fallbacks in many OpenFlow implementations can result in a denial-of-service condition if the controller becomes unreachable [29].

To address these issues, several countermeasures have been proposed. These include distributed controller architectures to eliminate a single point of failure, flow aggregation to reduce table pressure, and anomaly detection systems to identify malicious flow behaviors [3, 4]. Proper integration with access control mechanisms, such as 802.1X and centralized identity management, can further enhance the trustworthiness of OpenFlow-based networks [30, 31, 32].

In summary, OpenFlow switches provide the flexibility and programmability needed for next-generation networks but must be deployed with careful attention to security design, controller hardening, and flow policy management.

B. Port-Based Authentication with IEEE 802.1X

IEEE 802.1X Port-Based Network Access Control significantly strengthens network security by implementing stringent access control at the physical port level. As a foundational component of network admission control, 802.1X ensures that only authenticated devices and users gain access to the network, thereby maintaining integrity and reducing the risk of unauthorized intrusion [30].

The 802.1X framework operates with three core entities: the supplicant (client device), the authenticator (typically a switch or wireless access point), and the authentication server (commonly a Remote Authentication Dial-In User Service (RADIUS) server). When a device connects to a network port, it is initially placed into an unauthorized state. The authenticator acts as an intermediary, forwarding authentication messages between the supplicant and the server using the Extensible Authentication Protocol over LAN (EAPOL) [33].

The authentication sequence involves an initial access request, followed by a secure exchange of identity and credentials, and concludes with the authentication server evaluating the credentials and instructing the authenticator whether to grant or deny access. This structured process prevents unauthorized devices from entering the network and defends against threats such as MAC spoofing, rogue clients, and replay attacks. Mutual authentication using Extensible Authentication Protocol – Transport Layer Security (EAP-TLS) further enhances security by preventing credential interception and significantly reducing the risk of man-in-the-middle attacks.

Advanced deployments of 802.1X often integrate dynamic network configurations, such as VLAN assignment and ACL enforcement, based on the identity of the authenticated entity. This enables granular policy control and flexible segmentation of network resources. For example, guests, employees, and devices can be placed in separate VLANs with tailored permissions, immediately upon authentication.

While 802.1X offers robust security, its implementation can be complex. Challenges include managing certificates for mutual authentication, ensuring client compatibility, and avoiding service disruptions due to misconfiguration [30, 33]. Nonetheless, when properly deployed and integrated with directory services like Active Directory, 802.1X forms a critical layer of defense in modern enterprise networks.

In conclusion, IEEE 802.1X is an essential mechanism for enforcing authenticated, role-based access at the network edge. It combines strong identity verification with flexible policy

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

application, forming a resilient barrier against unauthorized access and internal threats.

C. Centralized Access Management via Active Directory and LDAP Authentication

Access control is a foundational building block of network security, ensuring that only authorized users can access critical systems and resources. Two widely used technologies for implementing centralized access management are Microsoft's Active Directory (AD) [31] and the Lightweight Directory Access Protocol (LDAP) [32]. Both systems facilitate authentication, authorization, and user management across a range of services and devices, albeit with different implementations and scopes.

Active Directory (AD) is a directory service developed by Microsoft for Windows domain networks. It acts as a centralized repository of user credentials, group memberships, and security policies. AD enables administrators to enforce Role-Based Access Control (RBAC) through Group Policy Objects (GPOs), automate login scripts, and manage user rights at scale. It also supports Kerberos-based authentication, which enhances security through ticket-based access mechanisms.

LDAP, on the other hand, is an open and vendor-neutral protocol used to access and manage distributed directory information services. While AD itself supports LDAP as one of its interfaces, LDAP can also be implemented in a platform-agnostic manner using solutions such as OpenLDAP or Apache Directory. LDAP directories typically organize information in a hierarchical structure and are used for centralizing authentication across services such as email servers, intranet portals, VPNs, and UNIX systems.

In enterprise environments, a centralized identity and access management (IAM) system is often built around AD, which integrates seamlessly with LDAP-aware services. This centralization facilitates consistent enforcement of security policies, simplifies user onboarding and offboarding, and reduces the administrative burden associated with managing multiple local accounts. It also enables Single Sign-On (SSO), allowing users to authenticate once and gain access to a variety of authorized services without repeated logins.

Moreover, integration with federated identity systems and multi-factor authentication (MFA) further strengthens the security posture. AD and LDAP are often combined with other authentication frameworks, such as Security Assertion Markup Language (SAML) and OAuth (Open Authorization), especially in hybrid environments that span both on-premises and cloud infrastructures.

Overall, the use of Active Directory and LDAP for access management supports scalability, interoperability, and a high level of control, making them indispensable components of modern enterprise security architectures.

IV. Proposed System

This section presents a comprehensive overview of our proposed system, designed to significantly enhance network security and efficiency through advanced ACL management and authentication mechanisms.

A. Previous Works: Problem and Approach

Building on our established framework for automating ACL generation through statistical analysis of communication patterns, this work seeks to further leverage and enhance the existing infrastructure. Our initial efforts in [34] laid the foundational groundwork for this approach, which saw significant development and refinement in subsequent studies, such as [35]. A primary challenge identified in our exploration was addressing the need for authentication proof for IP addresses.

We have refined our approach to take advantage of a typical user database, like Active Directory [31], a standard part of a company network. This centralized database offers a significant advantage because user and group management and their corresponding resource access permissions are already handled there. We aim to leverage this existing infrastructure to streamline the process and eliminate redundant tasks for administrators.

B. Architecture of the Proposed System

In the proposed system, we enhance network security through Port Access Control, which limits network port access exclusively to authenticated users. This approach is grounded in a security model where each port is individually secured and requires authentication before granting access. As a result, each user undergoes an authentication process, enhancing the network's overall security posture. The process for user connection is designed with precision to ensure a secure and efficient authentication mechanism and can be found in Figure 1.

Initially, the system is configured to allow only EAPOL messages, which are then directed to an authenticator component. This step ensures that there is no communication before the client authenticates.

The SDN Controller checks its internal state for preconfigured users based on MAC and IP addresses. This information is crucial for comparing against new data received during authentication. Authentication messages for EAPOL are forwarded to a RADIUS server, which validates the credentials against a common user database, typically an Active Directory.

Upon successful authentication, the system assigns an IP address to the specific MAC address by inserting a record in a DHCP server. This procedure ensures that the assigned IP address corresponds to a specific MAC address and is associated with a specific port. The system then generates User-Specific Access Control Lists tied to a particular port by requesting user groups for the specific username from the Active Directory via LDAP. The fundamental idea is that users in the same group as a specific server should also have access to that server. For instance, if the user *Ben* is a member of the *Mail* group, to which our Mailserver also belongs, the system creates ACLs permitting this specific traffic. Consequently, a whitelist is established to allow this connection while blocking all other traffic.

The system can identify users labeled as servers, which differ from standard clients, through a unique identifier group

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

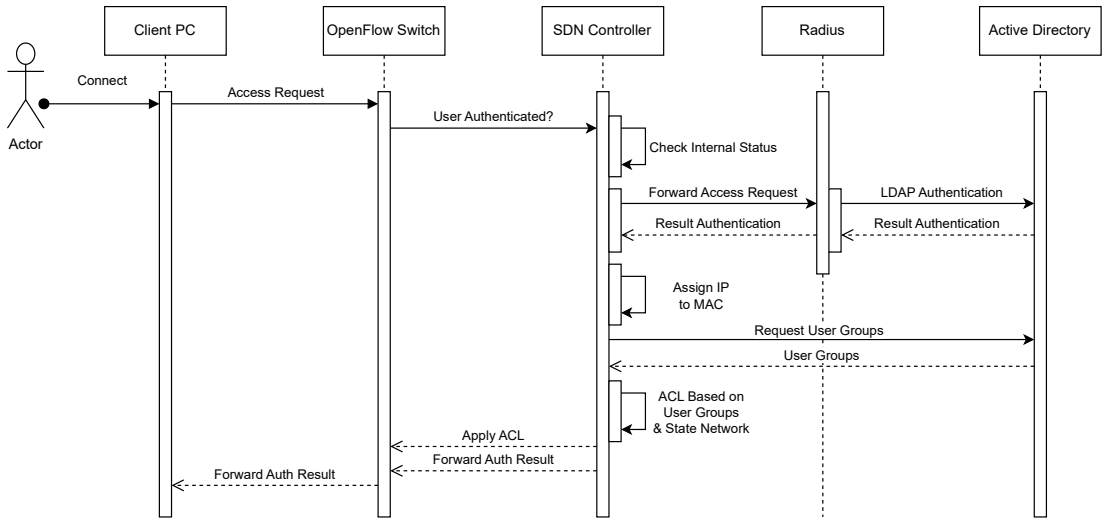


Figure 1. Dynamic ACL Adaptation based on Authentication Events

assigned explicitly to servers. The same concept could also be implemented using specific roles depending on existing usage in a company network, but in our case, we focused on groups. Thus, any user belonging to this shared group is able to establish a connection with the designated server. A port scan is conducted for servers to identify open ports and protocols. This information is linked to the user group of the server. For clients, the system constructs ACLs based on user groups and existing database information about servers, ensuring only communication between the user's MAC and IP address and the server's IP address and port. Since only the port is required for enforcing ACLs, the mechanism naturally extends to multiple SDN switches, as the ACL is bound to a port on a switch rather than a specific device.

Administrators can create templates for specific scenarios, such as restricting SSH access to administrators only. For example, port 22 can be explicitly bound to the *Administrator* group. These templates are scanned before actual ACL generation, with higher priority than user roles and dynamically discovered ports. Templates are also essential for managing Internet traffic, with administrators defining routing rules that cannot be derived automatically from database information.

Since updates of active users and checks for new ports on servers occur periodically (once a day by default, configurable by administrators), the system maintains a minimal ACL set and removes unused entries if a host is no longer connected, thereby improving efficiency [36].

C. Dynamic Authentication Events and IDS Integration

In addition to static ACL generation, the system accounts for dynamic authentication events. An LDAP proxy continuously monitors authentication activities to detect suspicious behavior, such as repeated failed login attempts for a particular service. Inspired by ALERT-ID [12], our approach extends the concept by directly adapting the network configuration when malicious behavior is observed.

Suspicious traffic is identified based on configurable thresholds. For example, an alarm is triggered when more than fifty packets are sent to invalid IP addresses, when communication attempts are made with five or more distinct blocked IPs (a common indicator of scanning activity), or when repeated connection attempts target sensitive ports such as SSH or non-standard high ports. When such conditions are met, the SDN Controller modifies OpenFlow rules to redirect all traffic from the suspicious user through the integrated Intrusion Detection System (IDS). This redirection applies to both internal and external flows, enabling centralized inspection of potentially malicious activity.

The IDS, implemented with Snort [37] in our prototype, evaluates the forwarded traffic and raises alarms that are logged in a structured JSON format. Depending on the severity of the alarm, different mitigation strategies are applied. For high- and medium-priority alarms, the system immediately updates ACLs to block the offending traffic and sends an alert to the administrator. For low-priority alarms, ACLs remain unchanged, but the administrator is notified so that the event can be investigated further.

This integration of threshold-based detection, real-time rule updates, and IDS alarms provides a layered defense mechanism. It ensures that suspicious activity is promptly identified and that the system can react adaptively, either by isolating malicious traffic or by escalating alerts to administrators.

D. Quantitative Analysis of Access Control Lists

We rely on a quantitative understanding of ACLs to evaluate the system's complexity. Using ACL counts as a metric is consistent with prior SDN research, where the number of ACL policies is directly linked to controller processing delay and scalability [36]. The ACL count is determined based on user, group, and port configurations, providing insights into the scale and complexity of the access control mechanisms.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

- **Number of ACLs per User** (N_u): This metric quantifies the number of ACL entries associated with each user. It is calculated by summing the ports across all groups a user belongs to, given by

$$N_{u_i} = \sum_{g=1}^{G_i} P_{g_i} \tag{1}$$

where P_{g_i} is the number of ports for group g for user i and G_i is the total number of groups for user i.

- **Global Number of ACLs** (N_{global}): The total number of ACLs across the network reflects the overall complexity of access control. It is computed as

$$N_{global} = \sum_{i=1}^{U} N_{u_i} \tag{2}$$

where U represents the total number of users, and N_{u_i} is the number of ACLs for user i.

- **Number of ACLs per Switch** (N_s): Understanding the ACL count for each switch helps in optimizing access control at the local level. This metric is determined by

$$N_s = \sum_{i \in S} N_{u_i} \tag{3}$$

where S is the set of users connected to the switch, and N_{u_i} represents the number of ACLs for user i in the set S.

These metrics not only provide a clear measure of system complexity but also form the basis for comparing different security configurations in the subsequent evaluation.

The subsequent section introduces the formal security model and threat analysis, establishing the assumptions and adversary capabilities before moving to implementation details.

V. Formal Security Model and Threat Analysis

This section formalizes the security objectives, assumptions, and threats addressed by the proposed system. We begin by outlining the intended security goals and the structural system model from a security standpoint. This is followed by trust assumptions, a detailed adversary model, and a threat analysis. We then extend the analysis to account for Intrusion Detection System (IDS) integration, before reflecting on limitations and directions for future work.

A. Security Objectives

The proposed system aims to strengthen network security through several concrete objectives. First, authenticated network access ensures that only verified users and devices may participate in network communication. Second, fine-grained access control is enforced via dynamically generated Access Control Lists (ACLs) based on user group membership, which are centrally managed through Active Directory. The system introduces accountability by logging authentication attempts and access control decisions, enabling forensic analysis. A key focus lies in dynamic response to emerging threats: repeated authentication failures trigger ACL updates, while only client traffic that violates an ACL is redirected to the IDS. Adhering to the principle of least privilege, clients are granted access only to services required by their role. Finally, the system ensures that security measures balance confidentiality and integrity with availability, so that malicious clients can be isolated without impairing legitimate traffic.

B. System Model

From a security perspective, the system model comprises multiple entities and communication interfaces. Clients represent end-user devices that must authenticate before being granted access to the network. Servers provide services such as mail or GitLab and are associated with defined user groups. The authenticator component is realized through an OpenFlow switch that enforces port-based access control using IEEE 802.1X. The authentication backend consists of a RADIUS server and an Active Directory instance that validates credentials and provides group membership information. A logically centralized SDN Controller orchestrates ACL generation and enforcement, while an LDAP proxy monitors authentication behavior. Suspicious traffic can be mirrored to a monitoring application or redirected through an IDS for deeper inspection.

All communication between components is assumed to be secured using encrypted and authenticated channels. EAPOL messages are transmitted between clients and the authenticator, TLS is used between the SDN Controller and the OpenFlow switch, and LDAP traffic is protected when transiting to the directory server. This design confines the trust boundary to a minimal set of components.

An architectural overview of these entities and their interactions is shown in Figure 2. While this section describes the model from a security standpoint, the detailed technical implementation is presented in Section VI.

C. Trust Assumptions

Several trust assumptions underpin the security guarantees of the system. The SDN Controller is assumed to be secure and uncompromised, as it orchestrates all access policies. The authentication server is assumed to correctly validate credentials and return accurate group memberships. The underlying user database, such as Active Directory, is assumed to provide accurate and up-to-date user and group information. Communication channels are presumed to be encrypted and authenticated to prevent interception or tampering. Switch firmware and enforcement logic are assumed to function correctly and reliably execute ACL rules. IDS components such as Snort are assumed to operate as specified, correctly classifying traffic according to known signatures. Finally, administrator-defined templates are considered trustworthy and free from malicious intent.

D. Threat Model

The system accounts for both external and internal adversaries. External adversaries may attempt to gain unauthorized

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

TABLE I. Threats, Vulnerabilities, and Mitigations

Threat	Vulnerability	Impact	Mitigation
Unauthorized access by unauthenticated device	Absence of port-level control	High	Port-based authentication with IEEE 802.1X
Brute-force or credential stuffing attacks	No rate limiting on login attempts	Medium	LDAP proxy monitors failures; ACLs updated dynamically
Privilege escalation by authenticated users	No role-based access control	High	ACLs based on user groups and administrator templates
MAC/IP spoofing	Identities not bound to MAC/IP	High	IP bound to authenticated MAC and port
Controller compromise	Centralized SDN control	Critical	Controller hardening, TLS restriction, redundancy
ACL evasion or bypass	Misconfigured or permissive rules	Medium	Automatically generated fine-grained ACLs
IDS overload	Indiscriminate traffic mirroring	Medium	Two-stage detection: monitoring threshold before IDS redirection

access without valid credentials or employ brute-force methods to compromise legitimate accounts. Internal adversaries include compromised users seeking to escalate privileges or move across the network.

Attackers are assumed capable of passively observing or actively injecting traffic. They may attempt to impersonate other users via spoofed MAC or IP addresses, perform credential stuffing, or exploit misconfigurations. Their primary goals include bypassing authentication, accessing restricted services, evading detection, and maintaining persistence. In addition, adversaries may deliberately generate high volumes of suspicious traffic to overwhelm the IDS or exploit delays between detection and mitigation.

E. Threat Analysis and Mitigation

Table I summarizes key threats addressed by the system. The classification follows established SDN threat taxonomies that group attacks into spoofing, unauthorized access, controller compromise, and policy manipulation, as discussed for example by Sharma and Tyagi [19] and Farooq et al. [29]. Each entry outlines the exploited vulnerability, its impact, and the mitigation strategy implemented. The system integrates multiple layers of defense, ranging from preventive mechanisms such as 802.1X-based port authentication to reactive measures like dynamic ACL updates and IDS redirection. Together, these mechanisms minimize attack surfaces, detect anomalous behavior, and respond swiftly to emerging threats.

F. Unauthorized Network Access

Multiple attack scenarios fall under unauthorized access. Privilege escalation occurs when users attempt to access services beyond their roles, such as SSH to administrative servers. ACL templates restrict such access to authorized groups. Brute-force or credential stuffing attacks target services like GitLab, attempting logins with common credentials. These are detected by counting failed authentication events via the LDAP Proxy and mitigated by blocking the offending client or redirecting its traffic through the IDS. Spoofing is thwarted by binding IP addresses to authenticated MAC addresses and ports, preventing impersonation.

An additional concern is compromise of the authentication backend, such as the LDAP server. While part of the trusted computing base, a compromise could undermine group-based ACL generation. Evasion attempts, such as behaving benignly during authentication but launching attacks later, are addressed by behavior-based detection. In such cases, traffic can be mirrored or redirected to the IDS for deeper inspection, ensuring continuous protection.

G. Security Limitations

While the proposed system improves security and reduces administrative overhead, certain limitations remain. First, the architecture relies on a trusted and centralized controller, which represents a single point of failure. The trust assumptions are static and do not verify runtime integrity, leaving the system exposed if a core component is compromised. Second, ACLs are restricted to header-based inspection. This minimizes performance overhead but prevents detection of application-layer threats. The IDS redirection strategy also operates in a reactive, threshold-based manner, which may delay detection of more adaptive attacks. A further limitation concerns reliance on IEEE 802.1X. Many legacy or IoT devices do not support this protocol and must be handled through weaker fallback mechanisms such as VLAN isolation or MAC-based controls, reducing the uniformity of the security model. Moreover, the system inherently depends on the correctness of the central user database. Incorrect group assignments or outdated entries immediately affect ACL generation, as no secondary validation layer is present. The current evaluation also remains qualitative. Metrics such as detection latency, mitigation time, or ACL update overhead have not been quantified and should be explored in future work.

Finally, while the IDS integration demonstrates feasibility, the accuracy of detection has not been evaluated. False positives or false negatives were not measured, and the overall impact of IDS misclassification on network behavior remains unexamined.

In summary, the formal model and threat analysis establish the security guarantees and limitations of the proposed system. Having defined these foundations, we now turn to the implementation, where the architecture is realized and integrated into a working prototype.

VI. Implementation

Following the conceptual framework outlined in Section IV, the practical implementation of the Port Access Control system

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

integrated various components. The goal was to create a working prototype that demonstrates the feasibility of fine-grained access control and prepares the ground for the evaluation. The design in Figure 2 presents how different parts work together.

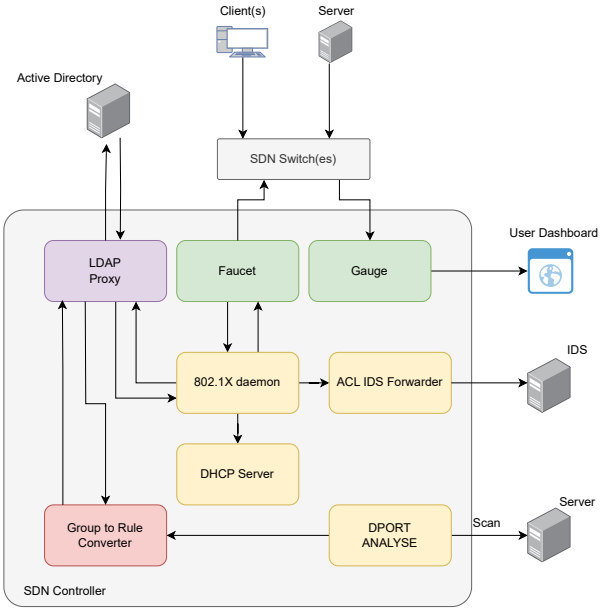


Figure 2. Architecture of the SDN Controller

We chose the Faucet SDN Controller [38] for our prototype, which uses Ryu [39] in the backend and Gauge to view events on the switch. It has the considerable advantage that the rules are defined in YAML (YAML Ain't Markup Language). One significant advantage of this architecture is that these files are human-readable and easy to understand. The initial setup of the OpenFlow switch contains only port information and requires authentication before connecting to the network. Furthermore, specific default rules, such as special treatment for the SDN Controller and the Active Directory, were specified beforehand, as these settings are essential when configuring a new network. We used the 802.1X daemon Chewie [38] as a starting point, and then it was heavily adapted to obtain user groups via a simple LDAP proxy. A second service called *Group to Rule* applies the ACLs as discussed in Section IV. An example rule can be found in Figure 3. It shows the resulting rule with a defined protocol, port, source MAC and IP address, and destination IP address. Since this is directly applied to the port, no other traffic can pass the OpenFlow switch port.

A Python script that searches for UDP and TCP ports on the server provides the open ports needed to craft the ACLs. It then saves this information into a database with the corresponding MAC address and IP address. One problem is that the server does not directly have an IP address when we try to scan it. We must wait until the IP address is handed over via the DHCP server to start scanning. Therefore, for a server, the ACLs can only be applied later on and not directly, which is

```
acls:
  mac_whitelist_user_ben:
    - rule:
        dl_type: 0x800       # ipv4
        nw_proto: 6          # tcp
        tcp_dst: 80          # port
        eth_src:  32:90:43:57:f2:01
        ipv4_src: 192.168.0.1
        ipv4_dst: 192.168.0.9
        actions:
            allow: True
    - rule:
            actions:
                allow: False
                mirror: 3
```

Figure 3. FAUCET ACL Configuration

not a problem since the default rules still block all access to the network and only DHCP is then allowed to obtain an IP address. A simple folder structure was defined for the templates where an administrator can place templates for groups and specific ports as well as initial network operations such as DHCP and DNS.

The dynamic adaptation of ACLs depending on authentication data is realized via the LDAP Proxy. All servers in the network attempt to authenticate their users via LDAP bind requests to the proxy, which then forwards them to the Active Directory. This setup allows us to track whether authentication was successful. We implemented a counter for each user with a configurable threshold (six failed attempts by default), which resets after a timeout period, similar to the mechanism described in ALERT-ID [12]. A lightweight monitoring script parses the authentication logs and forwards suspicious events to the SDN Controller.

For demonstration purposes, any traffic that would normally be blocked is mirrored to a dedicated port where a simple Python counter application is connected. This service maintains per-host violation counts (based on MAC and IP addresses) and alerts the SDN Controller once a threshold is exceeded. When thresholds are crossed, such as repeated attempts to access non-standard ports, scanning activities against multiple blocked IPs, or persistent connections to invalid destinations, the controller updates the OpenFlow configuration and enforces stricter handling of the suspicious host.

At this point, the IDS integration becomes active. In our prototype we used Snort [37], which inspects all traffic from hosts that exceeded the violation threshold. Critical alarms, such as confirmed scanning or exploitation attempts, immediately trigger ACL updates that block the corresponding host and alert the administrator. Medium-severity alarms also result in blocking, but the administrator receives a detailed event log for further analysis. Low-severity alarms are logged and reported without enforcing automatic blocking, leaving the final decision to the administrator.

This two-stage design ensures that the IDS is not overloaded by benign or low-level violations. Faucet's mirroring capability

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

TABLE II. ACL CONFIGURATION FOR FIVE CONNECTED CLIENTS

OpenFlow Port	Source MAC	Source IP	Group	Destination IP	Destination Port	Description
3	1C:69:7A:6D:C6:27	192.168.11.11	mail	192.168.11.101	25, 993, 995	Mailserver
3	1C:69:7A:6D:C6:27	192.168.11.11	gitlab	192.168.11.102	22, 80, 443	GitLab
4	1C:69:7A:43:7C:12	192.168.11.12	mail	192.168.11.101	25, 993, 995	Mailserver
5	1C:69:7A:6D:C8:B0	192.168.11.13	mail	192.168.11.101	25, 993, 995	Mailserver
5	1C:69:7A:6D:C8:B0	192.168.11.13	gitlab	192.168.11.102	22, 80, 443	GitLab
6	1C:69:7A:6D:C7:EE	192.168.11.14	mail	192.168.11.101	25, 993, 995	Mailserver
7	1C:69:7A:6D:C8:16	192.168.11.15	mail	192.168.11.101	25, 993, 995	Mailserver

enables efficient pre-filtering via the counter application, and only persistent or severe violations cause full redirection through the IDS. As a result, IDS integration is not a standalone add-on but an embedded part of the access control pipeline managed by the SDN Controller.

The implementation phase reaffirmed the proposed system's potential to enhance network security through fine-grained access controls and adaptive IDS support. At the same time, it highlighted the complexities of managing an extensive rule set, especially in larger networks where automated rule aggregation and optimization become critical.

VII. Evaluation

In this evaluation chapter, we begin with a detailed examination of the technical aspects of our experimental setup, laying the groundwork for a thorough assessment. We then delve into the feasibility of the proposed system, followed by a comparative analysis of its efficiency and complexity against existing systems. This analysis sets the stage for a nuanced discussion synthesizing our findings and their implications.

A. Experimental Conditions

Our experimental setup was designed to mirror a realistic environment consisting of multiple physical PCs and servers to simulate a conventional corporate network infrastructure. The network configuration included five Windows clients. We assigned the clients to different user groups in the Active Directory. The configuration of each client and its connected port can be found in Table III. In setting up our experiment, we went with a mix that one would typically find in an office: a mail server for emails and a GitLab instance for the devs to collaborate on code. This way, we could see how different roles, like developers needing GitLab and managers relying on emails, would interact with the system. It is a practical approach that helps us understand how our setup performs in a real-world scenario.

TABLE III. CLIENTS IN THE NETWORK

Client Name	Port	Source MAC	Groups
Client1	3	1C:69:7A:6D:C6:27	mail, gitlab
Client2	4	1C:69:7A:43:7C:12	mail
Client3	5	1C:69:7A:6D:C8:B0	mail, gitlab
Client4	6	1C:69:7A:6D:C7:EE	mail
Client5	7	1C:69:7A:6D:C8:16	mail

At the core of our network was an Active Directory on a Windows Server 2019, connected to a dedicated port at the OpenFlow switch. This switch was a Linux PC running Ubuntu 22.10, with an Intel(R) Core(TM) i7-8700 CPU supporting OpenFlow protocol version 1.3.

This detailed setup provides a solid foundation for evaluating the system's feasibility, performance, and complexity.

B. Feasibility

The project aimed to demonstrate the feasibility of such a system and highlight its advantages. Therefore, we conducted multiple experiments to verify the system's operability to achieve this. In our earlier conference paper [1], we presented two experiments demonstrating feasibility: initial connection and ACL enforcement, and failed login handling. In this extended version, we add a third experiment that evaluates the integration of an IDS for redirecting suspicious traffic. Together, these three experiments provide a comprehensive assessment of the system.

1) Experiment 1: Connection to the network: In our initial experiment, we aimed to verify the functionality of the system's initial configuration and the practical application of Access Control Lists. We began by attempting to connect a server to the network. Initially, all packets except EAP packets were blocked, preventing any network connection without proper authentication. To facilitate authentication, we configured the server's wpa_supplicant with EAP after setting up a dedicated user account in the Active Directory for the server, marked by the "server" group identifier, to distinguish it as such. Additionally, the server was assigned to the "mail" group to define its access rights. The authentication process utilized standard Username and Password credentials defined within the Active Directory.

Upon initiating these configurations, we observed successful authentication, followed by the server obtaining an IP address via DHCP. The IP address assignment was managed by the SDN Controller, ensuring the server's connectivity post-authentication. We then proceeded with a port scan, which was feasible only after the OpenFlow switch recognized the server's IP, confirming that the server was operational. The procedure was repeated for the second GitLab server.

Subsequently, we connected a client machine to the network. Like the server setup, this client was denied network access until authentication credentials were provided. After authentication, the SDN Controller dynamically generated ACLs based on the client's group memberships.

For example, the first client, identified as a developer, was granted access to both the mail server and GitLab, as reflected in the applied ACLs (refer to Table II, lines 1 and 2). This

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

access control was strictly enforced, with all unauthorized traffic being blocked at the port level based on the authenticated source MAC address and specified port. In contrast, a client identified as a manager, and thus only requiring access to the mail server, demonstrated restricted network access in line with their role (refer to Table II, line 3). Attempts to access GitLab by this client were blocked, illustrating the ACLs' role-based access control. After connecting all clients, each client and port results can be found in Table II.

Upon issuing a logoff command to the RADIUS server, all associated ACLs were cleared, reverting the system to its default state of blocking all traffic from the disconnected client. Logging in with a different username on the same PC triggered a reallocation of ACLs, aligning with the new user's access rights. This experiment demonstrated the feasibility of initially creating and effectively applying ACLs within our network environment.

2) Experiment 2: Failed Logins: In the second experiment, we tested failed login attempts to evaluate the system's response mechanisms. This test simulated incorrect authentication attempts on the GitLab server to observe the system's reaction.

The experiment began with a series of failed login attempts, with each unsuccessful attempt logged by the SDN Controller. After the sixth failed attempt, the SDN Controller adjusted the ACLs, cutting off the client's access to the server and other network components. An alert was automatically sent to the network administrator, who could either restore the client's access after a successful re-authentication or suspend the client for further investigation.

Additionally, we tested the network's traffic mirroring feature. In this part of the experiment, despite multiple failed login attempts, the client was not disconnected from the network. Instead, the client's traffic was mirrored to a specific port on an OpenFlow switch. This procedure was verified using tcpdump to confirm that the traffic mirroring was functioning as intended, without the integration of an IDS, since this was not in the scope of the experiment.

3) Experiment 3: Redirecting Suspicious Traffic to IDS: The third experiment evaluated the system's ability to identify suspicious clients and to enforce dynamic redirection through the IDS. The scenario consisted of the following key steps:

1) Introduce a malicious host that violated predefined policies.
2) Detect repeated violations until defined thresholds are exceeded.
3) Redirect all traffic from the host through the IDS.
4) Generate malicious traffic samples to trigger Snort rules.
5) Apply mitigation actions depending on alert severity.

To simulate malicious activity in detail, we introduced a host that attempted to connect to non-standard ports, repeatedly accessed invalid IP addresses, and performed network scanning across multiple targets. Each of these actions incremented the violation counter maintained by the monitoring application, which received mirrored traffic from the OpenFlow switch.

Once the violation thresholds were exceeded, the SDN Controller updated the Faucet rules for the offending host. Instead of simply dropping further packets, the rules were rewritten so that all subsequent traffic from the host was forwarded through the IDS. In our prototype, this redirection was achieved by replacing the `mirror` action with an `output` action that directed the traffic to the IDS ingress port.

To further assess the system's responsiveness, we generated malicious traffic using publicly available attack samples and network scanning tools in order to trigger common Snort rules. Snort [37], deployed as the IDS in our setup, successfully identified the injected traffic and produced alarms. Depending on the severity of the alert:

- Critical alerts caused the controller to immediately block the host and notify the administrator.
- Medium alerts also resulted in blocking but generated detailed logs for review.
- Low alerts were logged and reported without automatic blocking.

Figure 4 shows the CPU usage of the Snort instance during a 600-second evaluation run. The annotated timeline of events is as follows:

- **0s:** Start of experiment, baseline traffic only.
- **152s:** First client exceeded thresholds and was redirected to IDS.
- **307s:** A second malicious client was added, increasing IDS load.
- **531s:** Snort raised a critical alert for the first client; the SDN Controller blocked it, reducing IDS load.

This progression highlights how each additional client measurably increased CPU usage on the IDS, while removing a client reduced load again. CPU utilization is strongly correlated with traffic volume, consistent with prior measurements by Lukaseder et al. [40]. Although our test traffic was deliberately crafted to repeatedly trigger Snort rules (leading to a higher per-client load than in production), the experiment demonstrates that selective redirection scales with the number of suspicious hosts while preventing the IDS from being overwhelmed with benign traffic.

C. Complexity and Efficiency

To evaluate our system's complexity and OpenFlow rule management capability, we compared it against other SDN security methods by examining the number of OpenFlow rules in different scenarios. Our analysis included a baseline scenario without ACLs, a basic ACL setup, and scenarios involving VLANs. The scenario with Basic ACLs has only rules for direct IP access. That means we only specify that user X can access server Y without further defining which ports or protocols. The VLAN example does not have any specific ACLs. It splits the users into two groups, usually some kind of department in a corporate network. This option has the disadvantage of allowing clients from the same department to

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

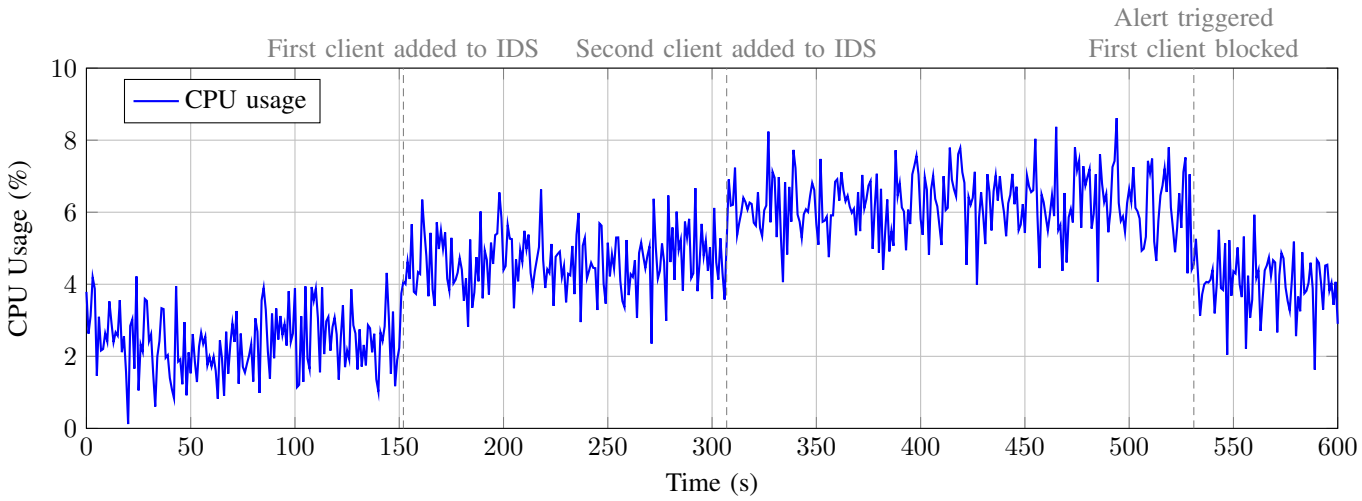


Figure 4. CPU usage over time with key IDS events highlighted.

communicate, which does not prevent malware from spreading.

Table IV summarizes the OpenFlow rule count for each scenario. As observed, the number of OpenFlow rules directly reflects the count of Faucet ACLs. As discussed in Equation (2), the number of ACLs will increase linearly with the number of users. The dynamic ACL configuration, while more complex, demonstrates the system's flexibility and responsiveness to network changes without significantly impacting performance.

TABLE IV. RULE COUNT COMPARISON

Scenario	Faucet ACLs	OpenFlow Rules
No ACLs	0	27
Basic ACLs	8	67
VLANs	N/A	67
Dynamic ACLs	32	91

D. Discussion

In discussing the outcomes and implications of our experiments, it is essential to consider both the implemented system's strengths and potential challenges. To offer a comprehensive understanding of our system's enhanced performance and its innovative approach to network security and management, we performed an extensive comparison across several key metrics, including security level, scalability, and manageability. This comparison, detailed in Table V, is based on empirical data from ACL number analytics, a comparative analysis of system architectures, and their maintenance needs, highlighting our proposed system's superiority in terms of security, scalability, and ease of management.

The introduction of automated, fine-grained whitelist ACLs represents a significant step forward in network security management. The configuration process substantially decreases administrative overhead and mitigates the risk of human error, which is prevalent in manual configurations. A crucial advantage of this approach is the centralization of security decisions, such as access rights, in a singular user database. This consolidation ensures that modifications to access rights are uniformly applied across the network and all services utilizing this common user database, thereby enhancing consistency and security within the system. As demonstrated in Experiment 1, the automation of ACL configuration significantly reduced administrative overhead since all needed restrictions are applied individually for each client without the need for additional adaptation by the administrator. In contrast, this centralized, automated approach ensures that only authenticated users and their associated MAC addresses are actively maintained in the system, limiting access to authorized entities and inherently reducing the risk of unauthorized access. Another significant benefit of our approach is its scalability and ease of integration across multiple switches without additional overhead. Since ACLs and clients are bound to specific ports and not to the physical switches themselves, our system can seamlessly scale to accommodate an extensive network infrastructure with multiple SDN switches. ACLs are applied uniquely to each switch, as delineated in Equation (3), ensuring efficient and tailored security measures are in place, irrespective of the size or complexity of the network.

However, this automation and simplification come at the cost of increased complexity due to the more significant number of ACLs required to maintain fine-grained control over network access. The number of ACLs does not directly impact the system's performance since it only inspects the TCP header to minimize performance impact, compared to, for example, complex rules that inspect the TCP payload. According to Cabrera et al. [41], the time required to check the payload is, on average, 4.5 times longer than that required for header checks. Therefore, even with many ACL rules, the focus on header information ensures minimal impact on network throughput, as even a single ACL with a TCP header rule necessitates the inspection of every packet. One drawback is that we need to prepare the clients and the server to perform

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

TABLE V. COMPARISON OF NETWORK SECURITY AND MANAGEMENT APPROACHES

Metric	**No ACLs**	**Basic ACLs**	**VLANs**	**Resonance**[16]	**ACL Based on X812**[18]	**Proposed System**
Security Level	Low	Medium	Medium	High	Very High	Very High
Port Security	None	None	None	None	Full	Full
Performance Impact	Low	Medium	Medium	Moderate	Moderate	Moderate
Scalability	High	Moderate	Good	Moderate	Moderate	Excellent
Manageability	Easy	Moderate	Moderate	Moderate	Moderate	Easy
Centralization	None	Low	Low	Medium	Medium	High
Flexibility	Low	Moderate	Low	Very High	Very High	High
Cost Efficiency	High	Moderate	Moderate	Low	Moderate	High
Integration Capability	Seamless	Moderate	Challenging	High	High	Low
Resilience	Low	Medium	Medium	High	High	High
Automation & Dynamic Response	None	None	None	Semi-Automated	Semi-Automated	Fully Automated
ACLs based on Authentication	None	None	None	None	None	Supported

an 802.1X authentication.

The experiments involving IDS redirection further confirmed the system's scalability and efficiency. As shown in Figure 4, the CPU usage of Snort increased proportionally with the number of malicious clients redirected, but decreased again once clients were blocked by the SDN Controller. This demonstrates that the selective forwarding mechanism ensures the IDS is only engaged for traffic that truly warrants inspection, preventing overload and allowing for efficient use of IDS resources. Such integration provides a stronger security posture without sacrificing performance for benign traffic, which continues to be handled exclusively by the ACL framework.

One of the more critical considerations is the system's approach to handling failed login attempts, as demonstrated in Experiment 2. Completely blocking access after a series of incorrect credentials can safeguard against brute-force attacks but also pose a risk to business continuity. For instance, automated tools using outdated credentials could inadvertently trigger these security measures, leading to unnecessary disruptions. This aspect of the system necessitates a careful balance between maintaining robust security and ensuring uninterrupted business operations.

Integrating traffic mirroring for suspicious hosts presents a nuanced approach to enhancing security monitoring without overloading the network or the IDS. By selectively mirroring traffic from potentially compromised hosts, the system can focus on analyzing and responding to genuine threats, improving overall security efficiency. This concept aligns with the approach discussed in [42], which proposes a clustering-based flow grouping scheme that assigns network flows to various IDSs based on routing information and flow data rates, aiming to optimize the load distribution among IDSs and enhance attack detection capabilities.

VIII. Conclusion and Future Work

In conclusion, the proposed system presents a straightforward yet powerful framework that significantly enhances network security by enforcing fine-grained access control rules. By leveraging a common user database, such as Active Directory, and binding access controls to specific MAC addresses, the system ensures that only authenticated users can access network ports, thereby establishing a robust security posture.

A key contribution of this work is the integration of an Intrusion Detection System into the access control framework. Rather than deploying the IDS inline for all traffic, our approach mirrors suspicious flows to a monitoring application and selectively redirects offending hosts once violation thresholds are reached. This two-stage mechanism ensures that benign traffic is not subjected to costly IDS inspection, while malicious clients are efficiently isolated and analyzed. As demonstrated in our evaluation, this selective redirection prevents IDS overload, allows scalable operation, and improves the responsiveness of the SDN Controller to detected threats.

Although the results demonstrate the feasibility of the proposed architecture, several limitations remain. The evaluation was conducted in a controlled laboratory environment, which restricts the generality of the findings. The approach relies on a trusted and centralized SDN Controller as well as correct group assignments in the user database, and these assumptions may not hold in all deployments. IDS behavior was not evaluated with respect to false positives or false negatives, and performance under larger or more dynamic network conditions remains open for further study.

Future work should extend the evaluation and broaden the applicability of the system. A first step is a more detailed performance study, including metrics such as ACL update latency, controller processing times, IDS alert latency, and the behavior of the system under higher client loads. The IDS integration should also be assessed in a more systematic way, for example by measuring false-positive and false-negative rates and by comparing different inspection strategies. Additional experiments with larger deployments would help validate scalability, while integration with advanced monitoring systems such as Zeek [43] could widen the visibility of network flows and enable deeper analyses. Another line of research concerns the interaction between dynamically generated ACLs, existing firewall rules, and higher-level network policies. Incorporating policy conflict detection and resolution mechanisms from related SDN research could further strengthen robustness in hybrid environments. Finally, evaluating unknown attack classes is orthogonal to the access-control-focused design of this system but may complement future studies that investigate more general SDN-based threat detection frameworks.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

ACKNOWLEDGMENT

The authors would like to thank Prof. Hiroki Takakura for useful advice. This work was partially supported by JSPS KAKENHI Grant Number JP23K28086 and JP24K14959.

REFERENCES

[1] F. Grießer, A. Shinoda, H. Hasegawa, and H. Shimada, "Automating SDN-ACLs with user groups and authentication events," in *Proceedings of the Thirteenth International Conference on Networks (ICN 2024)*. Barcelona, Spain: IARIA, May 2024, pp. 5–12.

[2] H. Zhou, C. Wu, M. Jiang, B. Zhou, W. Gao, T. Pan, and M. Huang, "Evolving defense mechanism for future network security," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 45–51, 2015.

[3] S. T. Ali, V. Sivaraman, A. Radford, and S. Jha, "A survey of securing networks using software defined networking," *IEEE Transactions on Reliability*, vol. 64, no. 3, pp. 1086–1097, 2015.

[4] F. Bannour, S. Souihi, and A. Mellouk, "Distributed SDN control: Survey, taxonomy, and challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 333–354, 2017.

[5] P. Wadhwani, "Software defined networking (sdn) market report, 2024–2032," Market Research Report, Global Market Insights Inc., Report ID: GMI2395, 2024, accessed: 2025-12-09. [Online]. Available: https://www.gminsights.com/industry-analysis/software-defined-networking-sdn-market

[6] F. S. Ali, R. Amin, M. Majeed, and M. M. Iqbal, "Dynamic ACL Policy Implementation in Software Defined Networks," in *2022 International Conference on IT and Industrial Technologies (ICIT)*, Oct 2022, pp. 01–07.

[7] S. T. Yakasai and C. G. Guy, "FlowIdentity: Software-defined network access control," in *2015 IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. IEEE, 2015, pp. 115–120.

[8] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: taking control of the enterprise," in *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 1–12.

[9] D. M. Ferrazani Mattos and O. C. M. B. Duarte, "AuthFlow: authentication and access control mechanism for software defined networking," *Annals of Telecommunications*, vol. 71, pp. 607–615, 2016.

[10] T. Xing, D. Huang, L. Xu, C.-J. Chung, and P. Khatkar, "SnortFlow: A OpenFlow-Based Intrusion Prevention System in Cloud Environment," in *2013 Second GENI Research and Educational Experiment Workshop*, March 2013, pp. 89–92.

[11] A. Le, P. Dinh, H. Le, and N. C. Tran, "Flexible Network-Based Intrusion Detection and Prevention System on Software-Defined Networks," in *2015 International Conference on Advanced Computing and Applications (ACOMP)*, Nov 2015, pp. 106–111.

[12] J. Chu, Z. Ge, R. Huber, P. Ji, J. Yates, and Y.-C. Yu, "ALERT-ID: analyze logs of the network element in real time for intrusion detection," in *Research in Attacks, Intrusions, and Defenses: 15th International Symposium, RAID 2012, Amsterdam, The Netherlands, September 12-14, 2012. Proceedings 15*. Springer, 2012, pp. 294–313.

[13] A. H. Janabi, T. Kanakis, and M. Johnson, "Survey: Intrusion detection system in software-defined networking," *IEEE Access*, vol. 12, pp. 164 097–164 120, 2024.

[14] B. Susilo and R. F. Sari, "Intrusion detection in software defined network using deep learning approach," in *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2021, pp. 0807–0812.

[15] B.-X. Wang, J.-L. Chen, and C.-L. Yu, "An ai-powered network threat detection system," *IEEE Access*, vol. 10, pp. 54 029–54 037, 2022.

[16] A. K. Nayak, A. Reimers, N. Feamster, and R. Clark, "Resonance: dynamic access control for enterprise networks," in *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, ser. WREN '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 11–18.

[17] V. Shah and P. Yadav, "An implementation of dot 1x for secure network access in sdn," in *2025 6th International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*. IEEE, 2025, pp. 1264–1269.

[18] B. J. C. de A. Martins, D. M. Mattos, N. C. Fernandes, D. Muchaluat-Saade, A. B. Vieira, and E. F. Silva, "An Extensible Access Control Architecture for Software Defined Networks based on X.812," in *2019 IEEE Latin-American Conference on Communications (LATINCOM)*, 2019, pp. 1–6.

[19] P. K. Sharma and S. Tyagi, "Security enhancement in software defined networking (sdn): A threat model," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 9, 2021.

[20] S. Pradeep, Y. K. Sharma, U. K. Lilhore, S. Simaiya, A. Kumar, S. Ahuja, M. Margala, P. Chakrabarti, and T. Chakrabarti, "Developing an sdn security model (ensures) based on lightweight service path validation with batch hashing and tag verification," *Scientific Reports*, vol. 13, no. 1, p. 17381, 2023.

[21] Y. Meng, Z. Huang, G. Shen, and C. Ke, "A security policy model transformation and verification approach for software defined networking," *Computers & Security*, vol. 100, p. 102089, 2021.

[22] P. Porras, S. Shin, V. Yegneswaran, M. Fong, M. Tyson, and G. Gu, "A security enforcement kernel for openflow networks," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, 2012, pp. 121–126.

[23] H. Hu, W. Han, G.-J. Ahn, and Z. Zhao, "Flowguard:

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Building robust firewalls for software-defined networks," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, 2014, pp. 97–102.

[24] A. Khurshid, W. Zhou, M. Caesar, and P. B. Godfrey, "Veriflow: Verifying network-wide invariants in real time," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, 2012, pp. 49–54.

[25] P. Kazemian, M. Chang, H. Zeng, G. Varghese, N. McKeown, and S. Whyte, "Real time network policy checking using header space analysis," in *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, 2013, pp. 99–111.

[26] C. Prakash, J. Lee, Y. Turner, J.-M. Kang, A. Akella, S. Banerjee, C. Clark, Y. Ma, P. Sharma, and Y. Zhang, "Pga: Using graphs to express and automatically reconcile network policies," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 29–42, 2015.

[27] S. Pisharody, J. Natarajan, A. Chowdhary, A. Alshalan, and D. Huang, "Brew: A security policy analysis framework for distributed sdn-based cloud environments," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 6, pp. 1011–1025, 2017.

[28] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[29] M. S. Farooq, S. Riaz, and A. Alvi, "Security and Privacy Issues in Software-Defined Networking (SDN): A Systematic Literature Review," *Electronics*, vol. 12, no. 14, 2023.

[30] "IEEE Standard for Local and Metropolitan Area Networks–Port-Based Network Access Control," *IEEE Std 802.1X-2020 (Revision of IEEE Std 802.1X-2010 Incorporating IEEE Std 802.1Xbx-2014 and IEEE Std 802.1Xck-2018)*, pp. 1–289, 2020.

[31] B. Desmond, J. Richards, R. Allen, and A. G. Lowe-Norris, *Active Directory: Designing, Deploying, and Running Active Directory*. " O'Reilly Media, Inc.", 2008.

[32] J. Sermersheim, "Lightweight Directory Access Protocol (LDAP): The Protocol," RFC 4511, Jun. 2006.

[33] J. Vollbrecht, J. D. Carlson, L. Blunk, D. B. D. Aboba, and H. Levkowetz, "Extensible Authentication Protocol (EAP)," RFC 3748, Jun. 2004.

[34] H. Hasegawa, Y. Sato, and H. Takakura, "Construction of Secure Internal Network with Communication Classifying System Using Multiple Judgment Methods," *International Journal on Advances in Telecommunications*, vol. 13, no. 3 & 4, 2020.

[35] Y. Sato, H. Hasegawa, and H. Takakura, "Construction of Secure Internal Networks with Communication Classifying System," in *ICISSP*, 2019, pp. 552–557.

[36] M. Ali, N. Shah, and M. A. Khan Khattak, "DAI: Dynamic ACL Policy Implementation for Software-Defined Networking," in *2020 IEEE 17th International Conference on Smart Communities: Improving Quality of Life Using ICT, IoT and AI (HONET)*, Dec 2020, pp. 138–142.

[37] M. Roesch, "Snort: Lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX Conference on System Administration (LISA '99)*. USENIX Association, 1999, pp. 229–238.

[38] FaucetSDN, "Faucet," 2024, accessed: 2025-11-20. [Online]. Available: https://github.com/faucetsdn/faucet

[39] Ryu SDN Framework Community, "Ryu sdn framework," https://ryu-sdn.org/, 2024, accessed: 2025-11-20.

[40] T. Lukaseder, J. Fiedler, and F. Kargl, "Performance evaluation in high-speed networks by the example of intrusion detection," *arXiv preprint arXiv:1805.11407*, 2018.

[41] J. B. Cabrera, J. Gosar, W. Lee, and R. K. Mehra, "On the statistical distribution of processing times in network intrusion detection," in *2004 43rd IEEE Conference on Decision and Control (CDC)(IEEE Cat. No. 04CH37601)*, vol. 1. IEEE, 2004, pp. 75–80.

[42] T. Ha, S. Yoon, A. C. Risdianto, J. Kim, and H. Lim, "Suspicious flow forwarding for multiple intrusion detection systems on software-defined networks," *IEEE Network*, vol. 30, no. 6, pp. 22–27, 2016.

[43] T. Z. Project, "Zeek: Network security monitor," https://github.com/zeek/zeek, 2024, accessed: 2025-11-27.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Security-risk-mitigation Measures for Service Oriented Vehicle Diagnostics SOVD

Masaaki Miyashita
Cybersecurity Group, Connected Car Off-board Development and Operation Department of Nissan Motor Corporation
Kanagawa, Japan
e-mail: m-miyashita@mail.nissan.co.jp

Benchadi Djafer Yahia M
Cybersecurity Group, Connected Car Off-board Development and Operation Department of Nissan Motor Corporation
Kanagawa, Japan
e-mail: b-djafer@mail.nissan.co.jp

Hiroki Takakura
National Institute of Informatics
Tokyo, Japan
e-mail: takakura@nii.ac.jp

***Abstract*— A new vehicle diagnostic standard "Service Oriented Vehicle Diagnostics (SOVD)" is expected to be used for the next-generation vehicles known as Software Defined Vehicles (SDV). SOVD supports various vehicle maintenance demands, including remote diagnosis, by implementing web server function into a high-performance in-vehicle component. However, this architecture introduces additional security risks to SDV, as this web server functionality becomes a new cyberattack entry point into the vehicle. In this paper, we present several security-risk-mitigation measures for such systems, extending our previous work. Specifically, we propose multi-layered defense measures including physical and logical isolation (Zone Separation) of the web server software from security-critical software modules and in-vehicle HMI-based authorization for critical diagnostic privileges. We conclude that these additional security measures significantly reduce the feasibility of remote cyberattacks against SOVD-based remote diagnostic systems.**

Keywords-Automotive cybersecurity; Remote diagnosis; UDS; SOVD.

I. INTRODUCTION

This work is a follow-up to our prior work "Security-risk-mitigation Measures for Automotive Remote Diagnostic Systems", published in the proceedings of SECUREWARE2024 [1]. As technology advances, the electronic systems in automobiles are becoming more intricate. These systems consist of numerous components that are connected through in-vehicle communication networks. Diagnostic systems specifically designed for vehicles are required to pinpoint any malfunction. These systems usually require a diagnostic tool to be directly connected to a dedicated connector on the vehicle and must be operated at a garage.

With wireless communication systems increasingly used in vehicles, remote diagnosis systems have become more prevalent. These services enable an operator to read diagnostic trouble codes and data logs through wireless communication. This prompts the driver to bring his/her vehicle to a garage for repairs before the trouble becomes more severe. Diagnostic communications are used not only to read such data but also to write data to in-vehicle parts, such as firmware updates and initial settings of replacement parts.

Studies have indicated that cyberattacks targeting vehicles through diagnostic communications can result in significant damage. For example, it has been demonstrated that some diagnostic Controller Area Network (CAN) messages impacted major critical vehicle control systems, such as the engine, brake, and steering systems [2]. Car theft and privacy breaches are also potential risks of cyberattacks through diagnostic communication [3].

On the other hand, European Union vehicle type approval regulation EU 2018/858 [4] Annex X requires that "Manufacturers shall provide to independent operators unrestricted, standardised and non-discriminatory access to vehicle OBD information, diagnostic and other equipment, tools including the complete references, and available downloads, of the applicable software and vehicle repair and maintenance information. Information shall be presented in an easily accessible manner in the form of machine-readable and electronically processable datasets. Independent operators shall have access to the remote diagnosis services used by manufacturers and authorised dealers and repairers.", if a vehicle has the remote diagnostic system. This requirement makes designing measures against unauthorized access complex, because their network access routes and credentials for user authentication become various.

In our previous work [1], we mainly focused on risk mitigation for conventional remote diagnostic architectures based on UDS communication. This extended study introduces a new perspective by addressing the security challenges of the emerging Service-Oriented Vehicle Diagnostics (SOVD) framework. Building on this shift, we present security-risk-mitigation measures specifically adapted to the SOVD-based remote diagnostic systems. These systems involve reading diagnostic trouble data and remote firmware-update tasks that were previously only executed at service stations. Our measures aim to reduce the potential security risks associated with these systems.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

The rest of the paper is structured as follows. In Section II, we discuss automotive diagnostic communication. Section III presents the current status and existing issues of remote diagnosis. In Section IV, we propose our security-risk-mitigation measures. In Section V, we show how to avoid constraints when implementing proposed measures in vehicle component. Section VI illustrates our prototype simulation for evaluation. Finally, we conclude our work in Section VII.

II. AUTOMOTIVE DIAGNOSTIC COMMUNICATION

The process of remote diagnosis involves the use of wireless communication between a vehicle and a diagnostic server located outside the vehicle. To diagnose the various components implemented in the vehicle, the in-vehicle wireless communication unit, which serves as the entry point to the vehicle, must communicate with other components through the in-vehicle communication network. To achieve this, it is most reasonable from a system-implementation standpoint to use the diagnostic communication protocol typically used for wired-connected diagnostic tools. While this protocol is effective for wired communication, there are security concerns when using it for wireless communication.

With this in mind, we examined the characteristics and issues of automotive diagnostic communications used in the in-vehicle network.

A. Overview of Diagnostic Communication

In 1991, the California Air Resources Board mandated the implementation of the On-Board Diagnostics (OBD) connector to standardize vehicle diagnostic communications. Today, the OBD2 connector is the industry standard interface and can use several communication protocols. CAN communication is prevalent in vehicle-embedded processors, and there is a shift towards faster diagnostic communication using Diagnostics over Internet Protocol (DoIP)-based communication with an Ethernet physical layer [5]. To address the need for faster communication and accommodate the increased complexity of automotive software, ISO14229-1 standardized the Unified Diagnostic Service (UDS) Protocol, which is now used as a standard communication protocol by many automotive companies. However, as software complexity increases, so do security concerns, as outlined in previous studies [6] and [7] on DoIP.

In 2022, ASAM (Association for Standardization of Automation and Measuring Systems) released a new vehicle diagnostic communication API (Application Programming Interface) “ASAM SOVD v1.0.0” [8] targeting the new generation vehicles with Software Defined Vehicle (SDV) architecture. This SOVD requires the vehicle architecture shown in Figure 1, because SDV requires High-Performance Computer (HPC) to have some Virtual ECUs as software components for easy upgradability of the vehicle functions. HPC is a key component of this architecture, because it hosts the SOVD server as a hub of diagnostic communication. It is one of big difference from UDS that SOVD supports the remote diagnosis as a native standard service. And SOVD also consider reusing old vehicle Electronic Control Units (ECU) with UDS protocol by CDA (Classic Diagnostic Adapter) as a communication translator between SOVD API and UDS. SOVD will be applied to many new generation vehicles with SDV architecture because it will be a new international standard ISO-17978 by the end of 2025.

B. Diagnostic Tool

Advancements in diagnostic-communication hardware and software have brought about changes in diagnostic tools used to identify failures in vehicles. Handheld terminals with basic Liquid Crystal Displays (LCDs) had been commonly used for diagnostic communication before the spread of CAN communication. However, with the increasing number of vehicles supporting diagnostic communication and the complexity of systems due to the introduction of IP communication, developing software for specialized hardware has become inefficient. Thus, it is now common to use a Personal Computer (PC) or tablet in Figure 2 as a diagnostic tool and connect it to an OBD dongle through USB, Bluetooth, wireless LAN, etc.

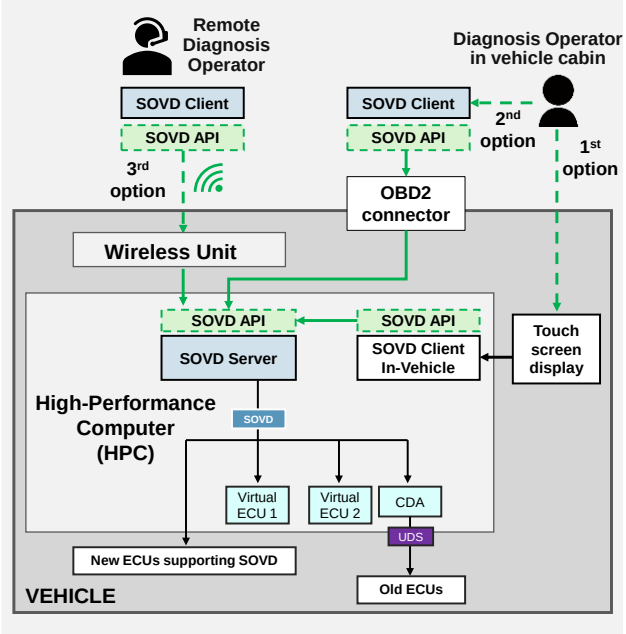


Figure 1. Example of SOVD vehicle architecture

This approach has the additional benefit of enabling developers of general diagnostic tools that support vehicles from multiple automobile companies to easily acquire diagnostic tool hardware. However, it also raises concerns

Figure 2. Diagnostic tools using PC/Tablet

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

that these devices, which are essentially PCs and tablets with network connectivity as standard equipment, could be used as gateways for attackers to intrude into vehicles. Since diagnostic communication protocols are standardized and diagnostic tools and software can be purchased inexpensively, attackers can find vulnerabilities through reverse engineering.

SOVD will also change the diagnostic tool. As Figure 1 shows, when a vehicle has a touch screen display in the vehicle cabin, SOVD can provide first option, “In-vehicle client” using the display to realize “Diagnostic tool-less” operation. A diagnosis operator can use access through the OBD2 connector as second option when the touch screen display is unavailable. When an operator requires access from a remote location, SOVD provides third option through the wireless interfaces. In this second and third options, any consumer devices (e.g., smartphones, tablets, personal computers) can access in-vehicle SOVD server by various web browsers, because the SOVD server uses REST (Representational State Transfer) API over HTTPS (Hypertext Transfer Protocol Secure) to communicate with the clients instead of the UDS which was used by old diagnostic tools. These options will give the diagnostic operator better flexibility than using the diagnostic tools, but requires stronger security measures, because implementing web server-client systems into the vehicle must bring new vulnerabilities as same as the information systems. In order to mitigate risks from old diagnostic tools, SOVD enforces vehicle-side authentication and authorization for all critical services. Conventional diagnostic tools in UDS system store ECU service information locally in the device and can be reverse engineered by an attacker. With SOVD, this risk is mitigated due to the diagnostic tool no longer contains vehicle-specific data or applications. All service information processes for ECU modification or software updates are handled by the in-vehicle HPC unit via the HTTPS server. This highly reduces the risk posed by traditional diagnostic tools to compromise ECUs.

C. *Security-critical Diagnostic Communication Services*

In UDS diagnostic communication, the functionalities offered by an ECU for using a diagnostic tool are referred to as "services". These services include reading and writing data to operate the ECU as well as diagnostic commands, such as fault code retrieval. The conversation surrounding automotive cybersecurity threats highlights the potential for attacks via the OBD connector by exploiting these services. Previous research [9] and [10] have demonstrated that the following UDS have been susceptible to exploitation.

- Input/Output Control Service: This service controls the input and output signals that are connected to the specified ECU from the diagnostic tool. Its primary function is to identify the failure point. For instance, if the wipers do not operate even after turning on the wiper switch, this service can be used to forcibly drive the wiper motor, and if the wipers start operating, it proves that the motor and its wiring have no problem. This approach helps in efficiently narrowing down the failure point. However, this service can lead to generating hazardous vehicle behavior that the driver did not intend.
- Write Data by Local ID Service: This service is designed for configuring the initial settings and adjusting the parameters of installed components. It can, for example, be used to write the dynamic radius value of a tire to the ECU to calibrate the speedometer or enable/disable optional parts. However, if this service is abused, users may experience adverse effects, such as inaccurate information display or suspension of certain functions.
- Reprogramming Service: This service is for rewriting ECU firmware installed in sold vehicles, usually to correct quality defects in the firmware. However, if this service is abused, it could result in various issues. For instance, the rewritten ECU may behave improperly or even spoof other ECUs, leading to more significant problems, such as sending malicious communication data to other ECUs. Therefore, it is crucial to use this service only for its intended purpose and avoid any abuse.

These services are locked by default as privileged operations within many UDS ECUs. To grant access to locked services, a process known as "security access (service ID27)" is typically used to verify the legitimacy of the user or diagnostic tool. New ECUs supporting SOVD will also have similar privileged services, and such services will be locked by SOVD server in HPC.

D. *Authentication by Service ID27 "Security Access”*

In diagnostic communication by using UDS, security access communication was generally executed using the following procedure (refer to Figure 3) with a pre-shared symmetric key K.

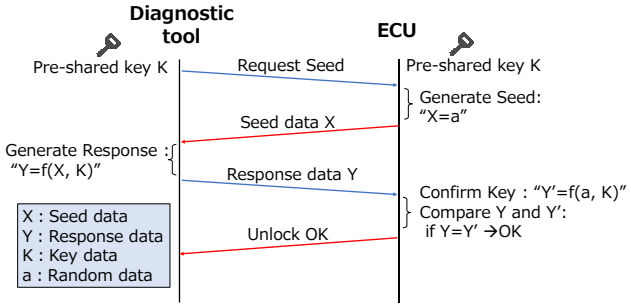


Figure 3. ECU unlock sequence by security access

1. The diagnostic tool to be authenticated sends a seed request (request seed) to the ECU to be unlocked.
2. Upon receiving the request, the ECU sends back seed data X, including random numbers, to the diagnostic tool to avoid the risk of replay attacks.
3. The diagnostic tool processes the obtained X using the key data K and computes the response data Y.
4. The diagnostic tool sends Y to the ECU. ECU calculates Y' from the K & X sent by ECU itself.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

5. If Y' and Y are the same value, the authentication is successful, and the ECU unlocks the locked critical services.

If a symmetric key is used for authentication in security access executed by such procedures, an attacker may be able to obtain the key information through reverse analysis of the ECU or diagnostic tools. Therefore, the following solutions have been devised.

- To minimize the risk of reverse key analysis, it is essential to safeguard the private key in asymmetric key authentication. The private key should not be stored in the diagnostic tool. It instead should be kept in the Hardware Security Module (HSM), which is located on the authentication server or in a secure location with restricted access outside the tool. This requires the diagnostic tool to be connected to the authentication server with the HSM. To achieve this, infrastructure development and maintenance are necessary, such as installing a network environment at the garage and managing accounts that enable the diagnostic tool to log into the authentication server.
- Service ID27 does not provide security functions, such as user-privilege management or session key exchange with authentication, requiring each auto manufacturer to develop its own customizations. To remedy these issues, ISO 14229-1 has been updated, and a new UDS service, Authentication (Service ID 29), began in 2020.

E. Authentication by Service ID 29 "Authentication"

This new authentication service has the following advantages in terms of security compared with the previously used security access.

- Support for Public Key Infrastructure (PKI)-based authentication mechanisms.
- Support for session key exchange during authentication.
- User-privilege management support.

This service is expected to spread and be implemented into in-vehicle basic software, such as AUTOSAR (AUTomotive Open System ARchitecture). This will make it easier for vehicle manufacturers and component suppliers to implement higher security measures than ever before.

Some automotive ECUs, however, use processors with low processing power, such as 16-bit microprocessors. PKI-based authentication requires certificate parsing, hash calculation, and processing of asymmetric key cryptography, which cannot be afforded by such processors.

To introduce user-privilege management, it is necessary to properly construct and operate a system outside the vehicle that manages the privilege settings for each user and their expiration dates. For example, there is a need for special diagnostic communication during the vehicle-development phase and vehicle-production processes, and the introduction of Service ID 29 will not be effective unless account management for users and production facilities with such special privileges is properly implemented. Therefore, it is necessary to improve not only technical measures, such as the development of ECUs and privilege-management systems, but also the management and operation of the user management process at the same time.

F. Authentication of SOVD

SOVD solves the problem of low processing power ECUs by its centralized in-vehicle network architecture shown in Figure 1. SOVD server in HPC can authenticate the clients as a representative for all in-vehicle ECUs, because all diagnostic communication requests come in the SOVD server.

ASAM API specification [8] does not have a single standardized authentication method but has an informative specification using a Token base authentication and authorization.

III. CURRENT STATUS AND ISSUES OF REMOTE DIAGNOSIS

A. What is Remote Diagnostics?

Section II described wired diagnostic communication. Remote diagnosis refers to diagnostic communication using a wireless communication unit installed in the vehicle, enabling remote diagnosis from a location away from the vehicle. Figure 4 shows a typical configuration for remote diagnosis.

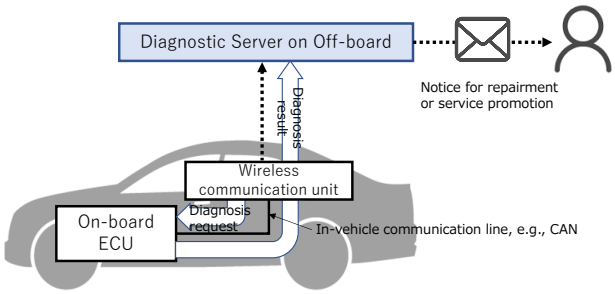


Figure 4. Example of remote diagnostic system

In remote diagnosis, the wireless communication unit in the vehicle requests the onboard ECU to self-diagnose if any failures occur. The onboard ECU sends back the diagnosis results, which the wireless communication unit forwards to the remote diagnosis server, enabling the diagnosis results to be obtained without entering the vehicle.

If a malfunction occurs, the diagnostic server notifies the user and urges them to repair or go to a garage, preventing the malfunction from becoming a serious problem.

While it is technically possible for the wireless communication unit to transmit requests, such as program rewriting and Input-Output (IO) control, these requests are designed for use under the control of a mechanic only when the vehicle is stopped for maintenance or repair. If operated remotely and unintentionally by the driver while the vehicle is running, they may cause safety-related problems.

In a previous study [11], security measures for remote diagnostic systems were proposed. These measures are based on the assumption that the wireless communication unit

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

(called the telematics module) is correctly installed in the vehicle and properly works. However, the vulnerability of the wireless communication unit can be exploited, making it an entry point for man-in-the-middle attacks through hijacking. This should be assumed as one of the major threats in recent automotive security risk analysis.

With current remote diagnostics, it is assumed that the wireless communication unit can be hijacked, thus the following risk mitigation measures were introduced as illustrated in Figure 5.

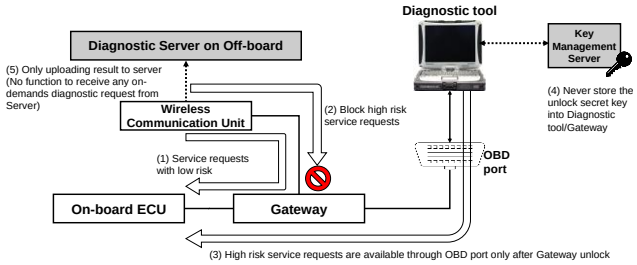


Figure 5. Example of conventional risk-mitigation measures

(1) The gateway is responsible for forwarding only low-risk service requests when the requests come from the Wireless Communication Unit, such as the reading of trouble codes and error log data. These available requests are registered in Gateway's static whitelist of authorized requests to prevent change it dynamically by any privilege escalation attack.
(2) If any high-risk service requests come from the Wireless Communication Unit, the gateway always blocks such requests because such requests are not in the whitelist.
(3) High-risk diagnostic service requests are available only by wired access through the OBD port after unlocking the Gateway's security protection. The in-vehicle network ports of Gateway for OBD port and Wireless Communication Units must be physically separated to identify the source of the service requests by the Gateway.
(4) The secret key required to unlock the Gateway protection are not stored in the diagnostic tool nor gateway to which the attacker can obtain physical access by purchasing them.
(5) The Wireless Communication Unit is not equipped with a function to receive arbitrary diagnostic requests on demands from an off-vehicle server but only uploads the diagnostic results. The Wireless Communication Unit should be able to transmit only predefined low-risk service requests to On-board ECU through Gateway, such as reading trouble codes.

B. Service Expansion Requirements for Remote Diagnosis

Contrary to the limitations imposed by the risk-mitigation measures described in Section III.A, the following use cases are required for remote diagnosis.

Use case 1: Remote use of critical commands (e.g., IO control services listed in Section II.C) required for pre-diagnosis to identify parts to bring to a repair place of a vehicle that is stopped on the road due to a malfunction.
Use case 2: Remote identification and handling of failure causes by senior mechanics (use case similar to telemedicine).
Use case 3: Remote diagnosis of whether a vehicle that has a trouble can be driven to a repair shop or whether it can be made drivable with simple road service assistance.
Use case 4: Understanding the status of a cyberattack (related to Section V.B.9).

C. Security Risks from Expansion of Remote Diagnostic Services

When responding to the need for service expansion as described above, the abuse of critical diagnostic services increases the risk that safety will not be maintained, and fatal incidents will occur.
Risk 1: Expanding the impact of incident occurrence: The impact of abusing critical diagnostic services becomes significant because such services can manipulate or illegally modify safety-related vehicle components, for example, the braking or steering system.
Risk 2: Failure to confirm the vehicle owner's consent and safe vehicle conditions: Conventionally, the owner's consent could be indirectly obtained by receiving the vehicle key to physically access the OBD connector inside the vehicle. The repair operator had to ensure that the vehicle was in a safe condition, such as by locking the wheels. By allowing work to be done remotely, the above measures cannot be used.
Risk 3: Risk of abusing remote operation authority: Conventionally, the OBD connector cannot be accessed unless the vehicle is physically in the hands of the mechanic, so there is no need to worry about workers to whom the owner has entrusted repairs in the past without the owner's permission. Remote operations do not have these restrictions, increasing the risk of insider attack by privilege holders.

To address these risks, the following countermeasures will be necessary:
Countermeasure against risk 1): To prevent the unlocking of critical commands through external communication only, a special in-vehicle operation for enabling remote diagnostics must be required as proof of the vehicle owner's consent.
Countermeasure against risk 2): In addition to electronically authenticating permission from the vehicle owner, the vehicle receiving the remote diagnostic command also checks the physical condition, indicating that the vehicle is not running but awaiting servicing as one of the conditions for conducting remote diagnosis.
Countermeasure against risk 3): When authenticating workers who conduct remote diagnosis, a mechanism to check whether the validity period of the work and the authority to carry out the work have been revoked is needed.

IV. Proposed Security-Risk-Mitigation Measures

An overview of the remote diagnostic system operation is shown in Figure 6. This system can execute remote diagnosis with the following procedure.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

A. Remote Operation Permission

The vehicle owner who wants to solve a problem with the vehicle or a mechanic who receives a repair request by the owner first conducts owner authentication in the vehicle. The following permission methods are possible.

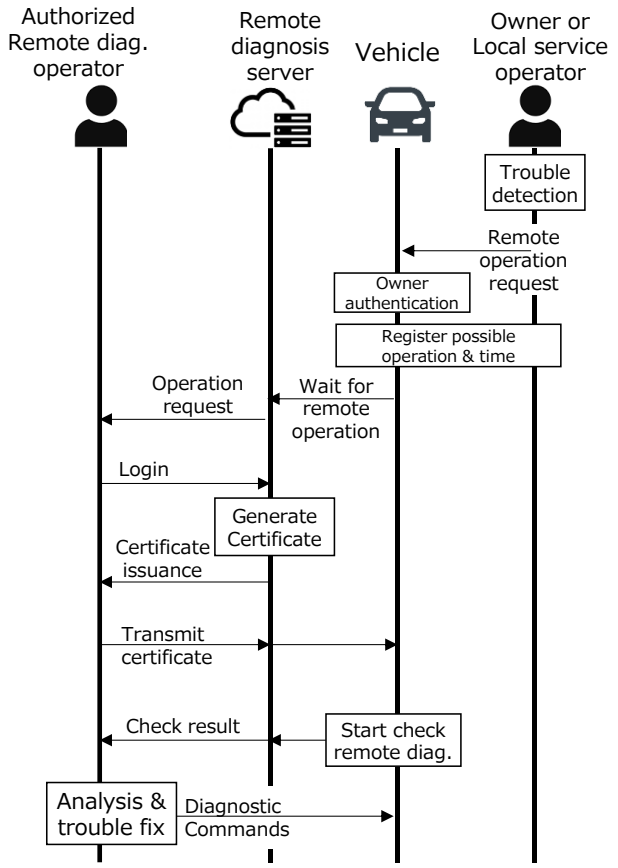


Figure 6. Overview of system operation

- The Human Machine Interface (HMI) in the vehicle (navigation-system screen, LCD of cluster meter, etc.) is used to authorize remote diagnosis. This can be done using a PIN or password preset by the vehicle owner to increase the reliability of the authentication.
- The presence of multiple intelligent keys in the vehicle is a condition for starting remote diagnosis permission. This is intended to detect differences from normal driving when only one key is present in the vehicle by the owner bringing a spare intelligent key into the vehicle.
- Pair the owner's smartphone with the vehicle and store the authentication information in the smartphone. The vehicle accepts remote diagnostics only for a certain period after successful Near Field Communication (NFC) authentication.

It is important to combine multiple conditions to increase the reliability of the remote diagnostic authorization described above.

B. Registration of Permitted Operations and Periods

Assuming that part of a vehicle component is malfunctioning, multiple input HMIs should be provided.

1) The owner's smartphone or operator's PC inputs the information and registers the operation information to be allowed to the remote diagnosis server and its validity period.
2) Input the information on an HMI in the vehicle and register the operation information to be allowed to the remote diagnosis server via the vehicle's wireless communication unit.

The user can select which operations to allow by using HMI of vehicle infotainment system or Web site of Remote diagnosis server, for example, reprogramming firmware or resetting the ECU.

C. Requesting Analysis via the Diagnosis Server

The remote diagnosis server notifies the target vehicle that the permitted operations and validity period of the work have been registered. At this time, the vehicle confirms that "permission for remote operation" has been granted in advance and that the vehicle is in a safe maintenance state (e.g., the vehicle is stopped, and the engine hood latch is open), and notifies the remote diagnosis server that it is "waiting for remote diagnosis".

The notification data from the vehicle can be supplemented with the vehicle's location information obtained from GPS, etc., and a request can be made to the diagnosis server to limit the locations where remote diagnosis is permitted to the area around the current location. Upon receiving this notification, the remote diagnosis server sends a failure-analysis request to an appropriate operator from among the "authorized remote diagnosis holders" registered in advance.

It is also effective to include a one-time password in the failure-analysis request to increase the reliability of the certificate-issuance process in the next step.

D. Generating and Issuing Certificate of Remote Diagnostic Operations

When an authority holder receives the notification, they log into the remote diagnosis server and request the issuance of a working certificate. To enhance security, it is recommended to require the entry of a one-time password, which is sent only to the authority holder when they receive the notification of the analysis request, as a condition for issuing the certificate.

The issuance of this certificate is also sent to an HMI of the vehicle and the registered smartphone of the vehicle owner. If this notification indicates that a remote diagnostic request was not intended by the driver or vehicle owner in the vehicle, the "waiting for remote diagnosis" status of the vehicle can be canceled, or an instruction can be sent to the remote diagnosis server to stop remote operation for the vehicle in question as a risk-mitigation measure.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

The remote diagnosis server issues a certificate to the authority holder as a token that records the expiration date and permitted operating privileges.

E. Access to Vehicles from Remote-diagnostic-authority Holders

The authority holder responsible for remote diagnosis sends a token to the target vehicle. The vehicle checks the token's signature using the remote diagnosis server's pre-shared public key, and if the token is issued by the legitimate remote diagnosis server and is still valid, the vehicle unlocks the remote diagnosis communication and authorized operation rights recorded on the token. The expiration date on the token prevents unauthorized access after the work is completed, which is not intended by the owner.

V. Avoiding Constraints When Implementing Proposed Measures in Vehicle Component

A. Implementation Constraints to Consider

The following are constraints in implementing the proposed measures in a vehicle.

1. Automobiles are equipped with dozens of ECUs that execute diagnostic communications, and changing all these ECUs to components that implement security measures for remote diagnostics would require large-scale development and take too much time to implement.
2. The resources required to adopt enhanced authentication algorithms, user rights management and expiry date management cannot be implemented in components with resource-constrained processors, such as 16-bit microcontrollers, which limits their applicability.
3. Direct end-to-end communication between the off-vehicle server, which is the connection source for remote diagnosis, and the ECU to be diagnosed, creates a pathway for a direct attack on the ECU inside the vehicle from the off-vehicle server if a vulnerability exists in the ECU communication software, so a workaround is necessary.
4. Introducing SOVD architecture will be able to solve the constraints from 1 to 3 above, but it will bring other security risks, especially new risk caused by in-vehicle HTTPS server, because it makes a new attack surface having an open port to the internet.
5. Since SOVD uses REST API base communication, popular user authentication protocols (e.g., OpenID Connect [12], OAuth2.0 [13]) for web services would be preferable of the remote operator authentication. However, there is no available authentication service provider covering global vehicle markets to prove that the remote access requester is not a cyber attacker, but a skilled vehicle diagnostic operator, because proving it requires identity verification to check the requester's car maintenance experiences. Most of the vehicle manufactures want to avoid localizing the authentication system for vehicle development efficiency. Therefore, minimizing diversity of the remote operator authentication is an important demand of the remote diagnosis.

B. Our measures for UDS Generation to avoid Constraints

We devised our security-risk-mitigation measures shown in Figure 7 to avoid the constraints described in Section V.A.

To reduce the security risk of remote diagnosis, these measures have the following features that the conventional measures shown in Figure 5 do not have.

Measure 1: The in-vehicle gateway is used as the master ECU to manage the remote diagnosis control.

Measure2: The master ECU has a zone for communication with the external server via a wireless communication unit (Zone 1) and another zone for in-vehicle communication (Zone 2), which verifies certificate data for remote diagnosis and sends and receives diagnosis commands to and from multiple ECUs in the vehicle. Zones 1 and 2 are separated by hardware or software, such as a hypervisor, to prevent direct attacks from outside the vehicle to Zone 2, which executes in-vehicle communication processing.

Measure 3: Zone 1 of the master ECU communicates with the remote diagnosis server using Transport Layer Security (TLS) to prevent the in-vehicle wireless communication unit from eavesdropping on and falsifying communication data

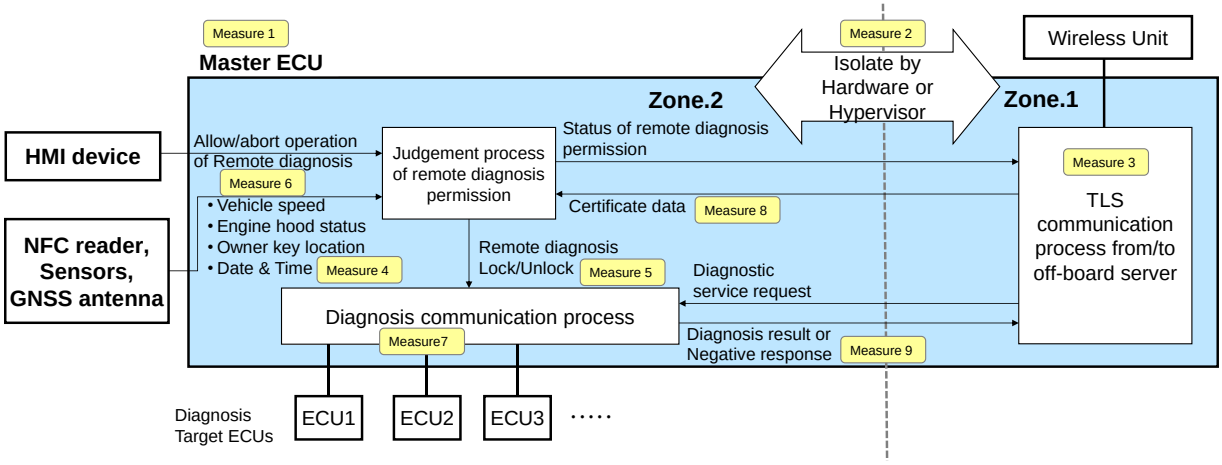


Figure 7. Implementation example for UDS generation using master ECU

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

between the master ECU and remote diagnostic server (a countermeasure against man-in-the-middle attacks).

Measure 4: To check the expiration date & time of the public key certificate for TLS, the master ECU must manage the absolute date & time using not only Global Navigation Satellite System (GNSS) data, but also trustable in-vehicle timer counter, because GNSS signals may get a replay attack. For example, the master ECU can detect the replayed GNSS signal when a newly received GNSS signal shows older time than the elapsed time of in-vehicle timer counter value or received signals in the past. Even if the timer counter's accuracy is low, e.g., a few seconds per month, it can still detect invalid GNSS signals when the difference between the result of adding the counter's elapsed time to the date and time of the last received GNSS signal and the date and time of the newly received GNSS signal exceeds the tolerance range.

Measure 5: The master ECU boots with the remote diagnostics as locked status by default. In the locked status, "Diagnosis communication process" in the master ECU rejects all diagnostic service requests coming from TLS communication process to prevent receiving any unexpected remote requests. Only when remote diagnosis is unlocked, the "Diagnostic communication process" in Zone 2 executes diagnostic communication in response to a remote-diagnostic-service request from Zone 1.

Measure 6: If the master ECU receives the result of the remote-diagnosis permission correctly executed with an HMI in the vehicle and the "remote diagnosis permission condition" is satisfied within a certain period after that, the master ECU unlocks the remote diagnosis process and enters the "waiting for remote diagnosis" state. The "remote-diagnosis-permission condition" is, for example, all the following conditions are satisfied.

(1) Successful verification of certificate received from Zone 1.
(2) The HMI executes remote diagnostic permission in the vehicle and is not canceled.
(3) No timeout has occurred since the operation in (2).
(4) The vehicle must be stopped.
(5) Signals indicating that the vehicle is in a service condition (e.g., engine hood is open) are detected.

Measure 7: The target ECU for remote diagnosis connected to the master ECU operates by receiving diagnostic commands from the "Diagnostic communication process" implemented in Zone 2. The master ECU executes the verification process of the certificate data and permission by the HMI, which are necessary as security measures of remote diagnosis, thus avoiding software and hardware changes in the target ECU.

Measure 8: If the verification of certificate data fails more than once, the time until accepting the next verification is extended.

Measure 9: If a diagnostic-service request that is not authorized by the certificate is received, the diagnostic communication process returns a negative response. This history is stored in remote diagnosis sever. The request commands thus rejected are signed and included in the negative-response history data to prevent repudiation by the authorized remote diagnosis operator.

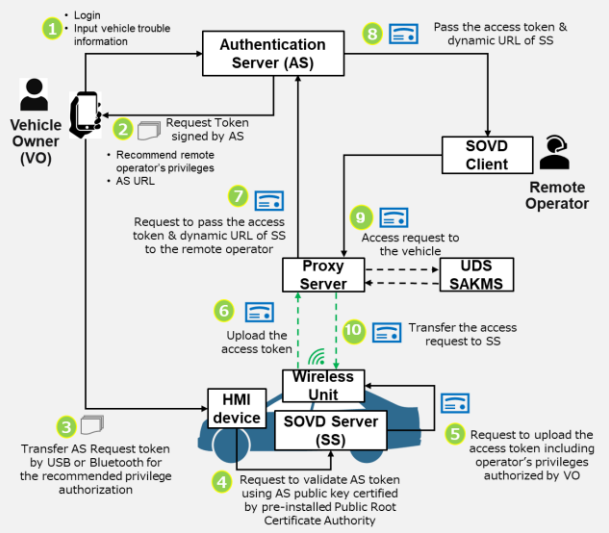


Figure 8. Example of remote SOVD sequence

In our previous paper [1], we inspected feasibility of implementing zone separation measures from a processing performance viewpoint. We conducted the experiment on Renesas R-carS4N-8A processor, and we confirmed that the proxy processing required for separating the zones could handle 96 Mbps of real-time video transfer with very low latency (1.675 ms), and we found no performance problem.

C. *Proposed SOVD Sequence to Address Constraints*

Figure 8 shows an example case of our proposal sequence to solve the SOVD constraints 4 & 5 described in Section V.A.

The SOVD generation will have the following sequence steps:

Step 1: A vehicle Owner (VO) can subscribe any remote diagnosis services (e.g., provided by the vehicle manufacturer, by a local car maintenance company etc.) and have its access account for remote service request. When VO wants to request for the remote diagnosis, the owner logins to Authentication Server (AS) and inputs the vehicle trouble information.

Step 2: Based on VO's input, AS generates a request token including a set of recommended remote operator's privileges. VO downloads this AS request token into VO's smartphone. If this request token is standardized among various remote diagnosis service providers and signed by PKI based certificate authority chain, the vehicle can manage the diversity of the service providers.

Step 3: VO transfers the downloaded AS request token to HMI device in the vehicle. HMI device extracts the recommended privileges and shows them in the touch screen display of HMI device for VO's approval. VO can accept them all or change them to a minimum set of privileges.

Step 4: After the approval by VO in step 3, HMI device sends the privileges authorized by VO to SOVD Sever (SS)

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

and requests SS to validate signature of AS request token. SS extracts the certificate chain information to get a public key of AS for the token signature validation.

Step 5: If signature of AS token is valid, SS generates an access token and requests Wireless Unit to upload it to AS by using AS URL in AS request token.

Step 6: The wireless unit passes the access token to the Proxy Server. This proxy server can be a bridge to UDS Security Access Key Management Server (SAKMS) of the vehicle manufacturer when an old UDS ECU sends a challenge to unlock its critical diagnostic operation. This bridge function can avoid connecting the remote client to the vehicle manufacturer's UDS SAKMS directly to get SA unlock response.

Step 7: The Proxy Server requests AS to send the access token to the remote operator. This Proxy Server hides the internet address of Wireless Unit by generating a random dynamic URL (Uniform Resource Locator) to prevent unexpected direct access to the vehicle from the public network. This dynamic URL is also shared with AS to inform it to the remote operator as a virtual URL of SS.

Step 8: AS passes the access token and dynamic URL to the remote operator.

Step 9: The remote operator starts accessing to the vehicle using the shared token & dynamic URL.

Step 10: The Proxy Server transfers the remote operator's access request to a target Wireless Unit specified by the dynamic URL.

As the Proxy Server hides the vehicle's access address from the public network, it can be the first firewall against the cyber-attack risk caused by constraint 4 in Section V.A.

Similarly, using PKI for the public key certificate chain at step 4 of this sequence enables covering various Authentication Servers in the global market, so it helps to solve constraint 5 in Section V.A.

An additional security advantage of this sequence is VO's approval by HMI device in the vehicle. This approval process requires some physical access actions in the vehicle cabin. This point can be a strong proof of VO's authorization.

Even though we implement security-risk-mitigation measures mentioned above, in-vehicle component also should have a similar zone separation as same as UDS generation in Figure 7 considering "Defense in depth" principle.

Figure 9 shows an example of the zone separation implementation for SOVD generation. In this example, "Public SOVD Server" in Zone-1 should provide HTTPS communication from/to the outside of vehicle through the Wireless Unit to mitigate the risk caused by constraint 4 in Section V.A. This "Public SOVD Server" has the similar functions of "Data communication process from/to off-board server".

Zone-2 hosts two new functions "Authorization Server" and "SOVD Manager" to match the SOVD software architecture.

"Authorization Server" has three token processes, the Request Token validation, the Access Token generation & check. The Request Token is authorized and validated by the vehicle owner's operation on HMI device and PKI Root CA Public Keys. If this authorization and validation are OK, the Access Token is generated using HPC's Private Key. When a remote operator starts access to the vehicle by sending its access token, "Authentication Server" also checks the access token sent by the remote operator. HPC must have a Secure Storage to protect the integrity of public keys and confidentiality of its private key.

"SOVD manager" has similar functions of "Diagnosis communication process" in Figure 7. It can lock or unlock the remote diagnostic communications using inputs from "Authentication Server" and "NFC reader, Sensors". "SOVD manager" switches the remote diagnosis communication path to the Virtual ECUs or old physical ECUs using UDS communication.

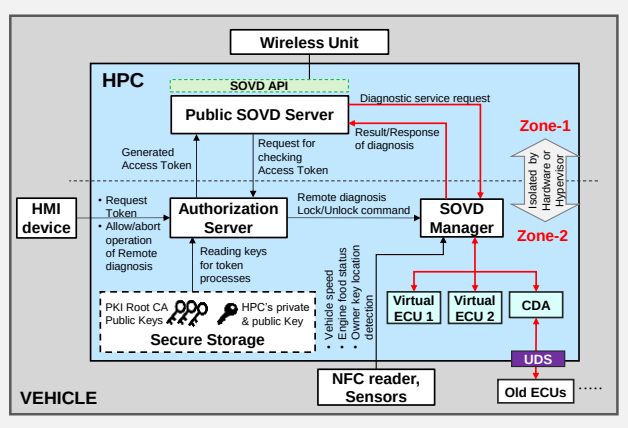


Figure 9. Implementation example for SOVD generation

VI. FUNCTIONAL SECURITY EVALUATION

Figure 10 illustrates the architecture of Proof-of-Concept (PoC) simulation environment based on our SOVD sequence described in Section V.C. In this evaluation, we focus specifically on the components highlighted in the figure, namely the Remote SOVD Client, the in-vehicle SOVD Public Server hosted on the HPC, and the Vehicle HMI. As these three elements represent the primary attack surface where authentication and authorization decisions occur.

Based on the simulation environment, we defined three functional test cases. The objective of evaluation is to verify the in-vehicle authorization concept functions correctly under realistic conditions. The evaluation focuses on two fundamental requirements:

Authentication Integrity: The SOVD Server must reject unauthenticated requests or invalid tokens, accepting only properly signed and valid credentials.

Authorization (scope enforcement): Access to diagnostic endpoints is determined by the operator's assigned privileges as reflected in the access token payload.

The following subsections detail the results of these functional test cases.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

A. Test Case 1: Unauthorized Access Token Blocking

In this test case, we verify that SOVD Server enforces token-based authentication correctly. We tested three scenarios:

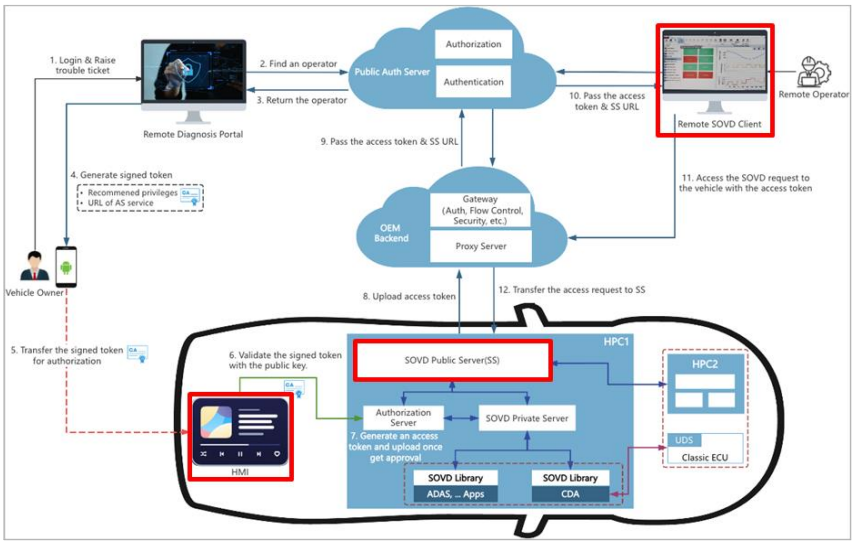


Figure 10. System Architecture of the SOVD Prototype Environment

1) Missing Authorization Header: A `GET` request was sent by the operator to the `/sovd/v1/Component` endpoint with an empty HTTP Authorization header. As shown in Figure 11, SOVD Server rejected the request with a "`401 Unauthorized`" response, which confirms that the Server does not allow unauthenticated operators to initiate any diagnostic query.
2) Invalid Token Signature: the same endpoint was accessed using a JWT token with an invalid signature. The Server returned "`401 Unauthorized`" as shown in Figure 12.
3) Expired Token: We attempted to access the API using a token that was structurally correct but possessed an expired timestamp. As Figure 13 shows, the Server responded with "`401 Unauthorized`" and identified the validation failure with a message "`claim timestamp check failed`".
4) Valid Token: Finally, a valid and correctly signed token was used. The Server responded with "`200 OK`", returning the structured component list defined by SOVD as shown in Figure 14.

B. Test Case 2: SOVD API Scope Enforcement

While Test Case 1 validates authentication integrity, Test Case 2 evaluates authorization by verifying whether protected SOVD API endpoints enforce the access permissions encoded in the JWT token. We define two operator roles: a "Viewer" role restricted to read-only operations, and a "Developer" role with full diagnostic permissions. Figures 15 and 16 show the permission configurations assigned to these roles in the simulation environment. To illustrate how these permissions are encoded in the issued credentials, we show a JWT token generated for a "Developer" operator in Figure 17. The token payload shows full access, including `GET`, `POST`, `PUT`, and `DELETE` operations, while the denyPermissions field is empty.

We then executed a functional test using the "Viewer" role. As shown in Figure 18, a "Viewer" operator successfully accessed fault information using `GET /sovd/v1/Components/adas-module/faults/C1456`, the operation succeeded with a "`200 OK`" response from the Server and returned the expected diagnostic data. However, when the same operator attempted to delete the fault code with a `DELETE` request, the Server correctly blocked the operation. Figure 19 shows the resulting "`403 Forbidden error`" with the message: "`Role 'Viewer' does not have permission to DELETE`". SOVD Server correctly interprets the permissions embedded in JWT token, and operations requiring elevated privileges (e.g., DELETE fault codes) are blocked for restricted roles.

C. Test Case 3: HMI UI Approval by VO

Finally, in Test Case 3 we validate the physical authorization step, to ensure that remote access cannot be established without explicit, in-vehicle approval by the Vehicle Owner (VO) through the in-vehicle HMI. Figure 20 shows the initial UI screenshot of HMI prompt displaying the list of permissions contained in the owner-provided certificate. Through the simulator, we show the requested privileges to VO to be selected (e.g., Body Control Module, HVAC, Power Line Communication). In this test scenario, VO selected only the "Body Control Module" permission and leave the other privileges unchecked. After approval, the HMI goes to confirmation screen shown in Figure 21,

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

indicating that “Remote diagnosis is in progress” with an option to “Quit Remote Diagnosis”. We confirm that the system enforces vehicle-side user approval before allowing any remote diagnostic activity.

D. Discussion

While the functional evaluations confirm that authentication, authorization, and in-vehicle approval mechanisms work as intended, we note that the timer counter's accuracy was not experimentally verified. However, our current vehicles already have decoded GNSS date and time information as in-vehicle CAN signals (e.g., via wireless communication unit). Therefore, even if the timer counter's accuracy is relatively low or GNSS time drift occurs, the system can still detect invalid GNSS signals when the difference exceeds a reasonable tolerance (e.g., several seconds). In practice, an attack that manipulates GNSS time by only a few seconds is highly unlikely, as such a minimal shift would not provide a meaningful advantage to an attacker. Consequently, the proposed detection approach remains effective even with coarse timer accuracy. Nevertheless, the evaluation does not yet quantify performance overhead or resilience against advanced attack scenarios such as DoS or token forgery. Future work should include large-scale stress tests, latency and resource profiling, and usability studies for HMI-based approval to validate feasibility in production environments.

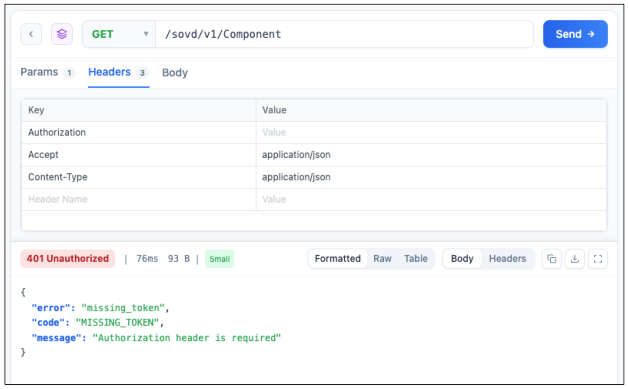


Figure 11. Response of an Empty Authorization Header

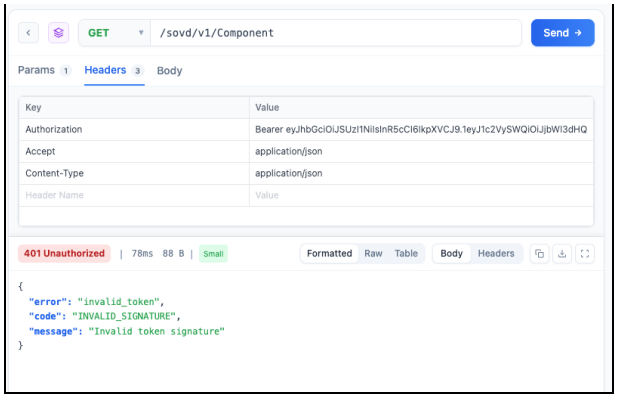


Figure 12. Response of Invalid Token Signature

Figure 13. Response of Expired JWT Token

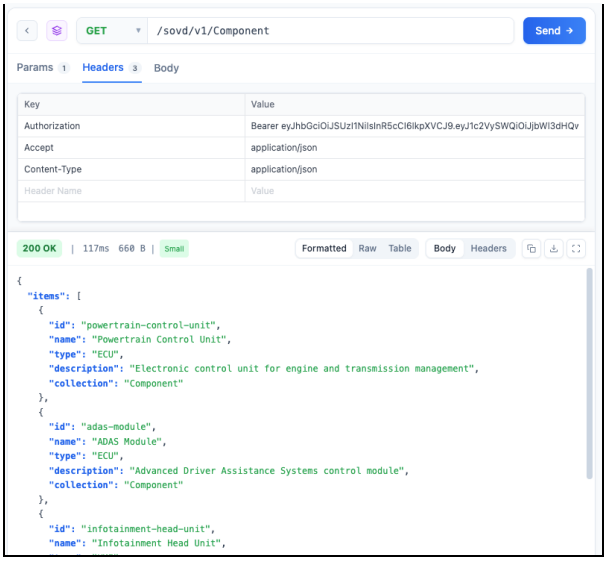


Figure 14. Response Success 200 OK with Component List

Figure 15. Viewer Role Permission Page

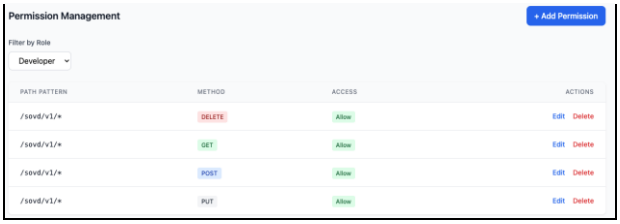


Figure 16. Developer Role Permission Page

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

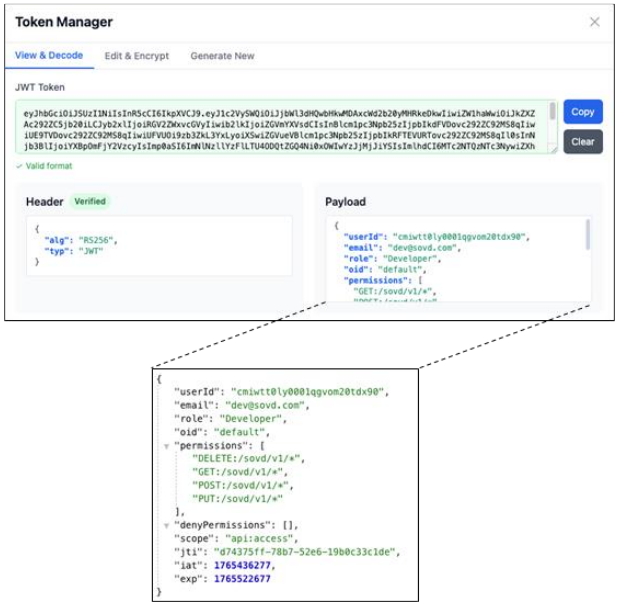


Figure 17. JWT Token Payload for Developer Role

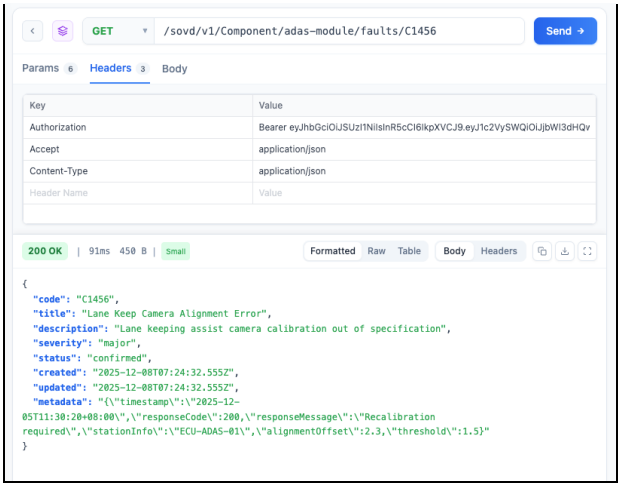


Figure 18. Viewer Operation “Read the fault info successfully”

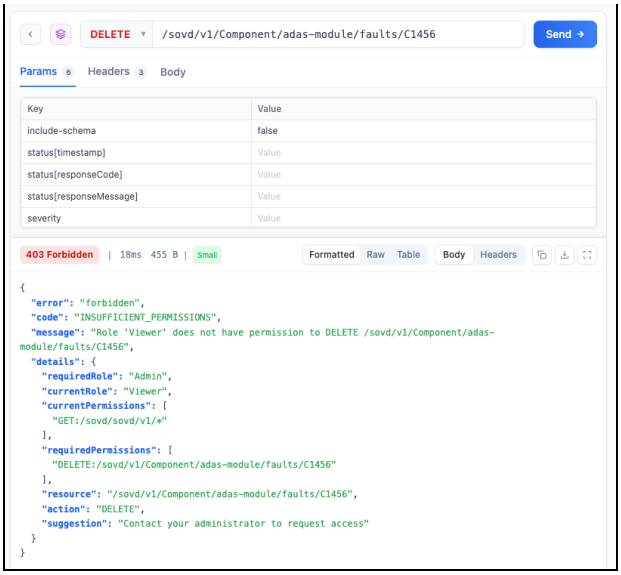


Figure 19. Viewer Operation “Failed to delete the fault”

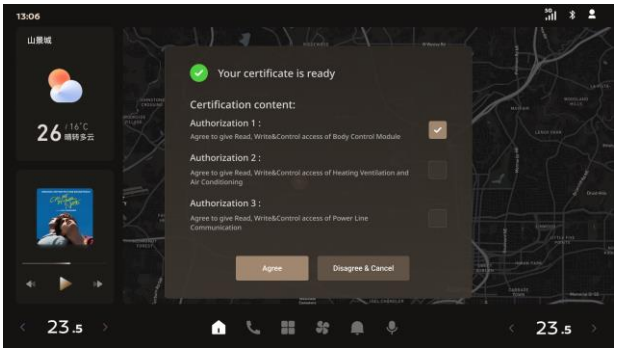


Figure 20. HMI Simulator UI for Selecting Diagnostic Privileges

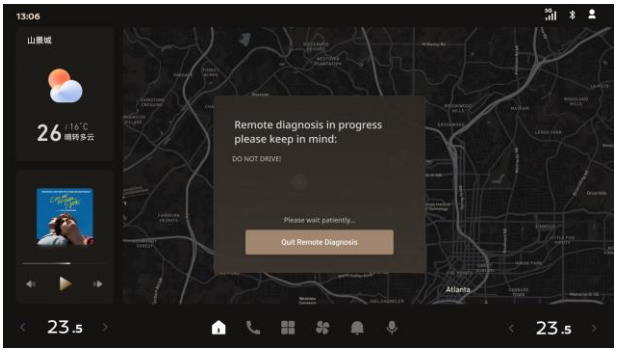


Figure 21. HMI Simulator UI: Active Remote Diagnostic Session

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

VII. CONCLUSION

Similar to our previous work, the communication software to the remote operator, "Public SOVD Server" in SOVD case, must be isolated from the security critical software modules "Authorization Server" and "SOVD manager". This measure will help to mitigate risks caused by implementation of HTTPS function in HPC.

The most important point of security-risk-mitigations for SOVD remote diagnosis is the privilege authorization in the vehicle cabin, because the feasibility of cyber-attack to the remote diagnosis system becomes easy for the attacker if both of user authentication and privilege authorization are possible on the public network. The authorization operation by in-vehicle HMI device can proof that VO (or a local maintenance operator trusted by VO) authorizes the necessary privileges for its requested remote diagnosis.

The second important point is having the proxy server between the vehicle and public network to hide the URL of in-vehicle HTTPS server. It can make difficult the port scanning by attackers and avoid unexpected access to open port 443 for HTTPS communication.

We conclude that these additional security-risk-mitigations can reduce cyber-attack feasibility to the remote diagnosis on SOVD systems.

ACKNOWLEDGMENT

We thank Nissan Technology Development Shanghai to make and test our PoC environment for this research.

REFERENCES

[1] M. Miyashita and H. Takakura "Security-risk-mitigation Measures for Automotive Remote Diagnostic System". In Proceedings of the Eighteenth International Conference on Emerging Security Information, Systems, and Technologies (SECURWARE 2024), Nice, France. 2024.

[2] C. Miller and C. Valasek, " Remote Exploitation of an Unaltered Passenger Vehicle", pp. 84–85, Blackhat Aug. 2015.

[3] H. Wen, Q. A. Chen and Z. Lin, "Plug-N-Pwned: Comprehensive Vulnerability Analysis of OBD-II Dongles as A NewOver-the-Air Attack Surface in Automotive IoT", pp. 960–961, Aug. 2020.

[4] Official Journal of the European Union. (2018). Regulation (EU) 2018/858, OJ L 151, 14.6.2018, p. 1.

[5] S. Robert and J.S. Jayasudha, "Overview of Diagnostic over IP (DOIP), Ethernet Technology and Lightweight TCP/IP for Embedded System", International Journal of Advanced Research in Computer Science, pp. 296–299, 2013.

[6] R. B. Gujanatti, S. A. Urabinahatti and M. R. Hudagi, "Suvey on Security Aspects Related to DoIP", International Research Journal of Engineering and Technology, pp. 2350–2355, 2017.

[7] M. Matsubayashi et al., "Attacks Against UDS on DoIP by Exploiting Diagnostic Communications and Their Countermeasures", 2021 IEEE 93rd Vehicular Technology Conference, pp. 1922–1927, 2021.

[8] ASAM e.V., ASAM SOVD v1.0.0: (Service-Oriented Vehicle Diagnostics). https://www.asam.net/standards/detail/sovd/ (Accessed: July. 22, 2025).

[9] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," in Blackhat USA. Las Vegas, NV, USA: Blackhat Press, pp. 86-88, 2015.

[10] S. Kulandaivel, "Revisiting remote attack kill-chains on modern invehicle networks," PhD thesis, Carnegie Mellon University, pp. 28, 2021.

[11] K. Daimi, "A Security Architecture for Remote Diagnosis of Vehicle Defects", The Thirteenth Advanced International Conference on Telecommunications, pp. 1-7, 2017.

[12] N. Sakimura, J. Bradley, M. Jones, B. De Medeiros, and C. Mortimore, "Openid connect core 1.0," The OpenID Foundation, p. S3, 2014.

[13] D. Hardt, "The OAuth 2.0 Authorization Framework," Internet Requests for Comments, RFC Editor, RFC 6749, October 2012. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6749.txt. (Accessed: July. 22, 2025).

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Algorithmic Attraction: Detecting Content Traps in YouTube's Recommendation Graph via Network Resilience and Topical Cohesion

Md Monoarul Islam Bhuiyan*, Nitin Agarwal*†
*COSMOS Research Center, University of Arkansas at Little Rock, Arkansas, USA
†International Computer Science Institute, University of California, Berkeley, California, USA
e-mail: {mbhuiyan, nxagarwal}@ualr.edu

***Abstract*—Algorithmic recommendation systems, such as YouTube, play a central role in shaping users' online content exposure, often inadvertently reinforcing *content traps*, which are clusters of videos exhibiting limited topical diversity and repetitive thematic patterns. Although prior research has highlighted concerns around algorithmic homogeneity and echo chambers, few studies provide a systematic and comprehensive framework for detecting, quantifying, and evaluating these content traps within the complex structure of YouTube's recommendation networks. In this study, we introduce a combined network resilience and topic-based framework to identify and analyze content traps by leveraging a diverse set of social network analysis (SNA) approaches applied to the recommendation graph. We evaluate the semantic coherence of the resulting clusters using BERTopic and a sensitivity-based thresholding technique to measure topical uniformity. Our findings indicate that focal structure analysis (FSA) consistently uncovers clusters with higher semantic homogeneity compared to traditional community detection methods and, under moderate threshold settings, outperforms several centrality-based approaches in capturing cohesive content regions. By integrating structural and semantic insights, this work provides a robust methodological foundation for detecting content traps and enhances our understanding of the intricate relationship between network topology and thematic repetition in algorithmically curated content.**

Keywords-Focal Structure Analysis; Social Network Analysis; YouTube; Network Resiliency; Content Trap; YouTube Recommendation Network.

I. Introduction

YouTube, the world's largest video-sharing platform, plays a central role in shaping user exposure to content, with over 70% of watch time driven by its recommendation algorithm. While this algorithm helps users discover relevant content, it can also lead to the emergence of *content traps*, which are repetitive and thematically concentrated recommendations. These contents may limit exposure to diverse viewpoints and reinforce algorithmic bias, particularly in politically or culturally sensitive topics.

Despite growing concerns around algorithmic homogeneity and echo chambers, there remains a lack of systematic approaches to identify and assess such content traps within YouTube's dynamic recommendation system. This study addresses that gap by analyzing content traps through a structural and topical lens to identify cohesive sets of videos that form homogeneous content regions. As a result, such structures contribute to thematic uniformity and limit user exposure to diverse content.

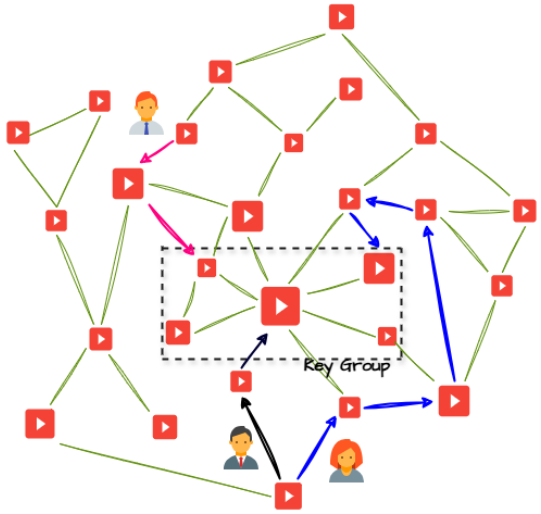


Figure 1. Illustration of a content trap within a YouTube recommendation network where the nodes represent videos and edges denote recommendation links. The "Key Group" refers to a structurally identified set of videos in the recommendation network that share highly similar themes, making them a potential content trap.

In this study, we construct a multi-hop YouTube recommendation network using a keyword-seeded crawl strategy and apply a set of SNA techniques, including FSA, Louvain, Leiden, Label propagation, and centrality-based slicing, to extract structurally meaningful clusters. We then assess the resilience of the network by measuring its fragmentation after removing each method's identified groups. To evaluate thematic coherence, we apply BERTopic to video transcripts and quantify topical uniformity using a sensitivity-based thresholding approach.

Network resilience [1] offers a complementary perspective by evaluating how the removal of structurally cohesive clusters affects the overall connectivity and stability of the recommendation network. Metrics such as flow robustness and the giant component ratio quantify the network's ability to maintain information flow in the absence of critical nodes or groups. By integrating resilience analysis with topical homogeneity assessment, we can identify clusters that not only exhibit strong thematic similarity but also act as pivotal bridges sustaining the recommendation pathways. This dual analysis provides a more nuanced understanding of the algorithmic structures that reinforce content traps [2] and their potential

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

influence on user exposure and engagement.

Our contributions in this study are thus as follows:

- We propose a network and topic-based framework to detect content traps by extracting structural groupings from YouTube's recommendation network and then applying topic modeling to assess their topical homogeneity.
- We introduce a sensitivity-based method to evaluate topical homogeneity across these groupings.

This work offers a novel perspective on how structurally derived groupings in YouTube's recommendation network reveal patterns of topical uniformity [3]. By identifying clusters of videos with high topical consistency, we show how certain regions of the recommendation network function as potential content traps, underscoring the need for a deeper analysis of algorithmic curation.

The findings of this study also have several practical and theoretical implications. Practically, our framework can guide platform designers and content moderators in detecting algorithmically reinforced content traps, enabling interventions to diversify user exposure and reduce the reinforcement of biased or polarized content. Theoretically, the study contributes to computational social science by demonstrating how network structure and topic homogeneity interact to shape information flow, providing a blueprint for future analyses of algorithmic influence in social media ecosystems. Furthermore, our resilience-based approach offers a quantitative measure to assess the potential impact of removing or modifying critical clusters, informing strategies to promote healthier information dissemination.

The rest of the paper is organized as follows. Section II reviews existing studies on identifying network structures, including detecting authoritative and community approaches, and measuring network resiliency metrics, along with content homogeneity in recommender systems. Section III outlines the experimental methodologies applied, while Section IV presents the findings of our research. Finally, Section V summarizes the study and suggests directions for future research.

II. Related Work

In this section, we review prior work on structural detection methods in social network analysis, network resiliency metrics applied in social network analysis, and the emergence of topical homogeneity in algorithmically curated content, highlighting the gap our study aims to address.

A. Structural Detection in Social Networks

Detecting structurally significant subsets of nodes, whether influential individuals or cohesive groups, has been a core focus in social network analysis. Among the most established techniques are centrality-based approaches, which quantify the importance of nodes using measures such as degree, betweenness, closeness, and eigenvector centrality [4]–[6]. These metrics have been widely used for identifying nodes that control information flow or act as central hubs in communication, transportation, or online platforms.

In contrast to node-level ranking, community detection algorithms group nodes into modular clusters based on internal density and inter-group sparsity. Notable methods include Louvain [7], Leiden [8], and Label Propagation [9], each of which strikes a balance between scalability and structural accuracy. These techniques have been extensively applied in contexts ranging from citation networks to social media interactions, revealing latent clusters that often correspond to real-world social or topical boundaries [10].

FSA [11] is one such method within the broader framework of SNAs. Initially proposed to extract small, densely connected structures that extend beyond traditional communities, it has been applied in domains such as biological and communication networks [12]. While not as widely adopted as classical community detection or centrality measures, it offers an alternate view of influence through group-level connectivity and density.

B. Network Resiliency Metrics

Network resilience, like influential node and community identification, is crucial in Social Network Analysis (SNA), denoting a network's ability to withstand disruptions while maintaining core functions. The study by Bertoni et al. [13] employs social network analysis to identify key contributors to resilience in an intensive care unit, integrating SNA-derived indicators with non-network attributes, whereas another research comprehensively reviews resilience functions and regime shifts in complex systems across various domains through empirical observations, experimental studies, and theoretical analysis [14]. Several metrics have also been developed to quantify network resilience in the face of disruptions, such as flow robustness [15], and giant component ratio [16].

However, a key gap exists in current research. While these metrics effectively measure network resilience, they have not been extensively applied to the context of social networks like YouTube. Our work aims to bridge this gap by incorporating network resilience approaches into the analysis of social networks, offering a more comprehensive understanding of their ability to adapt and function under various stresses.

C. Topical Homogeneity in Recommendation Systems

The prevalence of algorithmically curated content on social media platforms has intensified concerns about exposure to homogeneous information. Pariser's foundational work [17] describes how personalization algorithms isolate users from diverse viewpoints, potentially reinforcing cognitive bias. Subsequent empirical studies have confirmed this phenomenon on platforms such as Facebook [18] and Twitter, where content similarity and repeated exposure to aligned narratives have led to increased polarization and the spread of misinformation.

In the context of YouTube, recent research has documented how recommendation algorithms can promote narrow thematic loops, particularly around political and ideologically sensitive topics [19], [20]. Ribeiro et al. [20] analyzed YouTube's political video recommendations and found evidence of funneling toward more extreme content, while Hosseinmardi et al. [19]

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

examined engagement dynamics, revealing that recommendation pathways often reinforce existing content themes that may limit content diversity.

Efforts to quantify this homogeneity have included semantic similarity measurements using embeddings; topic modeling approaches such as Latent Dirichlet Allocation (LDA) and BERTopic; and entropy-based diversity metrics [21], [22]. BERTopic, in particular, has gained traction due to its ability to extract interpretable topics using transformer-based embeddings and class-based TF-IDF representations [23]. This enables a more nuanced evaluation of topical coherence across sets of content, making it well-suited for analyzing thematic consistency in recommendation networks.

Despite growing attention to content homogeneity, few studies have systematically analyzed the structural underpinnings of content homogeneity within recommendation networks, especially on YouTube [2]. Most existing studies have focused on content themes, topical polarization, and engagement metrics, or have examined the role of semiotics in guiding thematically aligned exposure patterns [24]–[26], leaving a gap in understanding the purely structural factors that influence how content traps can form. Our work addresses this gap by systematically applying topic modeling to the outputs of various structural detection methods, including Louvain, Leiden, Label Propagation, centrality-based groups, and FSA, within a real-world YouTube recommendation network. By quantifying topic similarity across these groupings, we provide a novel lens on how structure and semantics intersect [27]. This offers insight into how recommendation systems may reinforce thematic uniformity at the group level.

III. Methodology

This section describes the comprehensive methodology employed to analyze content traps within YouTube's recommendation network, focusing on both structural and topical dimensions. Our approach integrates network science, social network analysis (SNA), and natural language processing (NLP) techniques to capture how algorithmic curation amplifies cohesive content clusters. We first detail the dataset and crawling strategy used to construct a multi-hop recommendation network centered on culturally and politically significant content. Subsequently, we describe the methods for detecting structurally meaningful groups, including focal structures, centrality-based slices, and community detection algorithms.

To evaluate the importance of these groups within the network, we employ network resilience metrics, which quantify the impact of removing specific clusters on the connectivity and cohesion of the recommendation space. Finally, we incorporate topic modeling and sensitivity-based analysis to assess thematic consistency within structurally identified groups. By combining these structural and topical evaluations, our methodology provides a holistic framework for identifying and characterizing content traps in a large-scale recommendation ecosystem.

A. Dataset Background and Collection

The data collection in this study was carefully structured to systematically capture YouTube's algorithmic patterns via its 'watch-next' recommendation system. We focused on a specific context: the Cheng Ho propaganda dataset. The following sections provide a concise background for this context and explain the rationale behind its selection for analysis.

Cheng Ho Propaganda - In contemporary discourse, the Chinese Communist Party (CCP) has reinterpreted the story of the 15th-century admiral Zheng He, also known as Cheng Ho, to bolster its current political messages. Once celebrated for his peaceful sea voyages, Zheng He is now portrayed as a symbol of religious tolerance and diplomacy [28]. This shift aligns with China's efforts to address criticism regarding its treatment of Uyghur Muslims and to promote its Maritime Silk Road initiative. The CCP aims to enhance its soft power by rebranding this historical figure, particularly in Southeast Asia.

B. Keyword Selection and Crawling

We initiated the process by conducting workshops with subject matter experts to compile a targeted list of keywords related to the Cheng Ho Propaganda. These keywords, as shown in Table I, served as search queries on YouTube's search engine, generating an initial set of seed videos. From these seeds, the first level of recommendations was captured, and each recommended video was subsequently treated as a parent node for further crawling, continuing up to the fifth hop. As we explored deeper recommendation levels, the data volume increased exponentially. The analysis was restricted to a maximum of 5 levels for a more manageable evaluation of the data without compromising the integrity of the findings. Metadata and engagement statistics were retrieved using the YouTube Data API [29], and transcripts were collected via external transcript extraction services [30].

TABLE I. Keywords related to Cheng Ho propaganda

Keywords
Cheng Ho, Zheng He, Sam Po Kong, Sam Poo Kong, Daerah Otonom Uighur Singkiang, Singkiang, Hatta + 1957, Novi Basuki, Sam Po Bo, Cheng Ho / "Zheng He" + laksamana + damai, Sam Po Kong + Islam + Indonesia, 1421 Saat China Menemukan Dunia + "Gavin Menzies", Gavin Menzies, Cheng Ho / "Zheng He" "Columbus"

Following the initial collection, we performed data cleaning and pre-processing to ensure the quality and consistency of the dataset. Duplicate videos, inactive links, and videos without accessible transcripts were excluded from the analysis. Additionally, the textual content of the transcripts was preprocessed by removing stopwords, punctuation, and non-standard characters, while preserving key entities and named references relevant to the Cheng Ho propaganda narrative. This comprehensive preprocessing allowed us to construct a reliable, multi-hop recommendation network that accurately captures both the structural relationships between videos and their topical content, forming the foundation for subsequent structural and semantic analyses.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

C. Recommendation Network Construction

We modeled the recommendation space as a directed graph $G = (V, E)$, where nodes represent videos and edges indicate YouTube's recommendation links. The recursive crawl yielded a five-hop recommendation graph with 8,489 unique videos and 13,384 directed edges, as shown in Figure 2. This network serves as the basis for both structural analysis and semantic evaluation of content clusters. Unlike a tree structure, the network contains cycles in which seed or recommended videos may reappear through multiple recommendation paths.

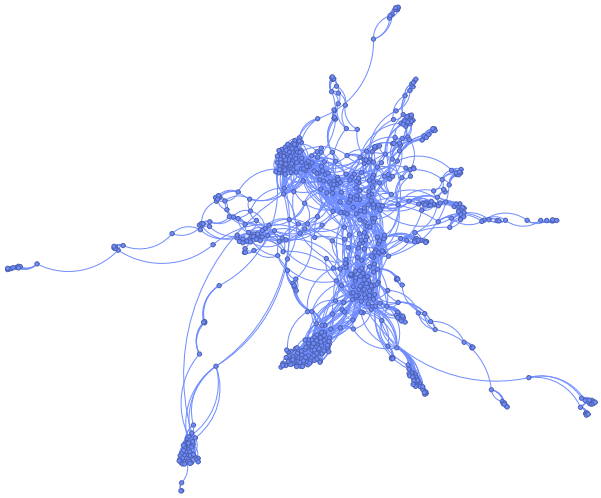


Figure 2. YouTube recommendation network centered on Cheng Ho-related content.

D. Structural Group Detection

Focal Structure Analysis (FSA): Focal Structure Analysis (FSA) is a network-centric optimization framework designed to identify structurally cohesive and influential subsets of nodes, referred to as focal structures, within a social network. Unlike traditional approaches that rely solely on node-level centrality metrics or global community detection, FSA focuses on extracting minimal yet structurally significant node groups that collectively exert strong influence across the network.

Formally, consider a social network represented by a graph $G = (V, E)$, where V denotes the set of users (nodes) and E represents their interactions (edges). The objective is to identify a subset $S \subseteq V$ such that the cumulative influence, as measured by a chosen centrality metric, is maximized under a cardinality constraint:

$$\max_{S \subseteq V,\ |S| \leq k} \sum_{i \in S} C(i) \tag{1}$$

Here, $C(i)$ denotes the centrality score of node i, which may correspond to degree, closeness, betweenness, or other centrality measures, and k is a tunable parameter that limits the size of the selected group.

To solve this combinatorial problem, FSA employs a decomposition-based optimization procedure that recursively partitions the network and refines candidate groups based on modularity and intra-group connectivity heuristics. To ensure the structural cohesiveness of the identified focal structures, the following density constraint is enforced:

$$\delta(S) = \frac{2|E_S|}{|S|(|S| - 1)} \geq \tau \tag{2}$$

where E_S denotes the set of internal edges within the subset S, and $\tau \in [0, 1]$ is a predefined threshold governing the minimum allowable density.

By jointly optimizing for aggregate influence (Equation 1) and structural tightness (Equation 2), FSA identifies compact, high-impact node clusters that are well-suited for analyzing influence dynamics, modeling coordinated behavior, and designing structural interventions in social networks.

Centrality-Based Node Slicing: involves selecting top-ranked nodes based on classical centrality measures that capture various notions of structural importance within the network. Specifically, we compute degree centrality (measuring direct connectivity), betweenness centrality (reflecting control over shortest paths), closeness centrality (indicating proximity to all other nodes), and eigenvector centrality (accounting for the influence of neighbors). For each metric, we extract the top-k nodes and treat them as influential slices. This method enables a comparative evaluation of individual node-based influence heuristics against group-based approaches such as Focal Structure Analysis and community detection algorithms.

Community detection algorithms: We apply three widely used methods: the Louvain algorithm, which optimizes modularity through hierarchical aggregation of communities; the Leiden algorithm, an improved variant of Louvain that guarantees well-connected communities and faster convergence; and the Label Propagation algorithm, which iteratively assigns node labels based on majority voting among neighbors, resulting in efficient yet potentially unstable partitions. The output of each algorithm consists of node clusters, from which we extract the top-ranked groups by adopting the network resiliency approach for further analysis. These community-based groupings serve as structural baselines for comparison with focal structures and centrality-driven selections.

These methods yielded non-overlapping groups of nodes. To ensure a fair comparison, centrality-based groups were sampled to match the node counts of top-ranked focal structures.

To identify structurally cohesive groups, we applied a collection of SNA methods, including centrality-based node slicing (degree, betweenness, closeness, and eigenvector) to form centrality based node groups, community detection algorithms (Louvain [7], Leiden [8], and Label Propagation [9]), and FSA [12]. Each method was used to extract groups of nodes for comparison. FSA was treated as one of several structure-based grouping strategies, with focal structures defined as non-overlapping, dense subgraphs $G' = (V', E') \subset G$, ensuring that no two structures fully contain one another. To ensure

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

fair comparison, the size of each centrality-based group was matched to the corresponding focal structure.

E. Network Resiliency Metrics

This section describes the metrics employed to quantify network resilience, providing insight into how structurally critical nodes or groups contribute to the stability and connectivity of the network. Network resilience is a vital concept in understanding the robustness of recommendation systems, as it reflects the system's ability to sustain content flow even when influential nodes or clusters are removed. By measuring resilience, we can identify which nodes or focal structures act as critical bridges or bottlenecks within the network and how their removal may amplify content traps or fragmentation.

1) ***Flow Robustness****:* Flow robustness is a fundamental metric that captures the network's capacity to maintain communication under perturbation [15]. A *flow* in this context refers to a path between any two nodes, and a flow is considered *reliable* if it remains intact despite the removal of certain nodes or edges. Flow robustness therefore quantifies the proportion of such reliable flows relative to all possible flows in the network, offering a comprehensive measure of connectivity resilience.

Mathematically, flow robustness (FR) for a network $G(V, E)$ is given by:

$$FR(G) = \frac{\sum_{i=1}^{n} |C_i|(|C_i| - 1)}{|n|(|n| - 1)}, \qquad 0 \leq FR \leq 1 \quad (3)$$

where C_i represents the i-th connected component of the network and $|C_i|$ is the number of nodes in that component. An FR value of 1 indicates that every node can communicate with every other node, denoting a fully resilient network, whereas a value near 0 implies extreme fragmentation with minimal connectivity. By computing FR before and after removing focal structures or high-centrality nodes, we can assess which groups have the greatest structural influence on maintaining the flow of recommendations across the network.

2) ***Giant Component Ratio****:* The Giant Component Ratio (GCR) is a complementary metric that provides a macroscopic view of network connectivity by measuring the proportion of nodes in the largest connected component (LCC) relative to the total network size. Formally, it is defined as:

$$GCR(G) = \frac{N_{LCC}}{N}, \qquad 0 < GCR < 1 \qquad (4)$$

where N_{LCC} denotes the number of nodes in the largest connected component and N is the total number of nodes in the network. A high GCR value indicates that the majority of the network remains interconnected, reflecting strong structural cohesion. Conversely, a low GCR signals that the network is highly fragmented, suggesting that critical nodes or groups have been removed. GCR is particularly informative when evaluating the effects of targeted removals, such as focal structures or community-based clusters, as it highlights the impact on overall connectivity and the potential isolation of subgroups.

3) ***Isolated Nodes and Cluster Analysis****:* In addition to global connectivity metrics, it is important to examine local fragmentation effects. The number of isolated nodes measures the extent to which nodes become disconnected from the network following the removal of influential structures, providing insight into potential disruptions in information dissemination. Cluster analysis, on the other hand, examines changes in the community structure, revealing how removal of key groups can fracture cohesive clusters into smaller subgroups. Together, these metrics offer a nuanced understanding of how node or cluster removal affects both the micro- and macro-level dynamics of the network, particularly in the context of recommendation systems where structural bottlenecks may amplify content trap effects.

F. Spearman's Rank Correlation Coefficient

To quantify the relationship between different resilience metrics, we employed Spearman's rank correlation coefficient [31], which assesses the strength and direction of a monotonic association between two variables without assuming linearity. This is particularly useful when comparing metrics such as flow robustness and giant component ratio, which may not vary linearly with each other. Spearman's ρ is computed as:

$$\rho = 1 - \frac{6 \sum_{i=1}^{n} (R(X_i) - R(Y_i))^2}{n(n^2 - 1)},$$

where $R(X_i)$ and $R(Y_i)$ represent the ranks of observations in X and Y, d_i is the rank difference, and n is the number of paired observations. In our analysis, we found a strong positive correlation of $\rho = 0.92$ between flow robustness and the giant component ratio, indicating that networks with higher flow resilience also tend to maintain larger connected components. Given this high correlation, we primarily rely on flow robustness to evaluate the structural importance of nodes and groups within the network while also considering the effects on overall network connectivity and cohesion.

G. Network Resilience Assessment

To assess the structural significance of the groups extracted by each method, we conducted a comprehensive network resilience analysis, drawing on methodologies established in prior studies [1]. Specifically, for each structural detection approach, we systematically removed the groups identified by that method from the recommendation network and quantified the resulting fragmentation of the network. Fragmentation was measured in terms of the number of disconnected or isolated clusters emerging after group removal, with higher fragmentation scores indicating a more pronounced disruption to the network's overall cohesion and connectivity. This approach enabled us to evaluate the relative influence of different structural groupings on the flow of information and the structural integrity of the recommendation network. To provide a consistent basis for cross-method comparison, we subsequently ranked the top five groups derived from focal structure analysis (FSA) and from community detection methods according to

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

their respective contributions to network fragmentation. In the case of centrality-based groupings, we selected top-ranked nodes such that the cumulative size of these nodes matched the size of the top five focal structures, ensuring a fair and equitable comparison across all methods. By combining these analyses, we obtained a nuanced understanding of which structural configurations most critically sustain or disrupt the network's connectivity, thereby shedding light on the underlying mechanisms through which content traps may influence user traversal within YouTube's recommendation ecosystem.

H. Topic Modeling and Sensitivity Analysis

To examine the thematic content within network structures, we employed the BERTopic model [23], a state-of-the-art topic modeling approach that can extract interpretable topics from large text corpora. BERTopic was chosen over traditional methods such as Latent Dirichlet Allocation (LDA) [32] and Non-negative Matrix Factorization (NMF) due to its superior ability to capture semantic relationships and contextual nuances, enabling the generation of more refined and granular topics. This capability allowed us to gain deeper insights into the thematic composition of videos within each network structure.

Given the length of YouTube video transcripts, BERTopic's token limitation (maximum 512 tokens) posed a challenge. To address this, transcripts were split into multiple coherent chunks, each under 512 words, with splits occurring at sentence boundaries to preserve contextual integrity. This strategy minimized information loss, reduced noise, and ensured more accurate topic extraction. Each identified topic was then mapped to its corresponding video ID, allowing us to quantify the distribution of topics across videos in each focal structure and assess the diversity or concentration of thematic content.

1) Content Trap Identification: A content trap within a focal structure was operationally defined as a scenario where a single topic dominated more than $\theta\%$ of the videos. Formally, the threshold T for detecting a content trap is expressed as:

$$T = \frac{n_{topic}}{n_{total}} > \theta \tag{5}$$

where

- T represents the detection of a content trap within a network structure,
- n_{topic} denotes the number of videos sharing a specific topic, and
- n_{total} is the total number of videos in the focal structure.

Key network structures exceeding this threshold were classified as containing content traps, highlighting clusters in which the recommendation algorithm disproportionately favored a single topic. Such homogeneity limits exposure to diverse content, potentially guiding users toward repeated engagement with similar or attractor videos. This identification provides insights into the structural and algorithmic mechanisms that contribute to content traps and their implications for user experience.

2) Sensitivity Analysis: To rigorously evaluate the degree of topical homogeneity within the structurally identified groups, we performed a sensitivity analysis over a range of thresholds spanning from 0.4 to 0.7. In this framework, a focal structure was deemed topically homogeneous if a single BERTopic-generated topic encompassed at least a proportion $\theta \in [0.4, 0.7]$ of its constituent videos. Thresholds set below 0.4 were found to be overly permissive, often resulting in false positives by categorizing broadly themed or loosely connected groups as homogeneous. Conversely, thresholds above 0.7 proved excessively stringent, excluding many groups that, while not perfectly uniform, nonetheless exhibited significant thematic coherence indicative of potential content traps. By selecting the threshold range of 0.4–0.7, we established a balanced and meaningful criterion that captures genuine instances of thematic dominance without overgeneralizing or being unduly restrictive. This systematic exploration of threshold sensitivity not only facilitated a robust assessment of content homogeneity but also enhanced the interpretability and reliability of our content trap detection methodology, ensuring that the identified clusters represent substantively cohesive and structurally relevant thematic groupings within the recommendation network.

IV. Experimental Results and Analysis

We evaluated each structural detection method by quantifying the number of their top ($k = 5$) extracted groups that exhibited high topical uniformity across varying thresholds ($\theta \in \{0.4, 0.5, 0.6, 0.7\}$). A group was labeled as a *potential content trap* if a single BERTopic-generated topic accounted for at least θ proportion of the group's videos. This evaluation allowed us to assess the alignment between structural cohesion and semantic homogeneity in the recommendation network.

Figure 3 illustrates the number of potential content traps identified by each method across different thresholds. FSA consistently outperformed community detection algorithms (Louvain, Leiden, and Label Propagation) in surfacing groups with high thematic concentration. Notably, under moderate thresholds ($\theta = 0.5, 0.6$), FSA also surpassed several centrality-based groupings. At the lower threshold ($\theta = 0.4$), both Eigenvector centrality and FSA identified five highly homogeneous groups, reaching the maximum detectable under our ranking constraint ($k = 5$). These results suggest that FSA is robust in detecting cohesive groups even when varying the sensitivity threshold, highlighting its utility in identifying latent content traps.

Table II provides a detailed summary of the top $k = 3$ groups ranked by each method, showing node counts and the size of the dominant topic within each group. FSA's top-ranked group comprised 127 videos, with 108 associated with a single topic, resulting in a dominance score of 85%. This demonstrates FSA's ability to surface clusters where algorithmic recommendation can disproportionately favor a single topic, potentially constraining user exposure to diverse content. In contrast, while closeness centrality and Louvain occasionally captured strong topic clusters in lower-ranked

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

TABLE II. NODE COUNT AND MAX TOPIC SIZE FOR TOP 5 NETWORK STRUCTURES ACROSS SNA APPROACHES

Rank Metric	Betweenness	Closeness	Degree	EigenVector	FSA	Label Prop.	Leiden	Louvain
Rank 1 NodeCount	127	127	127	127	127	89	94	92
Rank 1 MaxTopicSize	50	104	59	60	**108**	35	49	27
Rank 2 NodeCount	32	32	32	32	32	57	88	91
Rank 2 MaxTopicSize	23	25	27	**31**	17	16	28	45
Rank 3 NodeCount	28	28	28	28	28	51	87	88
Rank 3 MaxTopicSize	26	7	9	12	**21**	31	44	28

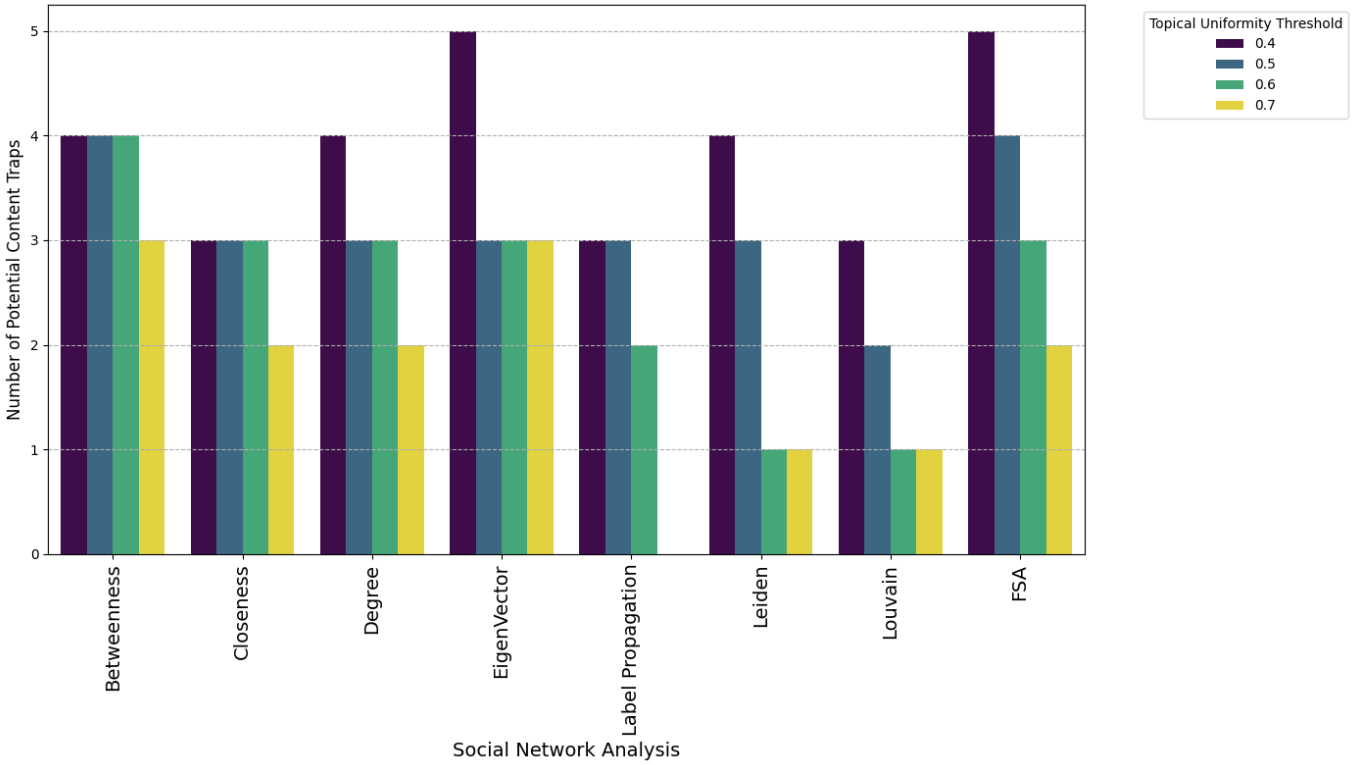


Figure 3. Number of potential content traps per method under varying topical uniformity thresholds.

groups, their performance was inconsistent. For instance, Louvain's top-ranked group contained only 27 videos under a single topic, despite better performance in subsequent ranks. Overall, FSA demonstrated stable topical cohesion across all ranks, reinforcing its effectiveness in identifying structurally and semantically cohesive content traps.

To visually contextualize these findings, Figures 4, 5, and 6 present network visualizations of the top three focal structures identified via FSA. Red nodes indicate dominant topics, while blue nodes represent associated videos within the respective group. The overwhelming dominance of a single topic within each group underscores the presence of tightly bound thematic pockets, directly reflecting potential content traps. By examining these structures, we observe how FSA captures not only densely connected subgraphs but also semantically coherent content clusters, illustrating the interplay between network topology and content homogeneity.

These results indicate that FSA, despite being primarily structurally grounded, excels in surfacing latent thematic attractors within recommendation networks. Unlike community detection methods, which often extract large but semantically heterogeneous clusters, or centrality-based methods that emphasize hub nodes without guaranteeing topical coherence, FSA identifies smaller, self-reinforcing topical pockets. The practical implication is that content traps may emerge in seemingly modest subgraphs where high structural connectivity aligns with strong thematic homogeneity. Recognizing such focal structures provides actionable insight into how recommendation algorithms can inadvertently restrict content diversity, potentially shaping user engagement patterns and reinforcing specific themes over time.

V. CONCLUSION AND FUTURE WORK

In this study, we proposed a comprehensive network and topic-based framework for detecting content traps within

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

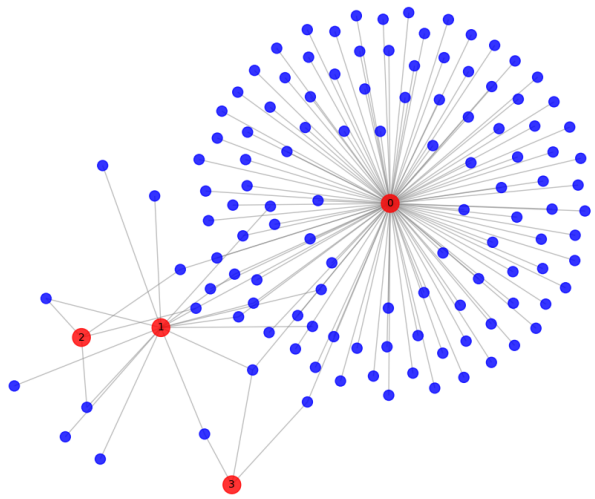


Figure 4. Network visualization of Ranked 1 focal structure showing red topic nodes linked to blue video nodes leading to a potential content trap.

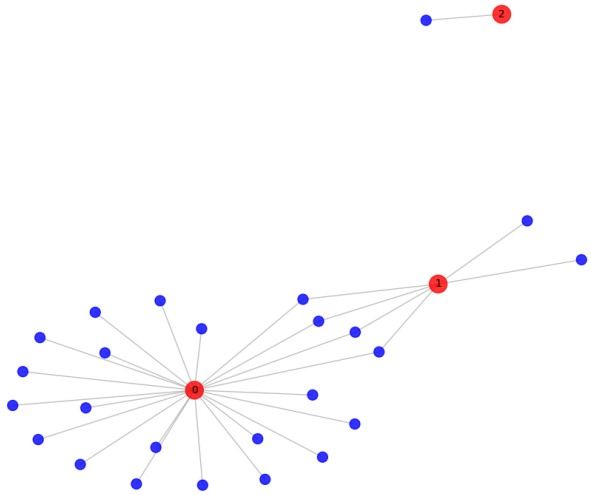


Figure 6. Network visualization of Ranked 3 focal structure showing red topic nodes linked to blue video nodes leading to a potential content trap.

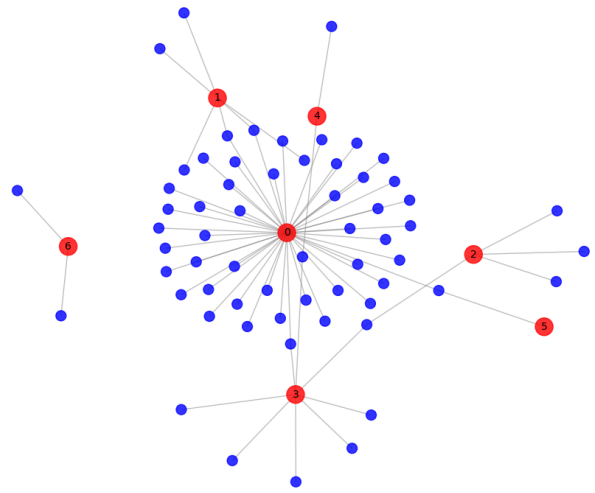


Figure 5. Network visualization of Ranked 2 focal structure showing red topic nodes linked to blue video nodes leading to a potential content trap.

YouTube's recommendation network. By leveraging Focal Structure Analysis (FSA) alongside a suite of traditional social network analysis (SNA) approaches, we provided a nuanced evaluation of how structural cohesion relates to thematic consistency in recommendation networks. Our methodology incorporates a sensitivity-based assessment of topical uniformity, enabling a fine-grained understanding of how closely related content clusters emerge and persist within the network. The results of our analysis reveal that certain structurally cohesive groups, particularly those identified through focal structures, consistently exhibit higher thematic alignment compared to groups derived from conventional community detection algorithms. In several cases, these focal structures even outperform centrality-based groupings across varying threshold levels, highlighting the importance of considering both structural and topical dimensions when analyzing recommendation networks. These findings underscore the potential of focal structures as a robust tool for detecting algorithmically reinforced content traps, offering insights into how recommendation systems may unintentionally guide users toward narrow and repetitive content.

While our current study primarily focused on structural and thematic properties, there exist several promising avenues for future research. Incorporating temporal dynamics could allow researchers to examine how content traps evolve over time, capturing the emergence, persistence, and potential decay of highly cohesive content clusters. Similarly, integrating sentiment analysis and user behavior metrics could provide a richer understanding of how these traps influence user engagement, perception, and decision-making within the platform. Beyond YouTube, extending the framework to other datasets and content platforms would enable comparative analyses, offering valuable insights into the similarities and differences in algorithmic content curation practices across platforms. Such extensions could further inform the design of intervention strategies aimed at mitigating the influence of content traps, ultimately contributing to more diverse and balanced information ecosystems. Taken together, our study lays the groundwork for a multi-dimensional approach to understanding and addressing the structural and thematic mechanisms underlying algorithmic recommendation systems.

This study presents a network and topic-based framework for detecting content traps in YouTube's recommendation network. We evaluate FSA alongside multiple SNA approaches using a sensitivity-based assessment of topical uniformity. Our findings demonstrate that certain structurally cohesive groups, particularly those identified via focal structures, exhibit higher

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

thematic consistency than those derived from community detection methods, and in some cases, outperform centrality-based groupings across varying threshold levels.

While our focus was on structural and thematic properties, future work may incorporate temporal dynamics, sentiment, or user behavior to better understand the persistence and impact of content traps. Additionally, applying this framework across other datasets and other platforms could offer comparative insights into algorithmic content curation.

Acknowledgements

This research is funded in part by the U.S. National Science Foundation (OIA-1946391, OIA-1920920), U.S. Office of the Under Secretary of Defense for Research and Engineering (FA9550-22-1-0332), U.S. Army Research Office (W911NF-23-1-0011, W911NF-24-1-0078, W911NF-25-1-0147), U.S. Office of Naval Research (N00014-21-1-2121, N00014-21-1-2765, N00014-22-1-2318), U.S. Air Force Research Laboratory, U.S. Defense Advanced Research Projects Agency, the Australian Department of Defense Strategic Policy Grants Program, Arkansas Research Alliance, the Jerry L. Maulden/Entergy Endowment, and the Donaghey Foundation at the University of Arkansas at Little Rock. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations. The researchers gratefully acknowledge the support.

References

[1] M. M. I. Bhuiyan, S. Shajari, and N. Agarwal, "Resilience and node impact assessment in youtube commenter networks leveraging focal structure analysis," *The Eleventh International Conference on Human and Social Analytics (HUSO 2025)*, 2025.

[2] M. M. I. Bhuiyan and N. Agarwal, "Identification and characterization of content traps in youtube recommendation network," *The Seventeenth International Conference on Information, Process, and Knowledge Management (eKNOW)*, 2025.

[3] M. M. I. Bhuiyan and N. Agarwal, "Detecting algorithmic homophily in recommendation graphs via weighted topic distribution," *2025 IEEE 37th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2025.

[4] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social Networks*, vol. 1, no. 3, pp. 215–239, 1978.

[5] U. Brandes, "A faster algorithm for betweenness centrality," *Journal of Mathematical Sociology*, vol. 25, no. 2, pp. 163–177, 2001.

[6] P. Bonacich, "Power and centrality: A family of measures," *American Journal of Sociology*, vol. 92, no. 5, pp. 1170–1182, 1987.

[7] 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, P10008, 2008.

[8] V. A. Traag, L. Waltman, and N. J. van Eck, "From louvain to leiden: Guaranteeing well-connected communities," *Scientific Reports*, vol. 9, no. 1, pp. 1–12, 2019.

[9] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, no. 3, p. 036 106, 2007.

[10] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3–5, pp. 75–174, 2010.

[11] F. Şen, R. Wigand, N. Agarwal, S. Tokdemir, and R. Kasprzyk, "Focal structures analysis: Identifying influential sets of individuals in a social network," *Social Network Analysis and Mining*, vol. 6, pp. 1–22, 2016.

[12] M. Alassad, M. N. Hussain, and N. Agarwal, "Comprehensive decomposition optimization method for locating key sets of commenters spreading conspiracy theory in complex social networks," *Central European Journal of Operations Research*, vol. 30, no. 1, pp. 367–394, 2022.

[13] V. B. Bertoni, T. A. Saurin, and F. S. Fogliatto, "How to identify key players that contribute to resilient performance: A social network analysis perspective," *Safety Science*, vol. 148, p. 105 648, 2022.

[14] X. Liu *et al.*, "Network resilience," *Physics Reports*, vol. 971, pp. 1–108, 2022.

[15] M. J. Alenazi and J. P. Sterbenz, "Comprehensive comparison and accuracy of graph metrics in predicting network resilience," in *2015 11th International Conference on the Design of Reliable Communication Networks (DRCN)*, IEEE, 2015, pp. 157–164.

[16] B. Bollobás and B. Bollobás, *Random graphs*. Springer, 1998.

[17] E. Pariser, *The filter bubble: How the new personalized web is changing what we read and how we think*. Penguin, 2011.

[18] A. Bechmann and K. L. Nielbo, "Are we exposed to the same "news" in the news feed? an empirical analysis of filter bubbles as information similarity for danish facebook users," *Digital Journalism*, vol. 6, no. 8, pp. 990–1002, 2018.

[19] H. Hosseinmardi *et al.*, "Evaluating the scale, growth, and origins of right-wing echo chambers on youtube," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 15, 2021, pp. 197–208.

[20] M. H. Ribeiro, R. Ottoni, R. West, V. A. Almeida, and W. Meira, "Auditing radicalization pathways on youtube," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 2020, pp. 131–141.

[21] S. Koehler, M. Stadelmaier, *et al.*, "Topic diversity in recommender systems," *Information Processing & Management*, vol. 58, no. 3, p. 102 536, 2021.

[22] K. Zhou, Y. Yang, X. Wang, and N. Z. Gong, "Recommender systems: A survey on privacy, fairness, and bias," *ACM Transactions on Recommender Systems*, vol. 1, no. 1, pp. 1–42, 2020.

[23] M. Grootendorst, "Bertopic: Neural topic modeling with a class-based tf-idf procedure," *arXiv preprint arXiv:2203.05794*, 2022.

[24] M. I. Gurung, M. M. I. Bhuiyan, A. Al-Taweel, and N. Agarwal, "Decoding youtube's recommendation system: A comparative study of metadata and gpt-4 extracted narratives," in *Companion Proceedings of the ACM on Web Conference 2024*, 2024, pp. 1468–1472.

[25] M. I. Gurung, N. Agarwal, and M. M. I. Bhuiyan, "How does semiotics influence social media engagement in information campaigns?" *Proceedings of the 58th Hawaii International Conference on System Sciences*, 2025.

[26] M. I. Gurung, N. Agarwal, M. M. I. Bhuiyan, and D. Poudel, "Symbolic signals on instagram: How visual media shapes engagement, emotion, trust, and diffusion," *Social Network Analysis and Mining*, vol. 15, no. 1, pp. 1–16, 2025.

[27] M. M. I. Bhuiyan and N. Agarwal, "Structure, semantics, and attraction: Analyzing homophily in recommender networks," in *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*, Springer, 2025, pp. 173–182.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

[28] R. Finlay, "The voyages of zheng he: Ideology, state power, and maritime trade in ming china," *Journal of the Historical Society*, vol. 8, no. 3, pp. 327–347, 2008.

[29] Google Developers, *YouTube Data API*, https://developers.google.com/youtube/v3, Accessed: April. 10, 2025, No Date.

[30] M. C. Cakmak and N. Agarwal, "High-speed transcript collection on multimedia platforms: Advancing social media research through parallel processing," in *2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2024, pp. 857–860. DOI: 10.1109/IPDPSW63119.2024.00153.

[31] C. Spearman, "The proof and measurement of association between two things," *The American Journal of Psychology*, vol. 100, no. 3/4, pp. 441–471, 1987.

[32] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, no. Jan, pp. 993–1022, 2003.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Developing Domain-Specific Threat Models for Greater Software Security

Aspen Olmsted
School of Computer Science and Data Science
Wentworth Institute of Technology
Boston, MA 02115
olmsteda@wit.edu

***Abstract*— Developing secure software applications in modern, complex environments presents significant challenges, as traditional threat modeling approaches often fall short in addressing domain-specific vulnerabilities. This paper introduces and discusses three novel, domain-specific threat models designed to enhance secure software development: BIRFS, CRIRTA, and PERTD. BIRFS (Bias, Input, Reasonable, Forensics, Sensitive) is a specialized threat model tailored for software systems that leverage Artificial Intelligence and Machine Learning algorithms, focusing on unique risks arising from data inputs, model behavior, and algorithmic biases. CRIRTA (Column, Row, Inference, Relationship, Table, Availability) provides a comprehensive framework for identifying and mitigating security threats in database applications, moving beyond generic data flow analysis to address specific database vulnerabilities. PERTD (Partition, Execution, Requisite, Timing, Data) is designed for Cloud Application Threat Modeling, emphasizing the distinct security challenges inherent in cloud environments, including distributed architecture, shared tenancy, and dynamic resource allocation. Collectively, these models aim to enable proactive risk identification during the design phase, enabling the implementation of targeted mitigation strategies earlier in the software development lifecycle. By moving beyond a sole focus on malicious user threats, these models address a broader spectrum of vulnerabilities stemming from poor design, misunderstood use cases, and environmental changes, thereby contributing to more robust and resilient software systems across diverse domains.**

Keywords- cyber-security; software engineering; software development lifecycle

I. INTRODUCTION

The landscape of software development has undergone a profound transformation in recent decades, driven by the proliferation of cloud computing, the pervasive integration of Artificial Intelligence (AI) and Machine Learning (ML) algorithms, and the ever-increasing reliance on complex database systems. While these advancements have unlocked unprecedented capabilities and efficiencies, they have simultaneously introduced a new generation of security challenges. Traditional approaches to secure software development, often centered on identifying and mitigating threats from malicious actors, are proving increasingly inadequate. Many contemporary vulnerabilities stem not from external attacks, but from inherent design flaws, a lack of understanding of system use cases, and insufficient planning for dynamic environmental changes.

In this context, the need for robust and adaptable threat modeling has become paramount. Threat modeling is a proactive security practice that enables developers and security professionals to identify potential threats and vulnerabilities early in the software development lifecycle, thereby allowing for the implementation of effective mitigation strategies before code is even written. However, the one-size-fits-all approach to threat modeling often fails to capture the nuanced risks specific to particular domains or technological paradigms. For instance, the security considerations for a cloud-native application differ significantly from those of an AI-driven system or a highly sensitive database.

This paper addresses this critical gap by introducing and discussing three novel, domain-specific threat models designed to enhance the security posture of modern software applications. We propose:

BIRFS (Bias, Input, Reasonable, Forensics, Sensitive): A specialized threat model meticulously crafted for software systems that integrate Artificial Intelligence and Machine Learning algorithms. BIRFS extends traditional security concerns to encompass unique risks such as data poisoning, model manipulation, algorithmic bias, and fairness issues, which generic threat models often overlook.

CRIRTA (Column, Row, Inference, Relationship, Table, Availability): A comprehensive framework developed to identify and mitigate security threats specifically within database applications. CRIRTA moves beyond conventional data flow analysis to address the unique vulnerabilities inherent in data storage, retrieval, and management, ensuring robust data protection and system resilience.

PERTD (Partition, Execution, Requisite, Timing, Data): A dedicated threat model for Cloud Application Threat Modeling. PERTD focuses on the distinct security challenges posed by cloud environments, including shared tenancy, complex distributed architectures, API security, and data privacy concerns across multi-tenant infrastructures.

By adopting these domain-specific models, organizations can achieve a more granular and practical approach to identifying and mitigating risks, leading to the development of inherently more secure, resilient, and trustworthy software systems. The subsequent sections of this paper will delve into the details of each of these models, outlining their principles, methodologies, and practical applications, followed by a discussion of their collective impact on the future of secure software development. This work is an extension of a previous published conference paper [1].

The organization of the paper is as follows. Section II describes the related work and the limitations of current methods. Section III describes workflow engines used in our motivating example of a distributed cloud application. Section IV discusses a current Threat Modeling technique called STRIDE. Section V discusses an alternative Threat modeling technique called DREAD. In Section VI, we give a motivating example from our distributed system modeling. Section VII describes our distributed modeling methodology. In Section VIII, we provide a motivating example from our database system modeling. Section IX describes our database modeling methodology. In Section X, we give a motivating example for

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

our AI/ML system modeling. Section XI describes our AI/ML modeling methodology. We conclude and discuss future work in Section XII.

II. RELATED WORK

Functional requirements can be defined and represented in various ways. While these requirements serve as the foundation for software development, non-functional requirements (NFRs) provide the essential guidelines for coding implementation. Many authors have examined NFRs and the challenges of incorporating them into the design process. Pavlovski and Zou [1] NFRs are defined as specific behaviors and operational constraints, including performance expectations and policy limitations. Despite many discussions surrounding them, they are often not given the attention they deserve.

Glinz [2] suggests categorizing functional and non-functional requirements to ensure that they are inherently considered during application development. Alexander [3] points out that the language used to describe requirements is essential, noting that words ending in "-ility," such as reliability and verifiability, often refer to NFRs. Much of this research focuses on identifying NFRs. Our work builds on these foundations by applying domain-specific models using our proposed modeling technique.

Ranabahu and Sheth [4] explore four different modeling semantics to represent cloud application requirements: data, functional, non-functional, and system. Their work primarily addresses functional and system requirements, with some overlap in non-functional requirements from a system perspective. They built upon research conducted by Stuart, who defined semantic modeling languages for modeling cloud computing requirements throughout the three phases of the cloud application life cycle: development, deployment, and management. Our work fills in the gap regarding the semantic category of non-functional requirements.

Ranabahu and Sheth [4] use Unified Modeling Language (UML) to model only functional requirements. UML [6] is a standardized notation for representing software systems' interactions, structures, and processes. It consists of various diagram types, with individual diagrams linked to different perspectives of the same part of a software system. We utilize UML to express non-functional requirements as a secondary step following the PERTD models.

Integrating UML Sequence, Activity, and Class diagrams can enhance the semantics of our models. UML offers extensibility mechanisms that allow designers to add new semantics to a model. One such mechanism is a stereotype, which helps extend the vocabulary of UML to represent new model elements. Traditionally, software developers interpret these semantics and manually translate them into program code in a hard-coded manner. In our book [6], we marry the models generated by each phase of the software development lifecycle into with threat modeling and risk mitigation techniques.

The Object Constraint Language (OCL) [8] is part of the official Object Management Group (OMG) standard for UML. An OCL constraint specifies restrictions for the semantics of a UML specification and is considered valid as long as the data is consistent. Each OCL constraint is a declarative statement in the design model that signifies correctness. The expression of the constraint occurs at the class level, while enforcement happens at the object level. Although OCL has operations to observe the system state, it does not include functions to modify it.

JSON [9] stands for "JavaScript Object Notation," a simple data interchange format that began as a notation for the World Wide Web. Since most web browsers support JavaScript, and JSON is based on JavaScript, it is straightforward to support it there, which stands for "JavaScript Object Notation," a simple format used for data interchange that originated as a notation for the World Wide Web. Since most web browsers support JavaScript and JSON is based on JavaScript, it is easy to work with in web environments. Many cloud-based web services now exchange data in JSON format. JSON Schemas [10] define correctness for data passed in JSON format. We utilize an extended form of JSON schemas on the aggregated data from several web services.

Our contribution to secure software development involves new Threat Modeling techniques, coupled with modeling standards, such as UML and OCL, utilizing their extensibility mechanism of stereotypes to model non-functional requirements effectively.

III. WORKFLOW ENGINES

Workflow engines like Zapier [11] and Power Automate [12] are powerful automation tools that enable users to create and manage workflows for integrating and automating tasks across various applications and services, whether in the cloud or on-premises.

Zapier is a popular cloud-based automation platform that allows users to connect to different web applications and automate their workflows. It operates on a simple "trigger-action" model, where an event in one application triggers an action in another. Users can create "Zaps" (automated workflows) by selecting a trigger and defining the subsequent actions. For example, when a new email arrives in Gmail (trigger), the attachments can be automatically saved to Google Drive (action).

Zapier supports numerous apps and services, including well-known ones like Gmail, Slack, Salesforce, and Trello. It features a user-friendly interface, pre-built Zap templates for everyday use cases, and advanced options like filters, delays, and data transformations. Additionally, Zapier allows for multi-step Zaps, making it possible to create complex workflows with multiple actions and conditions.

Power Automate is a cloud-based service from Microsoft that allows users to automate workflows and integrate applications and services within the Microsoft ecosystem and beyond. It offers connectors for various applications, including Microsoft 365 apps (such as Outlook and SharePoint), Dynamics 365, Azure services, and third-party services like Salesforce, Dropbox, and Twitter.

Power Automate features a visual design interface where users can create workflows by combining triggers, actions, and conditions. Available triggers include email arrivals, button clicks, data changes, and scheduled events. Actions can involve sending emails, creating tasks, updating records, etc. Power Automate offers advanced capabilities like loops, parallel branches, and approval processes.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Both Zapier and Power Automate provide extensive libraries of pre-built templates and connectors, making it easier for users to begin automating tasks. They offer options to monitor and manage workflows, handle errors, and track activity logs. These platforms cater to users with varying technical expertise, from business users to developers, and help automate repetitive tasks, streamline processes, and enhance productivity.

IV. STRIDE Threat Modeling

STRIDE [12] is a threat modeling framework that offers a structured approach for identifying and analyzing threats in software systems. It helps security practitioners and developers understand potential risks and implement appropriate security controls. STRIDE is an acronym representing six categories of threats:

1. Spoofing Identity: This category involves attackers impersonating legitimate users or entities to gain unauthorized access or deceive the system. For instance, attackers may spoof a user's identity by stealing credentials or manipulating authentication mechanisms.

2. Tampering with Data: Tampering threats involve the unauthorized modification or alteration of data within the system. Attackers may tamper with data in transit, modify stored data, or manipulate system parameters to achieve desired outcomes. For example, an attacker could alter the contents of a database, inject malicious code into an application, or change parameters to bypass security checks.

3. Repudiation: Repudiation threats allow users to deny their involvement in specific transactions or activities, posing challenges for auditing and accountability. For instance, an attacker might modify logs or manipulate transaction records to evade detection or deny their actions.

4. Information Disclosure: This category addresses threats related to unauthorized exposure or disclosure of sensitive information. Attackers may exploit vulnerabilities to access confidential data, such as personal information, financial records, or intellectual property. This can happen through insecure data transmission, weak access controls, or information leakage via error messages.

5. Denial of Service: Denial of Service (DoS) threats aim to disrupt or degrade a system's availability or performance. Attackers may overload resources, exhaust system capacity, or exploit vulnerabilities to cause a service outage, rendering the system unresponsive or unusable for legitimate users.

6. Elevation of Privilege: Elevation of Privilege threats involve attackers gaining unauthorized access to higher privileges or permissions than they should have. By exploiting vulnerabilities or design flaws, attackers can bypass security controls and gain elevated access rights, leading to unauthorized data access, system compromise, or further exploitation.

When applying the STRIDE framework, security practitioners and developers analyze the software system from the perspective of each threat category. They identify potential vulnerabilities and develop corresponding mitigation strategies to address the threats. This analysis facilitates informed decisions regarding security controls, system design improvements, and the prioritization of security efforts.

V. DREAD Threat Modeling

DREAD is a threat modeling framework designed to assess and prioritize software vulnerabilities based on their potential impact. The acronym DREAD stands for five key factors used to evaluate threats:

1. Damage Potential: This factor refers to the extent of harm that could be caused if a vulnerability is exploited. It evaluates the impact, which can range from minor inconveniences to severe consequences like data breaches, system compromises, or financial losses.

2. Reproducibility: This measures how easily an attacker can reproduce or exploit a vulnerability. Vulnerabilities that are consistently easy to exploit are considered more dangerous than those that require complex or unpredictable conditions for exploitation.

3. Exploitability: This factor assesses the level of skill or effort needed to exploit a vulnerability. Vulnerabilities easily exploited with readily available tools or techniques pose a higher risk. Conversely, vulnerabilities that are difficult to exploit or require specialized knowledge are considered lower risk.

4. Affected Users: This evaluates the number of users or systems a vulnerability could impact. A vulnerability affecting numerous users or critical systems is considered more significant than one impacting only a limited subset of users.

5. Discoverability: This assesses how likely an attacker is to find the vulnerability. Vulnerabilities that are easily discoverable—through public disclosures, known attack techniques, or automated scanning tools—are riskier than those that are harder to find or require advanced reconnaissance.

Using the DREAD framework, each factor is scored on a scale from 0 to 10, with 0 being the least concerning and ten being the most critical. These scores help prioritize vulnerabilities and allocate resources for mitigation efforts. Higher scores indicate a higher priority for addressing the identified vulnerabilities.

While DREAD is a valuable tool for assessing and prioritizing vulnerabilities based on their potential impact, it should be used alongside other threat modeling techniques and considerations to ensure a comprehensive security analysis and informed decision-making.

VI. Distributed Model Motivating Example

The challenge with the STRIDE and DREAD threat models is that they primarily focus on vulnerabilities associated with malicious user activities. However, many risks arise from architecture, the environment, or human error.

Consider a common architecture used by many businesses today: data generated by an online transaction processing (OLTP) system, either stored on-premises or logically on-premises, is synchronized to a cloud system considered off-premises and beyond the organization's control. This scenario is not uncommon in today's business landscape.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Consider a large performing arts venue employing a local SQL Server-based system for ticketing and donation transactions. Meanwhile, its marketing department uses a cloud-based email and SMS marketing system. The OLTP data must be extracted, translated, uploaded, and loaded regularly for the marketing system to function correctly.

Various issues can arise when multiple processes and data are transferred across networks that span domain boundaries. A UML activity diagram illustrates the steps involved in moving data from the on-premises OLTP system to the cloud-based system used by the marketing team. This model shows that activities occur in both environments. The challenge with the STRIDE and DREAD threat models is that the vulnerabilities modeled and the matching remediations target malicious user activities. Many times, risks come from architecture, environment, or human error.

A motiving example is an architecture that is used in many businesses today where data that is generated in OLTP systems that are either stored on-premises or logically on-premises is synchronized to a cloud system that is considered off-premises and outside the domain of control of the organization. To understand this better, consider a large performing arts venue that utilizes a local SQL Server-based system to process ticketing and donation transactions. The marketing department uses a cloud-based system for email and SMS marketing. The OLTP data must be extracted, translated, uploaded, and loaded regularly for the marketing system to be functional.

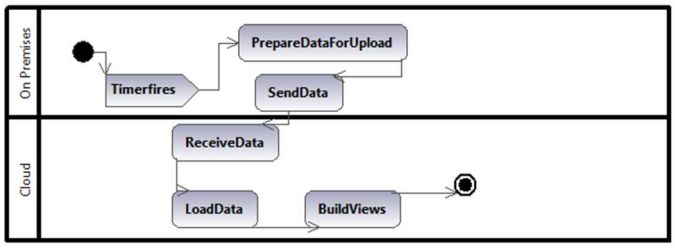


Figure 1 - Upload Activity

Understanding the data transfer process is crucial to prevent potential risks. Figure 1 shows a UML activity diagram for moving data from the on-premises OLTP system to the cloud-based system used by the marketing folks. In the model, you will see that activities happen in both partitions.

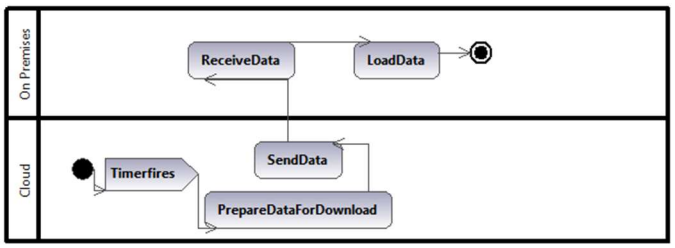


Figure 2 - Download Activity

Table 1 - Upload Activity STRIDE Model

Action	S	T	R	I	D	E
Timerfires						
PrepareDataForUpload						
SendData	X	X		X	X	
ReceiveData						
LoadData						
BuildViews						

Figure 2 presents a model outlining the execution path for retrieving data from the cloud system. The data includes sending activity for both emails and SMS text messages. This sending activity can be substantial, encompassing tuples for sends, opens, clicks, and bounces. Additionally, information regarding communication preferences and unsubscribed data is retrieved.

The marketing department requires service availability and data integrity for its business operations. For instance, NFRs could specify that the system must be available 99.999% of the time or that the data must be no more than 24 hours old. Whenever a distributed system is proposed, a model should be developed to represent these NFRs and the threats to the system's ability to meet them.

Unfortunately, the focus of STRIDE and DREAD on malicious users does not adequately address many of the risks in our motivating example. Table 1 Illustrates a STRIDE model corresponding to the update activity depicted in Figure 1, while Table 2 shows the STRIDE model related to the download activity from Figure 2. In the STRIDE model, actions are at risk from malicious users; however, many steps are also vulnerable to environmental issues that can impact the system's availability and integrity. Examples of these issues include network and system outages, concurrent computational usage on equipment, and a lack of control over the quality of source data.

Table 2 - Download Activity STRIDE Model

Action	S	T	R	I	D	E
Timerfires						
PrepareDataForDownload						
SendData	X	X		X	X	
ReceiveData						
LoadData						

VII. PERTD MODEL

We developed the PERTD Model to better assess the risks associated with distributed applications [13]. This model addresses four main environmental risk categories for distributed systems:

1. PARTITION

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Activities vulnerable to partition errors will fail if a network is partitioned between on-premises devices and the cloud. Risk reduction strategies include:

- Pausing the complete workflow and retrying

- Utilizing previous execution data

- Employing alternative data sources

2. EXECUTION

Activities that are susceptible to execution errors may fail due to ambiguous code requirements, leading to runtime or tooling errors. For example, queries that generate data might fail with future datasets. Risk reduction measures include:

- Utilizing previous execution data (most systems create a copy before execution)

- Using alternative data sources

3. REQUISITE

Table 3 - Upload Activity PERTD Model

Action	P	E	R	T	D
Timerfires		X			
PrepareDataForUpload		X			
SendData	X	X	X	X	
ReceiveData	X	X	X	X	
LoadData		X	X	X	X
BuildViews		X	X		

Activities with requisite vulnerabilities depend on prerequisite activities. If a prerequisite fails, the dependent activity becomes stale. Risk reduction can involve:

- Utilizing previous execution data

- Employing alternative data sources

4. TIMING

Activities at risk due to timing need to finish within a specific time window or under a threshold duration. Risk reduction strategies include:

- Utilizing previous execution data (most systems create a copy before execution)

- Using alternative data sources

5. DATA

Activities are at risk because data are often combined from different sources. Unfortunately, schema correctness specifiers only apply to one data source. Risk reduction strategies include:

- Additional workflow steps to verify correctness

In Tables 3 and 4, we apply our PERTD model to analyze risks associated with uploading and downloading activities. The PERTD model captures significantly more risks than the STRIDE model.

After identifying NFRs in the PERTD model, we develop standard UML Class, Sequence, and Activity Diagrams. The threats to the system are modeled using UML stereotypes. UML stereotypes extend the standard UML language by introducing custom or specialized elements, properties, and behaviors. They allow adding domain-specific annotations, constraints, or semantics to UML elements, enhancing expressiveness and tailoring modeling to specific contexts. Stereotypes are indicated by guillemets (<< >>) placed above the name of the stereotyped element.

Stereotypes can be attached to classes, messages, attributes, and activities. With the PERTD model, we incorporated the four risk categories as stereotypes: <<PARTITION>>, <<EXECUTION>>, <<REQUISITE>>, <<TIMING>> and <<DATA>>. These stereotypes are then tagged to messages in UML Sequence and Activity diagrams, while data classes and individual attributes can also be tagged if they are susceptible to these risks.

Table 4 - Download Activity PERTD Model

Action	P	E	R	T	D
Timerfires		X			
PrepareDataForDownload		X			
SendData	X	X	X	X	
ReceiveData	X	X	X	X	
LoadData		X	X		X

Additionally, OCL is included to specify invariants that can define additional semantics related to the correctness of method calls, classes, or attributes. For instance, if data in a particular class must be no older than three days, this can be expressed using the last_update attribute.

To verify data from when it is vulnerable, we utilize an extended version of JSON Schemas [10]. Our extension allows the Schema to reference different data sources. JSON schema supports a CONTAINS operator to verify the existence of an element in a collection. We added a CONTAINEDIN operator to span across schemas represented by different data sources in the distributed system. We also added a NOTCONTAINEDIN to verify the absence of an element. Figure 3 shows two sample schemas. The top schema is a simplified version of a patron, the bottom schema is a simplified version of a ticket. They share an email field which is designated in the tickets schema to require the existence in the patron data.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

```
{
  "$id": "https://aspenolmsted.com/patron.schema.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "array",
  "items": {
    "type": "object",
    "properties":{
      "name": {"type": "string" },
      "email": {"type": "string" }
    },
  }
}

{
  "$id": "https://aspenolmsted.com/tickets.schema.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "array",
  "items": {
    "type": "object",
    "properties": {
      "event": {"type": "string" },
      "email": {"type": "string",
        "containedin' : "https://aspenolmsted.com/patron.schema.json"},
      "tickets": {"type": "integer"}
  },
  }
}
```

Figure 3 - Sample Schema

To mitigate the risk of data integrity issues, we validate the data against the specified schemas as part of the data workflow.

VIII. Database Model Motivating Example

The challenge with the STRIDE and DREAD threat models for database application security is that the vulnerabilities modeled and the matching remediations target malicious user activities. Many times, risks come from architecture, environment, or human error.

To explore a motivating example, consider an event ticketing system used in a consortium of performing arts centers, such as a symphony, opera, ballet, and Broadway venue. A transaction processing (OLTP) system stores the data generated from the patron transactions on-premises for ticket purchases. Data needs to be logically partitioned so patrons are shared, but only the transaction data appropriate to the organization is visible on software screens and reports. Patrons can go to individual constituent box offices to work with an agent on transactions related to that constituent organization, or they can use self-service on a web or mobile interface that allows them to purchase tickets to any constituent organization or see all their individual transaction histories.

To illustrate a weakness in the traditional threat models when applied to database security, we picked out four use cases from our motivating example:

- A constituent box office agent accessing a patron's ticket history. In this case, the data rows must be confidential, so only those related to the constituent organization should be visible.
- A constituent box office agent selecting a patron's general admission (GA) seats. In this use case, the agent should only have access to constituent inventory and not exceed capacity.
- A patron accessing their ticket history through a self-service web page. In this case, the data rows must be confidential to just the patron's data and not make other patrons' histories available.
- A patron selects general admission (GA) seats through a self-service web page. In this use case, the patron should not exceed the venue capacity.

Unfortunately, the focus of STRIDE and DREAD on the malicious user does not account for many of the risks in our motivating example. Table **5** shows a STRIDE model to match the four use cases we are analyzing. Table 6 shows the equivalent DREAD model. In both models, we can see a large surface area of vulnerabilities to malicious user attacks. As stated earlier, most security risks to database software and software generally come from not understanding and enforcing security requirements in the software engineering process. This lack of enforcement leaves an application vulnerable to user error and malicious attacks.

Table 5 - STRIDE Model for Database Software

Action	S	T	R	I	D	E
Agent View of Ticket History	X			X	X	X
Agent Reserving GA Seat Selection		X	X		X	X
Self-Service View of Ticket History	X			X	X	X
Self-Service Reserving GA Seat Selection		X	X		X	X

IX. CRIRTA MODEL

We developed the CRIRTA Model to better model the risks associated with database applications. The model covers a database system's seven main environmental risk categories.

A. Column Confidentiality

Activities vulnerable to exposing column confidentiality have data with sensitive data, where some users should have restricted read and write access. Other users will need access to the column values. Risk reduction can include:

- Removing all access at the table level for users who should not have access.
- Creating database views without the column values and granting access to these user groups

B. Row Confidentiality

Activities vulnerable to row confidentiality have user access that must be restricted to specific values for some user groups. Examples include self-service apps where users can only see their data, or departmental users can only see department records. Risk reduction can consist of:

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

- Removing all access at the table level for users who should not have access.
- Creating database views with where statements that restrict the rows that are visible to these user groups

C. Column Inference

Activities vulnerable to column inference allow users to infer other values based on values in these columns. Gender or race are often examples of this when the greater data population is relatively homogenous. Risk reduction can include:

- Limiting access to the column to essential users (see column confidentiality reductions above)

D. Relationship Correctness

Activities vulnerable to relationship correctness issues have correctness that spans a relationship between tables. An example would be an order that needs a shipping address that exists in a customer's address. Risk reduction can include:

- Utilizing database foreign keys
- Utilizing database check constraints with queries that check for the existence of rows or values in related tables.
- Utilizing database triggers

E. Table Correctness

Activities vulnerable to table correctness issues have column values restricted based on other columns or sets of column values. Risk reduction can include:

- Utilizing database check constraints with predicates that check row values
- Utilizing database triggers

F. Availability

Activities vulnerable to availability issues often use locks to access one process at a time. The exclusivity is done with the database isolation level. Risk reduction can include:

- Minimize code with higher restriction levels
- Table design changes

TABLE 6 - DREAD Model for Database Software

Action	D	R	E	A	D
Agent View of Ticket History	2	3	7	1	5
Agent Reserving GA Seat Selection	3	3	9	9	2
Self-Service View of Ticket History	6	6	8	3	5
Self-Service Reserving GA Seat Selection	9	6	9	9	2

We model the risks to the motivating example system in Table **7** - *CRICRTA* Model that utilizes our CRIRTA model. As you can see, the CRIRTA model captures many more risks than the STRIDE and DREAD models.

Table 7 - *CRICRTA* Model

Action	C	R	I	R	T	A
Agent View of Ticket History		X		X	X	
Agent Reserving GA Seat Selection			X	X	X	
Self-Service View of Ticket History		X		X	X	
Self-Service Reserving GA Seat Selection			X	X	X	

After identifying NFRs in the CRIRTA model, standard UML Class, Sequence, and Activity Diagrams are developed. The threats to the system are modeled by utilizing UML Stereotypes. UML (Unified Modeling Language) stereotypes are a way to extend the standard UML language by introducing custom or specialized elements, properties, and behaviors. Stereotypes allow you to add domain-specific annotations, constraints, or semantics to UML elements, making them more expressive and tailored to specific modeling contexts. Stereotypes are denoted by guillemets (<< >>) placed above the name of the element being stereotyped.

Stereotypes can be attached to classes, messages, attributes, and activities. With the CRIRTA model, we added the six CRIRTA categories as stereotypes: << ColumnConfidentiality >>, <<RowConfidentiality >>,<< Column Inference >>,<< RelationshipCorrectness>>,<<TableCorrectness>>,<<Availability>>. These stereotypes are then tagged to messages in UML Sequence and Activity diagrams; data classes and individual attributes can be tagged with the stereotype if the data in the class or attribute is susceptible to the risk.

OCL is added to provide invariants that can specify additional semantics related to the correctness of a method call, class, or attribute. An example is if data in a particular table must have a related table with a specific attribute range.

X. AI/ML MODEL MOTIVATING EXAMPLE

The challenge with the STRIDE and DREAD threat models for applications that consume ML and AI algorithms is that the vulnerabilities modeled and the matching remediations are aimed at malicious user activities attacking a system. Many times, risks come from architecture, environment, or human error.

A motivating example is an architecture that is used in many businesses today, where data that is generated in online transaction processing (OLTP) systems that are either stored on-premises or logically on-premises is synchronized to a cloud system that is considered off-premises and outside the domain of control of the organization. Once the data is in the cloud, it is augmented utilizing AI or ML algorithms. To understand this better, consider a large performing arts venue that operates a local SQL Server-based system to process ticketing and

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

donation transactions. The marketing department uses a cloud-based system for email and SMS marketing. The OLTP data must be extracted, translated, uploaded, and loaded regularly for the marketing system to be functional.

Once the data is in the cloud, the data is augmented for several purposes, including segmentation, lookalike matching, data flows, and personalized advertising. Segmentation arranges potential customers into groups based on attributes and activities in the input data. Lookalike matching is used to match the attributes and activities of new customers to similar customers who have been with the organization for a more extended period. Input data is utilized to design interactions with the new prospects to increase their engagement with the organization. The input data from the attribute or activities is also used to design personalized advertisements to motivate the new prospect to engage in that next activity.

Unfortunately, the input data may have been entered incorrectly in a self-service fashion, such as through a mobile application, web page, or kiosk. A tired or poorly trained customer service representative may have entered the data incorrectly. A third option for poor input data is the data may have come incorrectly from an external organization, such as a biographical data service provider that may suggest updates to addresses, phone numbers, or email addresses. A fourth option could be a malicious user intentionally polluting the data in revenge for some wrong they feel was done to them by the organization.

The best case is that an improper email is sent or the prospect displays and ignores an advertisement or offer. Worse case, an organization's reputation is damaged, and sales and relationships with customers are lost. It would be best to discover the vulnerabilities early in the design process or software development to minimize the risk.

Table 8 – AI/ML Activity STRIDE Model

Action	S	T	R	I	D	E
Segmentation						
LookALike						
Data Flow	X					
Personalized Advertisements						

Table **8** shows the four activities that utilize AI or ML in the cloud arranged in a STRIDE Model. We marked the columns for spoofing with the data flow activity, as we can envision a scenario when a user may want to qualify for a new customer offer by creating a fake account. The rest of the model is left blank as they are inappropriate threats to the activities. The small number of threats in our model could lead a software development team to believe the application is not at risk based on the limited representation in the STRIDE model.

XI. BIRFS THREAT MODELING

To better model the risks associated with applications that utilize AI or ML algorithms, we developed the BIRFS Model [14]. The BIRFS Model covers a process or system's five leading risk categories that utilize AI or ML algorithms.

B-potential biases in output

The B in BIRFS comes from the risk of potential biases in the AL or ML algorithm output. AI algorithms can exhibit biases in their production due to various reasons. These biases can arise from the data used to train the models, the design of the algorithms, or both. Here are some familiar sources of biases in AI algorithms:

- **Biased Training Data:** If the training data used to train the AI model is biased, the model will learn and perpetuate those biases. For example, if historical data used for training reflects societal biases, the model may replicate and amplify those biases.
- **Data Sampling Bias:** If the training data does not represent the entire population, the model may be biased toward the overrepresented groups. The sampling bias can lead to inaccurate predictions or decisions for underrepresented groups.
- **Labeling Bias:** Biases in labeling training data can also contribute to biased models. If the labels used for training data are biased, the model may learn and propagate those biases.
- **Algorithmic Bias:** The design and structure of the algorithm itself can introduce bias. The bias may happen if the algorithm uses feature proxies for sensitive attributes (e.g., using ZIP code as a proxy for race) or if the algorithm inherently reflects certain societal biases.
- **Implicit Bias in Training Examples:** Biases in the examples used to train the model can also contribute to biased outputs. For instance, if a model is trained on text from the internet, it may learn and reproduce the biases present in that text.
- **Lack of Diversity in Development Teams:** The composition of the teams developing AI models can also influence biases. If development teams lack diversity, there may be a lack of perspectives and awareness regarding potential biases in the models.
- **Feedback Loop Bias:** Biases can be reinforced through feedback loops. For example, biased predictions can lead to biased outcomes, which are then used as input for future predictions, creating a self-reinforcing cycle.
- **Contextual Bias:** The context in which the AI system is deployed may introduce bias. For instance, a model trained on data from a specific cultural or geographical context may not generalize well to other contexts.

Addressing biases in AI algorithms is a complex and ongoing challenge. It requires careful consideration at every stage of the AI development process, including data collection, model training, and deployment. Strategies such as diverse and representative data collection, regular audits of model outputs, and involving diverse teams in AI development can help somewhat mitigate biases.

I - input is outside the domain of control.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

The I in BIRFS comes from the risk when the input data comes from outside the organization's control. When it comes to AI models, some factors and inputs are outside the direct control of the developers or operators. These external influences can affect the performance and behavior of AI models in various ways. Here are some examples:

- **External Data Changes:** AI models are often trained on historical data, and if the real-world data changes over time, it can impact the model's performance. For example, sudden shifts in user behavior, market dynamics, or other external factors might lead to a mismatch between the training data and the current environment.
- **Changing User Expectations:** User expectations and preferences can evolve. AI models trained to meet specific user needs may become less effective if user expectations change, and these changes are beyond the direct control of the AI developers.
- **Regulatory Changes:** Changes in regulations or legal frameworks can significantly impact how AI models operate, especially in highly regulated industries. Developers may need to adapt models to comply with new laws or regulations.
- **External Security Threats:** AI systems can be vulnerable to external security threats. Malicious actors may attempt to manipulate or compromise AI models, leading to undesirable outcomes. These threats are often beyond the control of the AI developers and require ongoing security measures.
- **Environmental Factors:** The performance of AI models may be affected by environmental factors, such as changes in weather, network conditions, or other external variables. For example, a model designed for specific weather conditions might perform differently in a completely different climate.
- **Integration with External Systems:** AI models are often integrated into larger systems and workflows. Changes or issues in these external systems, which are not under the direct control of AI developers, can impact the overall performance of the AI model.
- **Global Events and Catastrophes:** Unforeseen global events, such as natural disasters, economic crises, or pandemics, can have widespread effects on various industries. AI models operating in affected domains may face challenges due to disruptions caused by such events.
- **User Feedback and Interactions:** User interactions and feedback can influence the behavior of AI models. Users providing biased or unrepresentative feedback may impact the model's learning and performance. User behavior is often outside the direct control of AI developers.

Addressing the challenges posed by external factors requires robust system design, ongoing monitoring, and adaptability. Developers should consider building models that adapt to environmental changes and implement continuous monitoring and update mechanisms.

R - output result does not deviate from a reasonable range

The R in BIRFS comes from the risk when the output is wrong but in a reasonable range for the input data. Ensuring that the output results of AI algorithms fall within a reasonable and expected range is crucial for the reliability and safety of AI systems. Deviations outside a reasonable range can pose risks and challenges. Here are some considerations related to the risk of output results deviating from a reasonable range in AI algorithms:

- **Out-of-Distribution Data:** If an AI model encounters data significantly differently from what it was trained on (out-of-distribution data), it may produce unpredictable and unreliable results. The new data ranges can happen if the model encounters scenarios or inputs not adequately represented in the training data.
- **Overfitting:** Overfitting occurs when a model learns the training data too well but fails to generalize to new, unseen data. In such cases, the model may perform well on the training set but poorly on real-world data, leading to outputs that deviate from a reasonable range.
- **Data Quality Issues:** Poor-quality or biased training data can contribute to the model learning incorrect patterns or making inaccurate assumptions. If the data used to train the model is not representative or contains errors, the model's outputs may deviate from what is considered reasonable.
- **Lack of Explainability:** If an AI model is too complex or lacks interpretability, it may be challenging to understand why it produces a particular output. This lack of transparency can make identifying and addressing deviations from a reasonable range challenging.
- **Concept Drift:** Over time, the underlying patterns in data may change, a phenomenon known as concept drift. If an AI model is not regularly updated to adapt to these changes, its outputs may become less accurate and fall outside the expected range.
- **Adversarial Attacks:** Adversarial attacks involve intentionally manipulating input data to deceive an AI model and produce incorrect outputs. If an AI system is vulnerable to such attacks, the outputs may deviate from the reasonable range, posing security and reliability risks.
- **Uncertainty and Confidence Estimation:** AI models should ideally provide measures of uncertainty and confidence in their predictions. If a model is overly confident in its outputs, even in situations where it should be uncertain, it may lead to outputs that deviate from a reasonable range.
- **Monitoring and Validation:** Monitoring and validating model outputs against real-world data are essential. If the model's performance degrades or deviates from expected behavior, it should trigger alerts for further investigation and potential retraining.

It's vital to employ good practices in data collection, preprocessing, model training, and ongoing monitoring to mitigate the risk of output results deviating from a reasonable range. Additionally, incorporating human oversight and feedback mechanisms can enhance the system's robustness and help identify and correct possible deviations. Regular updates and retraining of models based on new and relevant data are also crucial for maintaining performance in dynamic environments.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

F - forensics or logging to defend results

The F in BIRFS comes from the risk when the algorithm process cannot be audited to discover why it produced a particular output based on a set of input data. Forensics and logging play crucial roles in defending the results generated by AI algorithms. They provide a means to trace, understand, and validate the processes and decisions made by AI models. Here are critical aspects of forensics and logging in the context of AI:

- **Data Logging:** Record all relevant data in the AI model's training and inference processes. The logging includes input data, preprocessing steps, feature engineering, and any transformations applied to the data. Having a detailed log helps in understanding the context and ensuring transparency.
- **Model Configuration and Hyperparameters:** Document and log the configuration settings and hyperparameters used during the training of AI models. This information is crucial for reproducibility and understanding the model's setup.
- **Model Training Logs:** Record information about the training process, such as training loss, accuracy metrics, and other relevant statistics. The logs help track the model's performance during training and identify potential issues.
- **Algorithm Versions and Updates:** Keep track of the versions of algorithms and models used. When updates or changes are made to the algorithm, logging ensures you can trace which version was used for specific results.
- Timestamps and Versioning: Timestamps in logs can be critical for establishing a chronological order of events. Additionally, versioning of data, models, and algorithms helps associate specific results with the corresponding versions used.
- **Explainability and Interpretability Logs:** Record explanations and interpretations provided by the AI model, especially in cases where explainability is crucial. The logs can include feature importance, attention weights, or any other information that aids in understanding the model's decisions.
- **User Interactions and Feedback:** If the AI system involves user interactions, log relevant user inputs and system responses. This information is valuable for analyzing user experiences and addressing issues from the user's perspective.
- **Monitoring and Anomaly Detection Logs:** Implement monitoring logs to track the model's performance in real-time. Logging anomalies or unexpected behavior helps identify when the model's outputs deviate from expected ranges.
- **Security Logs:** Incorporate security logs to capture information about potential adversarial attacks or unauthorized access. Security logs can help identify and mitigate security risks.
- **Compliance and Regulations:** Ensure that logging practices align with relevant compliance requirements and regulations. Some industries and regions may have specific guidelines regarding data handling and logging.
- **Forensic Tools and Techniques:** Develop or utilize forensic tools and techniques to investigate issues or discrepancies in AI model results. These tools can help conduct detailed analyses of model behavior and decision-making processes.
- **Human-in-the-Loop Logging:** If human reviewers are involved in decision-making, log their interactions and decisions. This information can be valuable for understanding the role of human oversight in the system.

By incorporating comprehensive logging and forensic practices, developers and operators of AI systems can enhance transparency, accountability, and the ability to defend the results produced by AI algorithms. These practices are critical in applications where the stakes are high, such as healthcare, finance, and critical infrastructure.

S - Sensitive or private data needs to be protected

The S in BIRFS comes from the risk associated with protecting private or sensitive input data. Protecting sensitive or private data is a critical aspect of using AI algorithms, and there are several techniques and best practices to ensure data privacy and security. Here are key considerations:

- **Data Encryption:** Implement encryption mechanisms to protect data at rest and in transit. Encryption ensures the data remains unreadable even if unauthorized access occurs without the appropriate decryption keys.
- **Secure Data Storage:** You should store sensitive data in secure, access-controlled environments. Utilize secure databases and storage systems with proper access controls to restrict unauthorized access to sensitive information.
- **Data Masking and Anonymization:** Before using data in AI algorithms, consider techniques such as data masking and anonymization to replace or generalize sensitive information. Masking and anonymization reduce the risk of exposing private details introduced during model training.
- **Federated Learning:** Consider federated learning when data cannot be centralized. This approach allows models to be trained across decentralized devices or servers without exchanging raw data, preserving privacy.
- **Differential Privacy:** Implement differential privacy techniques to add noise or randomness to data, making it harder to link specific data points to individuals. Differential privacy protects individual privacy while still allowing meaningful analysis.
- **Access Controls and Authorization:** Implement strict access controls and authorization mechanisms to ensure that only authorized personnel can access sensitive data. Regularly review and audit access permissions.
- **Secure Model Deployment:** When deploying AI models, ensure the inference process is conducted securely. Limit access to the model and its outputs to authorized users and systems.
- **Secure APIs:** If AI models are accessed through APIs (Application Programming Interfaces), secure the APIs by implementing authentication and authorization mechanisms and encrypting data transmitted between the client and the API.
- **Regular Security Audits:** Conduct regular security audits to identify vulnerabilities in the system. The regular security

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

audits include the AI models and the infrastructure supporting them.

- **Compliance with Privacy Regulations:** Ensure your AI system complies with relevant privacy regulations such as GDPR, HIPAA, or other industry-specific standards. Understand the legal requirements for handling sensitive data and incorporate necessary safeguards.
- **Data Lifecycle Management:** Establish clear policies for the entire data lifecycle, including collection, storage, processing, and disposal. Regularly review and update these policies to align with evolving privacy and security standards.
- **Educate and Train Personnel:** Train personnel handling sensitive data on privacy best practices and security protocols. Human error is a common source of data breaches, so awareness and education are crucial.
- **Incident Response Plan:** Develop a comprehensive incident response plan to address potential data breaches. This plan should outline the steps during a security incident, including communication strategies and legal obligations.
- **Transparent Communication:** Communicate to users and stakeholders how their data will be used, processed, and protected. Transparent communication builds trust and helps users understand the measures to safeguard their privacy.

By incorporating these measures, organizations can significantly reduce the risks of handling sensitive or private data in the context of AI algorithms. Data privacy should be a fundamental consideration throughout the entire AI development and deployment lifecycle. Table **9** displays the same four AI/ML activities discussed earlier in the BIRFS model. As you can see from the model, many more risks and vulnerabilities are represented by the model than in the previous STRIDE model. The mitigations discussed, along with the BIRFS model elements can next be applied to reduce the risks.

Table 9 - AI/ML Activity BIRFS Model

Action	B	I	R	F	S
Segmentation	X	X	X	X	
LookALike	X	X	X	X	X
Data Flow	X		X	X	X
Personalized Advertisements	X		X	X	

XII. Conclusions and Future Works

In this work, we provide three domain-specific modeling methodologies to handle issues in cloud, database, and AI/ML software related to NFRs in distributed systems. We present modeling methodologies aimed at uncovering issues related to NFRs in the software development lifecycle. Our models enable us to identify significantly more fine-grained risks associated with the development of these systems than traditional threat modeling techniques. Additionally, we have enhanced the modeling of functional requirements by employing UML stereotypes to represent the NFRs identified in the models. Future work will incorporate code generation to mitigate the risks identified and modeled throughout this process.

References

[1] A. Olmsted, "PERTD - Cloud Application Threat Modeling," in *Proceedings of CLOUD COMPUTING 2025, The Sixteenth International Conference on Cloud Computing, GRIDs, and Virtualization*, Valencia, Spain, 2025.

[2] C. J. Pavlovski and J. Zou, "Non-functional requirements in business process modeling," *Proceedings of the Fifth on Asia-Pacific Conference on Conceptual Modelling, vol. 79,* pp. 1-10, 2008.

[3] M. Glinz, "Rethinking the Notion of Non-Functional Requirements," *Third World Congress for Software Quality, Munich, Germany,* pp. 1-10, 2005.

[4] Alexander, I, "Misuse Cases Help to Elicit Non-Functional Requirements," *Computing & Control Engineering Journal, 14, 40-45,* pp. 1-10, 2003.

[5] R. Ajith and A. Sheth, "Semantic Modeling for Cloud Computing, Part I," *Computing,* vol. May/June, pp. 81-83, 2010.

[6] Object Management Group, "Unified Modeling Language: Supersturcture," 05 02 2007. [Online]. Available: http://www.omg.org/spec/UML/2.1.1/. [Accessed 11 Nov 2025].

[7] A. Olmsted, Security-Driven Software Development: Learn to analyze and mitigate risks in your software projects, Birmingham, UK: Packt Publishing, 2024.

[8] Object Management Group, "OMG Formally Released Versions of OCL," 02 2014. [Online]. Available: http://www.omg.org/spec/OCL/. [Accessed 11 Nov 2025].

[9] JSON.org, "Introducing JSON," 2024. [Online]. Available: https://www.json.org/json-en.html. [Accessed 11 Nov 2025].

[10] Open Collective, "JSON Schema," 2024. [Online]. Available: https://json-schema.org/. [Accessed 11 Nov 2025].

[11] Zapier Inc., "Automate without limits," 2024. [Online]. Available: https://zapier.com/. [Accessed 11 Nov 2025].

[12] Microsoft, "Power Automate," 2024. [Online]. Available: https://www.microsoft.com/en-us/power-platform/products/power-automate. [Accessed 11 Nov 2025].

[13] R. Khan, D. Laverty, D. McLaughlin and S. Sezer, "STRIDE-based threat modeling for cyber-physical systems,," in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, Turin, Italy, 2017.

[14] A. Olmsted, "BIRFS is a Threat Model for Software Systems That Utilize Artificial Intelligence or Machine Learning Algorithms," *International Conference on Artificial Intelligence and its Application,* pp. 23-37, 2024.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

From Principles to Practice: An End-to-End Approach for Trustworthy ML in Critical Systems

Afef Awadid, Lucas Mattioli, Karla Quintero
IRT SystemX, France
email: {afef.awadid, lucas.mattioli, karla.quintero}@irt-systemx.fr

Juliette Mattioli
Thales, France
email: juliette.mattioli@thalesgroup.com

***Abstract*—This work presents one of the products of the Confiance.ai research program which addresses an end-to-end method for engineering trustworthy ML-based systems [1]. The proposed methodology revisits software and systems engineering as it encompasses all development phases of the system while integrating the specificities related to the development of ML-based components within the system. The method leverages vastly researched and deployed standard procedures from design to validation and maintenance in order to provide rigor, structure, and traceability when developing ML-models.**

Keywords- trustworthy AI; trustworthiness attributes; trustworthiness risk analysis; trustworthiness assessment; safety-critical ML-based systems; end-to-end engineering methodology; trustworthy AI engineering.

I. Introduction

The term "AI" (Artificial Intelligence) was first used in a workshop held at Dartmouth College in 1956. It was introduced as a branch of computer science that tries to mimic human thinking by using symbols and knowledge bases that are also symbol-based. Any technology, even AI, is developed to provide a service fulfilling some needs. The AI discipline aims to embed cognitive capacities such as perception, learning, reasoning, planning, decision and dialogue, to an artificial system. In February 2025, the European Commission defines an AI system (see Figure 3) as the following: *a machine-based system that is designed to operate with varying levels of autonomy that may exhibit adaptiveness after deployment and, for explicit or implicit objectives, infers from the input it receives how to generate outputs, such as predictions, content, recommendations or decisions that can influence physical or virtual environments*.

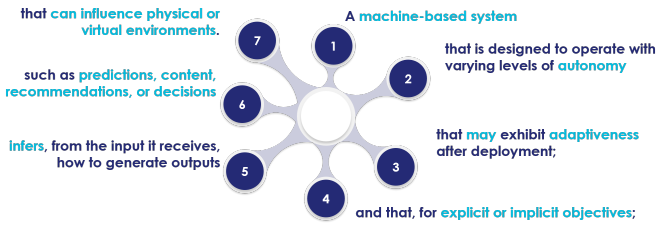


Figure 1. The European Commission AI-system definition comprises seven main elements (Feb. 2025)

In our context, an AI-based system is defined as a system that incorporates AI components. AI-based critical systems, which can have severe consequences in case of failure, are considered to be "high risk" under the European regulation known as the "EU AI Act" [2]. These systems can, for example, represent safety components of regulated products which are required to undergo a third-party conformity assessment. Examples of such systems can be found in the fields of transportation, healthcare, defense, and security in general. The deployment of such systems is contingent upon their demonstrated capacity to deliver the anticipated service in a secure manner, while meeting user expectations with regard to quality and continuity of service. Furthermore, users might consider negative any surprising or unexpected actions from the system.

In order to characterize such systems with a view to quality assurance, [3] proposed considering several dimensions, including the artifact type dimension, the process dimension and the trustworthiness characteristics, which are relevant to software product or system quality. Furthermore, the focus of the series of standards SQuaRE (Systems and Software Quality Requirements and Evaluation) is on software quality [0]. In addition, the specific nature of AI is addressed in order to provide a quality model for AI systems. Consequently, the design of critical AI systems requires the demonstration of their trustworthiness, as asserted by [4].

Trustworthy AI is based on these three components [5]: it must comply with all applicable laws, adhere to ethical principles, and be robust. This shift is driving the new discipline of AI engineering [6] to support the industrial design of such systems. Therefore, the development of AI-enabled systems is heavily dependent on the application of specific traditional software and system engineering practices. For example, engineering teams must conceptualize AI systems that can handle the inherent uncertainty of their components, data, models, and outputs — particularly when implementing data-driven AI. The user experience with AI systems is dynamic [7]. Interfaces must clearly show what the system is doing, how outputs are generated (dataset quality), and when the system is not behaving as expected (monitoring throughout the lifecycle). Therefore, engineering teams must account for the different rhythms of change, including changes in data, models, systems, and business processes.

This discipline aims to ensure that critical AI-based systems in safety-, mission- and business-related domains are valid, explainable, resilient, safe, secure, and compliant with regulations, standards, and responsible practices (ethics, sustainability, etc.). When dealing with critical systems, additional

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

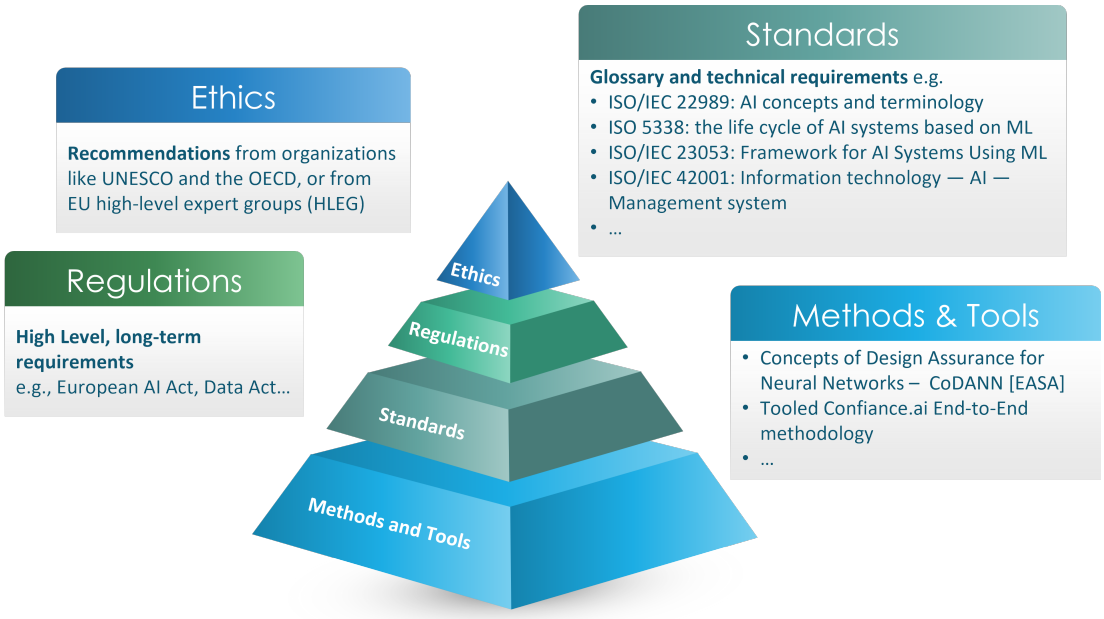


Figure 2. From ethics to the end-to-end methodology through regulation and standards

constraints must be considered. In the context of system design, processes must be optimized, justified, replicated where possible, and improved. However, it is also essential to ensure that the system meets the appropriate level of trustworthiness [8] [9]. This includes robustness (the ability of a system to withstand errors during execution and cope with erroneous input), cybersecurity, and dependability (including reliability, availability, maintainability, and safety properties), among others.

Thus, in the following, we will first remind the current context of AI regulation and standardization of the definition of "trustworthiness" as "*the ability to meet stakeholders' expectations in a verifiable way*". To determine whether a given risk is as such, it should be first identified. Then it should be analyzed with respect to the "intended purpose" which is defined as "*the use for which an AI system is intended by the provider". It also includes "the specific context and conditions of use*".

It is imperative that a well-established engineering discipline oversees AI capabilities. "**AI Engineering**" is an emerging discipline that focuses on applying AI in real-world contexts. AI engineering involves applying engineering principles and methodologies to create scalable, efficient, trustworthy and responsible AI-based solutions. It merges aspects of data engineering, knowledge engineering, algorithm engineering, software engineering, system engineering, cyber-security, safety and ethical engineering and also cognitive engineering including human factors to accelerate the development and the deployment of AI-based capabilities. It also speeds up the maturation of individual tools. This is particularly evident in high-stakes scenarios such as responding to national security threats and military operations. Therefore, to maximize the potential of AI in such situations, we must address the unique challenges that AI systems encounter. While the capability to develop AI systems has increased due to greater computing power and more extensive datasets, these systems often only function in controlled environments and are difficult to replicate, verify and validate in real-world scenarios. AI Engineering aims to provide a framework and tools to proactively design AI systems to function in environments characterized by high degrees of complexity, ambiguity, and dynamics.

Then, we present an end-to-end methodology to support "trustworthy AI engineering", which encompasses the entire lifecycle of AI-based systems, from operational design domain (ODD) specification to maintenance [1]. This holistic methodology covers the design, development and deployment of AI systems in critical environments, including data engineering, algorithm design and development, deployment and monitoring. By integrating the principles outlined in international and national initiatives with our advanced internal engineering practices, this lifecycle ensures that AI systems perform their intended functions with the desired level of performance. It also makes AI-powered solutions transparent, responsible, and ethical. This systematic approach involves organizing multidisciplinary and fragmented approaches to trusted AI and applying a continuous workflow. Measures to improve AI trustworthiness must be implemented at every stage, including data sanitization, robust algorithms, anomaly monitoring, and risk auditing.

II. Regulation and Standardization

To ensure safety, reliability, availability, and maintainability, AI systems must perform, and continue to perform, as intended under sufficient conditions. Hazard analysis and risk assessment must be tailored to the unique characteristics of AI systems. This includes identifying potential critical errors in

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

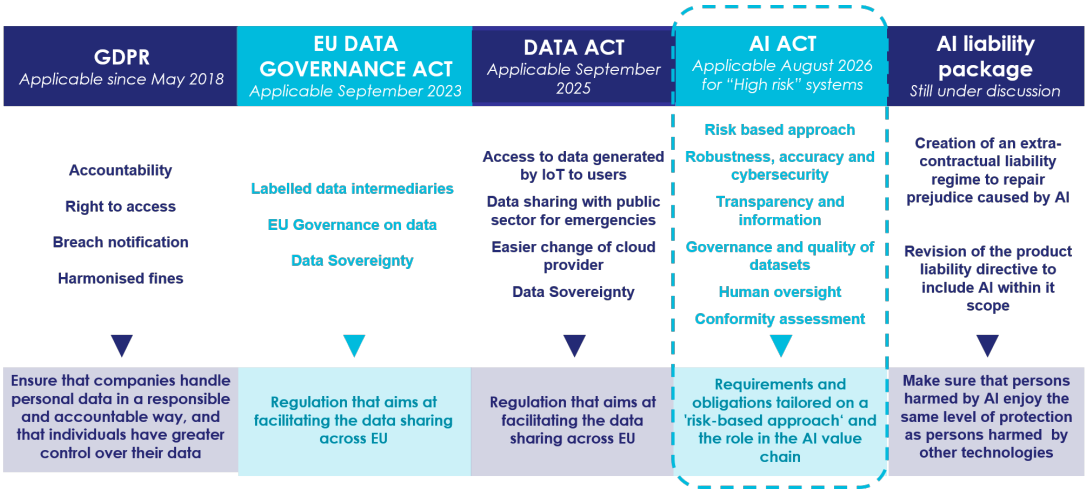


Figure 3. European Data and AI Regulation

the training data or knowledge representation and assessing the AI model's ability to generalize to unseen operational data. Performance requirements for AI algorithms are often driven by safety objectives, which limit the worst credible approximation error to an acceptable threshold.

However, trustworthiness is closely linked to accountability, which can be considered a measure of trust or an alternative to it. In [10], dependability is used to represent a system's overall quality based on four sub-attributes: security, safety, reliability, and maintainability. Subsequently, security and dependability became key attributes of trust in computer-based systems [11].

Another requirement relates to the quality of datasets used to train, validate and test models for high-risk AI systems. This considers a non-exhaustive list of issues, such as data collection processes, data engineering activities (*e.g.*, annotation, labeling, cleaning, enrichment and aggregation), data quality assessment and identification of possible data gaps or shortcomings and how these can be addressed. Last but not least is the mitigation of possible biases likely to affect the health and safety of individuals or lead to discrimination.

In 2019, the U.S. National Artificial Intelligence Research and Development Strategic Plan [12] stressed the importance of standard metrics for quantifying AI technologies. "Standard metrics are required to define quantifiable measures in order to characterize AI technologies". As a matter of fact, [13] have recently stated that "a great deal of effort is required to determine which suitable measurements should be utilized to evaluate system performance across characteristics for responsible AI and across profiles for specific applications/contexts". Governments are responding with regulations typically associated with human rights. In 2024, the European Union adopted the AI Act (see Figure 3). These regulations set out long-term, high-level requirements, sometimes based on recommendations from organizations such as UNESCO [14] and the OECD [15] [16], or from High-Level Expert Groups (HLEG) [5].

These high-level requirements need to be operationalized for companies and developers. As shown in Figure 3, standards and regulatory frameworks define more detailed requirements, but they focus on what to do rather than how to do it. This leaves the choice of tooled, end-to-end methodology for developing AIs that fulfill these requirements to companies and developers.

The Assessment List for Trustworthy AI considers 7 pillars of trustworthiness: 1) human agency and autonomy, 2) technical robustness and safety, 3) privacy and data governance, 4) transparency, 5) diversity, non discrimination and fairness, 6) societal and environmental well-being, 7) accountability. This List is one of the basis of the AI Act [2] which requires companies to take measures to ensure that their products developed or deployed in the European Union are safe and comply with ethical principles.

In the aeronautic domain, EASA [17] proposes a model of trustworthiness based on the characterization of the machine learning (ML) application (high-level function/task, concept of operations, functional analysis, classification of the ML application), safety assessment, information security management, and ethics-based assessment (which includes the 7 pillars of the ALTAI [18]). The Fraunhofer [19] offered an analysis of the standard [20] on management system for AI, stating compliance to the standard can contribute to ensuring AI trustworthiness since it encompasses the pillars of the ALTAI, provided that a third-party verification has been performed and along with an adapted quality management system.

In the same period, the characteristics of trustworthy AI system specified by the NIST include: "*valid and reliable, safe, secure and resilient, accountable and transparent, explainable and interpretable, privacy-enhanced and fair with harmful bias managed*". Then the NIST produced an analysis of the components of trust [21] and highlighted several top level aspects for the design of a trustworthiness model, that should encompass the user experience, the perceived technical trustworthiness, the pertinence of each trustworthiness characteristic in the user's specific context of use...

Standards provide a framework for legislation and rules by recording the current state of the art and recommended

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

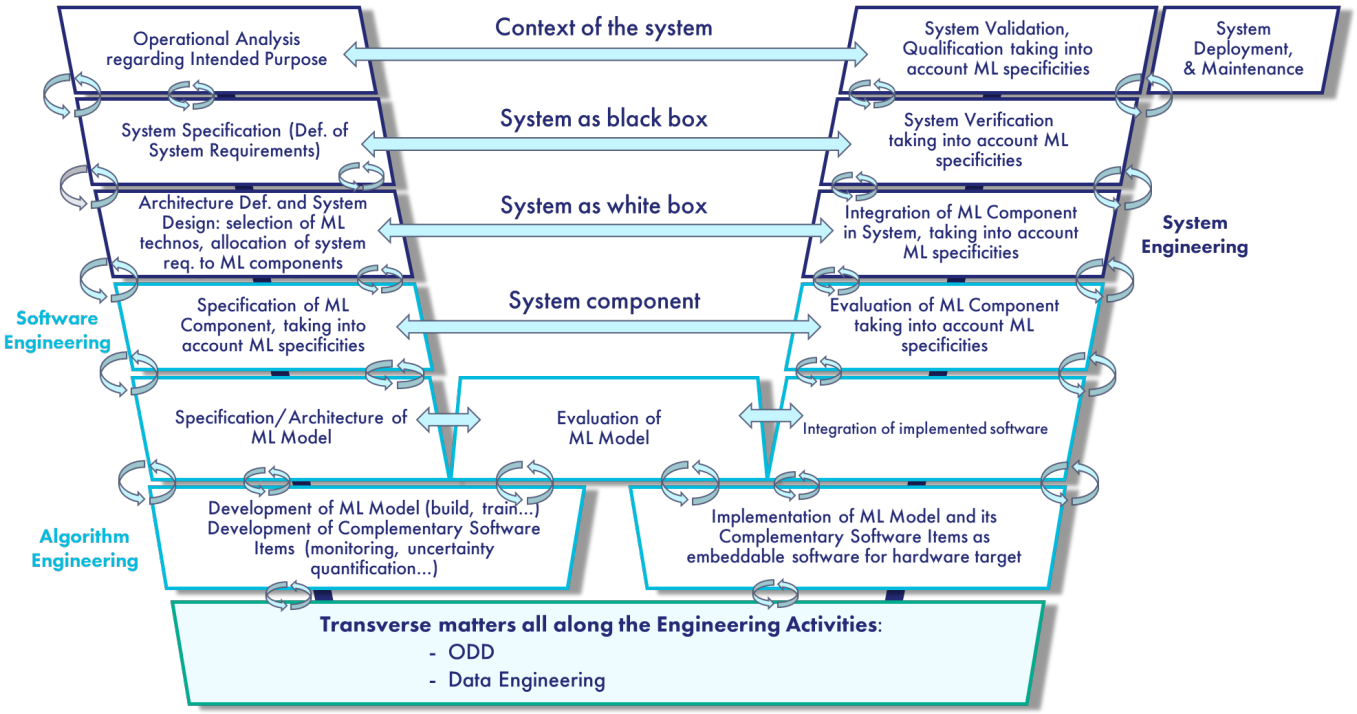


Figure 4. High-level view of the end-to-end methodology (source: https://bok.Confiance.ai).

practices, and offering a foundation for showing adherence and accreditation. Several organizations and initiatives, such as ISO/IEC, IEEE and NIST, are currently working on developing relevant AI standards. The standards developed by ISO/IEC [20] cover a wide range of AI aspects. These include terminology, performance metrics, data quality, ethics, and human-AI interaction. The ethical implications of AI technologies are the focus of the IEEE P7000 series of standards [22]. The NIST framework [21], meanwhile, provides guidance on managing risks, ensuring data quality, and promoting transparency and accountability in AI systems.

In 2019, ETSI set up an Industry Specification Group on Securing AI (ISG SAI) [23] to provide existing and potential mitigation against threats for AI-based systems. Robust security measures must protect AI systems from cyber-attacks, data breaches and unauthorised manipulation. These should include advanced threat detection and mitigation strategies and resilience mechanisms to operate securely in hostile environments. Cybersecurity should be embedded in the system and data pipelines. The lines between security and safety are not always clear when it comes to AI. Incorrect outputs can be caused by malicious actions or natural events.

Ethical engineering focuses on the need for fairness, transparency, and accountability in AI. This involves ensuring that algorithms are unbiased, produce explainable results, and adhere to societal and legal values. This engineering requires ongoing review by engineers, ethicists and domain experts.

However, it is crucial to recognize that the transfer of AI technology, particularly ML, must adhere to specific standards and processes to successfully transform research outcomes into industrial products fit for purpose that meet customer needs. For example, since data collection and analysis are crucial for developing any ML-based system, prioritizing data quality is essential. This requires adherence to compliance regulations, such as those relating to data privacy. Concurrently, operational requirements encompassing maintenance must be addressed. It is therefore evident that developing and implementing AI/ML systems involves both technical and business aspects, from problem conception to customer delivery. The development and operation of critical AI systems therefore requires an end-to-end, tool-based AI engineering methodology, which will be outlined subsequently.

III. The End-to-End Methodology

The version of the methodology presented herein has been produced as a result of the work within the French program Confiance.ai [1] [25] [26] [27] which was a pillar of the Grand Défi "AI for industry" initiative, which is pioneering methodologies for the development of trustworthy AI systems across sectors. Its associated roadmap is nourished by industrial needs and the evolution of the state-of-the-art [28]. Namely, several industrial projects and research initiatives have derived from Confiance.ai, generating the emergence of an ecosystem for the engineering of trustworthy AI for critical systems. In addition, the **European Trustworthy AI Association** (https://www.trustworthy-ai-association.eu/) is built on an open-source, community-driven approach, serving as a key enabler, giving stakeholders access to a dynamic ecosystem where they can learn from peers and co-develop tools. These tools are designed to ensure the adoption of

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

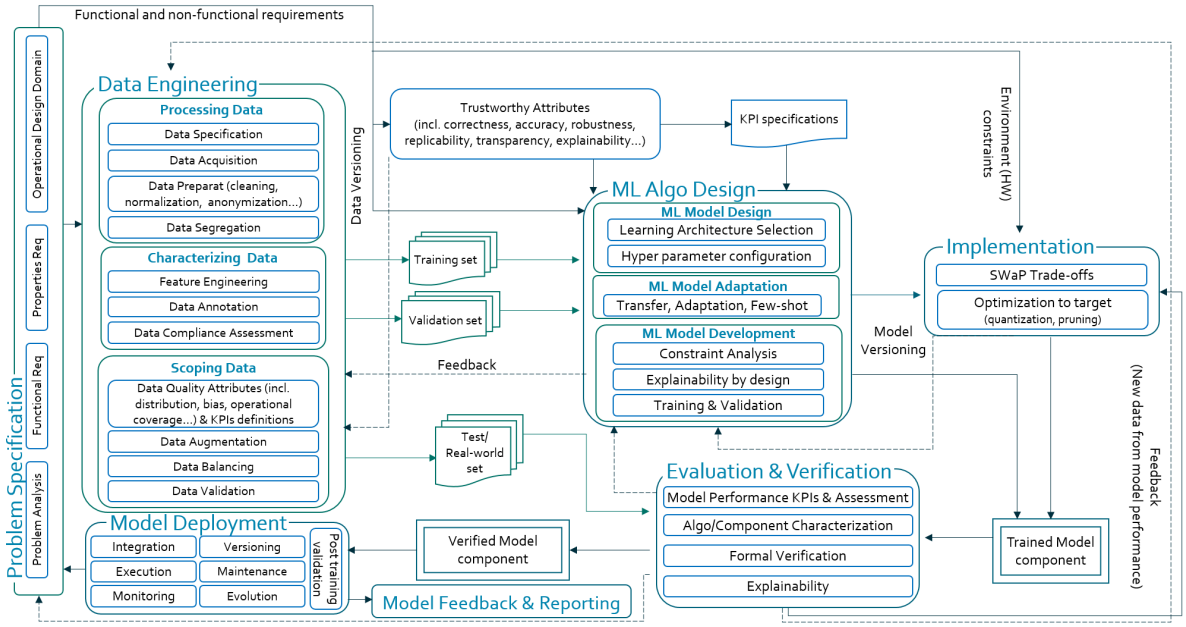


Figure 5. ML algorithm engineering pipeline [24]

scalable, secure and trustworthy AI based on this end-to-end methodology, which supports the engineering of trustworthy AI-based systems. The proposed methodology addresses following issues [29]:

- How can AI/ML models be designed to satisfy trustworthy attributes (explainability, robustness, accuracy, etc.)?
- How can these models allow a clear understanding of their behavior in the operational domain?
- How can AI/ML models be implemented and embedded on hardware, by making them fit to the target without discarding their trustworthy properties?
- Which data engineering methods should be applied to manage large volumes of data and account for the evolving operational domain?
- What kinds of verification, validation, and certification processes should be considered when dealing with AI/ML-based systems?

By addressing these challenges, the end-to-end methodology aims to answer the research question: How to ensure the reliability and trustworthiness of AI-based safety-critical systems? It is based on the premise that the development of ML-based critical systems should be structured with a trustworthiness imperative from the design phase, thus providing precise requirements for integration, verification, and validation, as well as for proper deployment and maintenance [30] [31]. It is a multi-domain collaboration that leverages concepts and procedures coming from different fields into the agnostic proposal of engineering trustworthy AI/ML-based critical systems. The result is the formalization, through a common language, of the structure and workflow for all actors involved in the process of designing trustworthy AI-based critical systems, *i.e.,* data engineers, systems engineers, safety engineers, software engineers, among many others.

A. ML Algorithm Engineering

The engineering ML-based systems is often portrayed as involving the creation of an ML model and its deployment. But in practice, the ML model is only a small part of the whole system. Much more is needed to ensure that an ML model is trustworthy and its behavior is predictable. This includes things like designing data pipelines, monitoring and logging, and so on. We defined the ML algorithm engineering pipeline to capture these aspects of AI engineering (see Figure 5). This pipeline differentiates between three types of development: requirements-driven, outcome-driven and AI-driven [32]. The starting point is that data must be available for training. Data engineering provides the foundation for various data collection and qualification methods, which can then be divided into training, testing, and cross-validation sets.

The following steps are encapsulated as sub-tasks within the pipeline:

1) **Problem specification**: Inclusion of the operational design domain (ODD), which is the description of the specific operating condition(s) in which a safety-critical function or system is designed to operate properly, including but not limited to environmental conditions and other domain constraints [33]. These requirements and architecture are the result of subsystem design activities and are part of specification activities. These requirements describe the specific function that the ML items should implement. They also describe the safety, performance and other requirements that the machine learning items should achieve.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

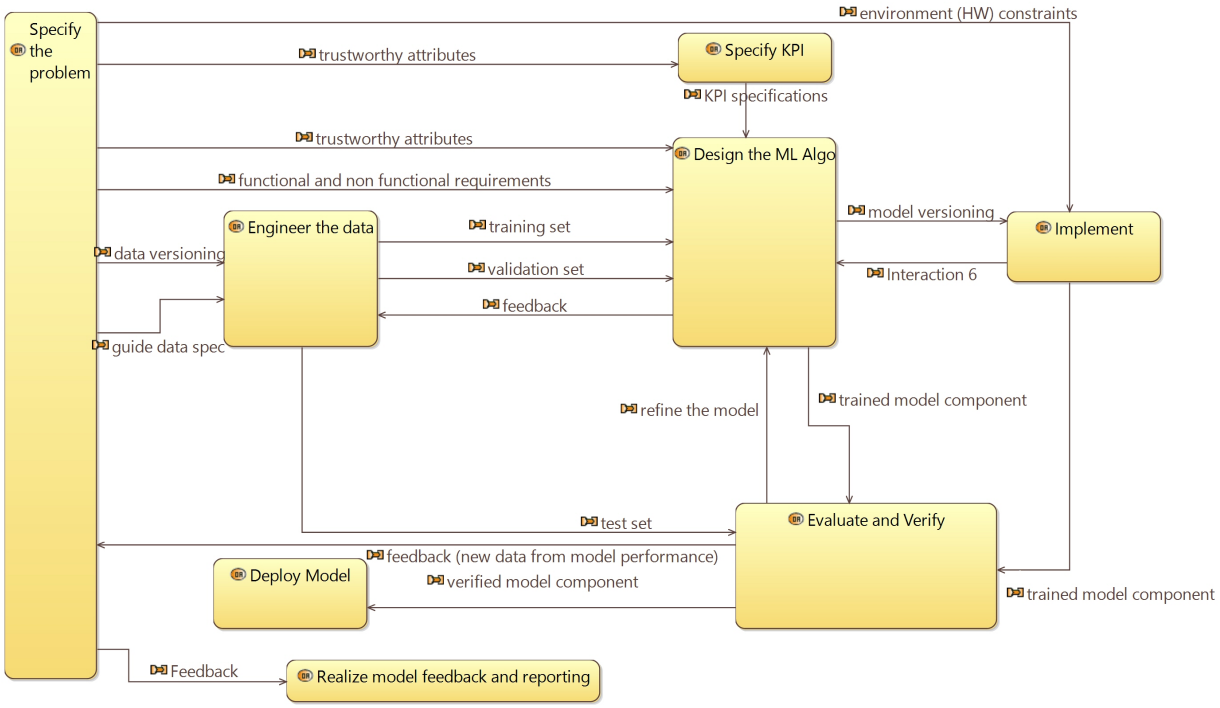


Figure 6. ML algorithm engineering process

2) **Data engineering**: Large amounts of data are needed to train a machine learning model so that it can learn to carry out its function. Before it can be used, data needs to be collected. It also needs to be prepared. The process of collecting data from a number of sources is known as data aggregation. The collected data needs to be substantial, accessible, comprehensible, reliable and usable. The process of transforming raw data into usable information is known as data preparation, or data preprocessing.
3) **ML Algorithm Design**: The ML algorithm can be trained using a feeding set, allowing it to learn appropriate parameters and features. The model will be refined once the training is complete, using the validation dataset. This process may involve modifying or discarding variables, as well as tweaking model-specific settings (hyperparameters) until an acceptable level of accuracy is achieved.
4) **Implementation**: For an ML component to be developed, the targeted hardware platform, the IDE (Integrated Development Environment) and the language for development must be decided on. There are a number of options to choose from. The majority of these would undoubtedly satisfy our requirements, as all of them provide the implementation of AI algorithms that have been discussed to date. However, it is sometimes necessary to take into account embedded constraints.
5) **Evaluation and verification**: Once we have found an acceptable set of hyperparameters and optimized the model's accuracy, we can test it. The testing process employs our test dataset and is intended to verify that our models are utilizing accurate features. We may decide to retrain the model based on the feedback we receive. This could lead to improvements in accuracy, adjustments to output settings, or the deployment of the model as required.
6) **Model Deployment**

The next step is to design or select an AI algorithm from an existing library (*e.g.*, Scikit Learn [34]) to create a model. The model is then trained iteratively so that the result closely aligns with the "correct answers" from the ground truth. The model is then ready to be deployed, like any other component, once validation has been successfully completed, provided that the specified criteria have been met.

As algorithm engineering workflow, ML pipelines consist of several steps to train a model (see Figure 6). Such pipelines are iterative as every step is repeated to continuously improve the accuracy of the model and achieve a successful algorithm. Pipelines are not one-way flows. They are cyclic in nature and enables iteration to improve the scores of the machine learning algorithms and make the model scalable.

The method addresses as a whole both the system engineering layer and the ML algorithm engineering layer. The system layer accounts for all underlying phases that should design and specify to further along verify and validate the overall system's objective and performance as carried out in classic systems engineering. The ML layer then covers all phases related to the ML component that inherit system requirements to then refined requirements specific to the ML-components to be developed. This process aims to ensure the compliance of

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

the AI/ML components with the overall system requirements and intended purpose.

B. Data Engineering

The quality of the data-set depends on the processes and technologies that ensure data values are consistent with the ODD. These values are then assessed for quality using various methods and key performance indicators (KPI) [35]. In certain instances, when a natural language processing application contains spelling errors, it is possible to ascertain the caliber and dimensions, such as accuracy [36]. However, it is more difficult to detect admissible but incorrect values. As discussed in [37], most ML research focuses on improving model performance rather than datasets. Typically, classical ML practices involve using existing datasets and enhancing the complexity of techniques to address performance challenges. Conversely, data-driven AI adopts a more comprehensive strategy, placing significant emphasis on the data itself [38] [39]. Instead of just looking for patterns and relationships in the features that are given, data-driven AI involves collecting, processing and analyzing lots of data to create models that are more accurate and robust. Furthermore, it is a real challenge in the present day to link datasets together. These should be linked to the ODD. This should be done at the operational level of the system definition.

As highlighted in [40], dataset quality may have a bigger impact on performance than model design. Poor data quality is a major risk in data-driven AI, as it can cause issues at every data engineering step, like collection, annotation and feature engineering, and can lead to problems being missed. To overcome these challenges, the Confiance.ai research program proposes a methodological process for assessing data trustworthiness.

Data trustworthiness evaluation indicates the degree to which data and data items satisfy expectations. An overview of the main metrics used for the data quality assessment is summarized in Figure 7. This evaluation can be carried out at various stages of the process, typically during data development (*e.g.*, raw data or dataset preparation), but also during IVVQ and deployment (*e.g.*, to detect data drift).

C. ML Algorithm Design

In this phase, a modeling technique is chosen and applied, and its parameters are set, then an ensemble model is developed and tested. The variant and structure type are both determined here, as is the algorithm. This process is referred to as "training", wherein data and outcomes are employed to optimize the configuration of the model. This process constitutes the "learning" aspect of ML.

Various ML techniques are available. These include multiple types of classification models. These models identify the category that the input belongs to. There are also regression models. These predict a continuous-valued attribute for supervised tasks. Then there are clustering models. These group similar items into sets for unsupervised tasks. Finally, there are reinforcement learning models. These provide an optimal set of actions.

A common question is "*Which ML architecture should I use?*". The following table [41] is provided by the DEEL project (https://www.irt-saintexupery.com/deel/), which summarizes the most common ML techniques and their main applications. Each ML technique relies on one or more hypothesis function spaces and one or more exploration algorithms (not listed in this document) to minimize a loss function on the training dataset.

Techniques	Applications
Linear models: Linear and logistic regressions, SVM	Classification, Regression
Neighborhood models: KNN, K-means, Kernel density	Classification, Regression, Clustering, Density estimation
Trees: decision trees, regression trees	Classification, Regression
Graphical models: Bayesian network, Conditional Random Fields	Classification, Density estimation
Combination of models: Random Forest, Adaboost, XGboost	Classification, Regression, Clustering, Density estimation
Connexionist and statistical models: Neural networks, Deep learning...	Classification, Regression

After choosing the model, among the various algorithms present, one needs to tune the hyper parameters of each model to achieve the desired performance.

- Select the right algorithm based on the learning objective and data requirements.
- ConFigure and tune hyperparameters for optimal performance and determine a method of iteration to attain the best hyperparameters.
- Identify the features that provide the best results.
- Determine whether model explainability or interpretability is required.
- Develop ensemble models for improved performance.
- Test different model versions for performance.
- Identify requirements for the overall lifecycle.

The resulting model can then be evaluated to determine whether it meets the business and operational requirements.

D. The ML-based system lifecycle

Developing ML-based systems can be visualized as a "W-shaped" life-cycle (see Figure 4). This W-shape can be split into two parts. For AI systems, "intended goal"/"intended purpose" and "intended domain of use" are very high-level requirements that have to be translated into "engineering terms". The engineered "intended domain of use" is called Operational Design Domain (ODD). The ODD is the operational conditions for which an AI system is specified, designed, verified, assessed, operated, and disposed. ML engineering life-cycle begins with defining AI/ML algorithm requirements refined from system specification. This ML specification step includes the characterization of the ODD.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

	Approach / method	Machine Learning use	Data
Diversity	DPP [42]	Training and test sets	Image (object localization) & text (document summarization)
	R-DPP [43]	proposition of new metric	No data implemented
	D-MCL [44]	Training & test sets	Image (classification)
	PDS [45]	Training & test sets	Image (MNIST [46]), audio & 2-D synthetic data
Completeness	MADI [47] [48]	Not used for ML	Numerical & categorical data (from hospital for [48])
	POVM [49]	Test set (evaluation by using deep learning approaches (ICCnet & FidNet))	Tomography images
	MCAR, MAR, NMAR [50]	Training & test sets	Numerical data (diabetic data)
Representativeness	R-indicator [51] [52]	Not used for ML	Internet Data sources for [52]
	CI [53]	Test set	Tabular data
	Log Disparity [54]	Sampling, training and test sets	Clinical trials (classification)
Coverage	SelectiveNet [55] [56]	Training and test sets	Image (MNIST, Cifar [57] & ImageNet)
	Neuron Coverage [58]	Test set	Image (MNIST, ImageNet, Driving datasets) & numerical/categorical (Contagio, VirusTotal & Drebin datasets)
	DeepTest based on neuron coverage [59]	Test set	Image (real driving camera & synthetic)
	TensorFuzz [60]	Test set	Image (MNIST)
	TDA-AI2 [61]	Test set (applied on DRL)	3-D cloud data
Corner cases	[62], [63], [64],[65],[66],[67]	Anomaly detection	Images or videos
	[68]	Anomaly detection	Images (optical, radar & lidar)
	[69]	Anomaly detection	Numerical data (trajectories)
	[70],[59]	model evaluation	Images
	[58]	model evaluation	Images, PDFs, Android apps

Figure 7. A brief overview of the approaches and metrics used for data quality evaluation

This engineering activity is a critical step that changes the way AI researchers and engineers work. It involves a detailed description of all possible operating conditions, called the operating environment of the system, to enable data collection and knowledge representation. The reliability of the AI-based system depends on the correctness and completeness of this description, particularly for rare events or combinations of conditions that could be unsafe. The validity of a system is established by its intended use [71]. The ODD description is developed using a combination of top-down and bottom-up approaches. ODD aligns data and functional intent, *i.e.*, the data used for training and the resulting ML model(s) with their intended use, covering a wide range of conditions.

Data engineering is key. It involves the identification, collection, preprocessing and extraction of features from large datasets. These datasets are essential for designing and verifying ML models. This phase often involves advanced techniques. These techniques improve the representativeness, completeness, and relevance of the dataset (minimizing the simulation-to-reality gap). Rigorous quality controls, guided by Data Quality Requirements (DQRs), ensure data inputs are accurate and consistent. During model design, engineers select appropriate learning algorithms and improve model architectures through training and evaluation cycles. Optimization strategies balance computational efficiency and performance.

The second "V" of the "W-shaped" life-cycle includes the implementation engineering processes performed on the target platform (*e.g.*, specific hardware embedded in a ground or aerial vehicle). Validation and verification activities are driven by key trustworthiness properties, specified in low-level ML requirements. Validation activities ensure the correctness and completeness of ML requirements by verifying, analyzing, and tracing them back to higher-level requirements. Verification activities include extensively simulating, testing edge/corner robustness, scenario-based testing, analyzing the ML model explainability, and ODD coverage analysis [72]. The first level of verification ends with a selected AI model, which meets all its requirements in the development (learning) environment and serves as a design specification, ready for implementation into software and/or complex electronic hardware elements in the second level of verification. Figure 8 shows a high-level view of the verification phase of an automated feature based on ML and the interaction with the specification and validation phases.

MLOps, or Machine Learning Operations, and AI Engineering, while closely related, serve distinct roles within the ML lifecycle. MLOps focuses on the operationalization of machine learning models, ensuring that they are deployed efficiently and maintained effectively in production environments. In contrast, ML Engineering is primarily concerned with the

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

development and maintenance of an ML-based system. Thus MLOps emphasizes the operational aspects of machine learning, while AI Engineering is centered on the overall lifecycle of the system covering all system engineering concerns (from specification to maintenance) which includes MLOps for ML-based systems. MLOps involves collaboration between data scientists, ML engineers, and IT operations teams when AI Engineering involves system ad software engineers, data scientists, safety and cyber-security engineers. The end-to-end methodology (see Figure 4) supports all AI engineering activities where MLOps covers ML algorithm engineering and data engineering.

E. Deployment of a ML-component in the system

Deploying an ML component involves integrating it into an existing system, validating it and making it accessible for real-time or batch processing. The challenge for AI/ML-based systems lies in integrating, deploying and scaling a solution. The end-to-end methodology validates the quality of the data and knowledge, the process, and the added value delivered by AI/ML components, at a lower cost than the classical software/system engineering effort involved in automating and integrating a non-validated application with in-house and third-party systems. Combining the different engineering steps can require significant development effort, creating cost barriers when testing and validating ideas and prototypes that depend on integration with the rest of the system. Validating ML-based systems is more complex than manually coded systems. This is due to the behavior of ML-based systems, which depends heavily on data and knowledge, and for which models cannot be strongly specified a priori. Therefore, training data or knowledge-based models require qualification, similar to code [73].

Thus, verification testing of the ML-based system must also rely on the integration tests already performed at the "Integration of ML Component in System" step. They are usually run in a simulated environment (*i.e.*, with test benches or synthetic input data, or pre-recorded operational data). Consequently, testing the ML-based system consists again at least in regression testing (*i.e.*, test of no functional or non-functional regression), enabling to verify that the ML component features previously tested, integrated in the final ML-based system, still operate in conformance with their requirements based for example on

- Test by sampling and perturbation (empirical testing),
- Testing by formal verification of robustness (formal testing)...

However, it is also necessary to develop and run new verification tests, with potentially new tools and new test datasets, in order to ensure a complete requirements test coverage, prior to delivery to the Validation level.

F. Validation and Verification

Once the ML-based System has been successfully verified, it can be provided to the next engineering phase: the validation of the ML-based System. The difference between “verification” and “validation” is the following:

- Verification: evaluation against the system requirements that were written in order to design the ML-based System. Have we built the system right (as per design intent)?
- Validation: evaluation against the high-level needs identified during operational analysis. Have we built the right system (as per initial need)?

Then the "Validation, Qualification of ML-based System" engineering activity has to be performed:

- each time the “Operational Analysis” activity releases a new version of the operational/stakeholder needs (especially Intended Purpose Summary) against which the ML-based System shall be validated;
- and each time the “System Verification” activity releases a new version of verified ML-based System that is ready for validation.

Particular attention should be paid to stakeholders/operational requirements relating to the automation objectives that the system’s feature implemented by ML is intended to satisfy. Furthermore, ML-based system tests must be run in the target operational environment. It is no longer possible to simulate the environment using a test bench, synthetic input data or pre-recorded operational data.

The performance of a safety-critical system in its intended operational environment is a mandatory part of overall system validation. The process of traditional software validation involves establishing a chain of evidence that connects requirements to system-level tests. However, the use of machine learning techniques frustrates this approach due to the use of training data rather than a traditional design process. It is essential that software validation is based on tests that demonstrate a performance level commensurate with the criticality of the risks. These tests should be performed on a dataset that is fully representative of the factors that influence the model. The way in which the model’s functional characteristics and operational environment are specified may result in numerous factors influencing performance. To demonstrate the model’s effectiveness would require extensive testing datasets, potentially numbering in the millions of samples. Achieving this goal is an interesting prospect, but it is still at an early stage of research. Formal verification and simulation are interesting tracks to pursue. Therefore, verification requires ensuring that training and testing data cover all relevant operational conditions. In practice, this problem is generally made tractable by constraining the operational environment to a subset of all possible situations that could be dealt with by a human operator. The adoption of an ODD is the term given to that approach to limiting the operational needs of the system. Testing an ML component aims to detect discrepancies between the actual and intended behaviors of ML models. The term "ML testing" is used to describe any activity that is designed to reveal any bugs in ML items. "ML bugs" refer to any imperfection in an ML item that causes a discrepancy between the output.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

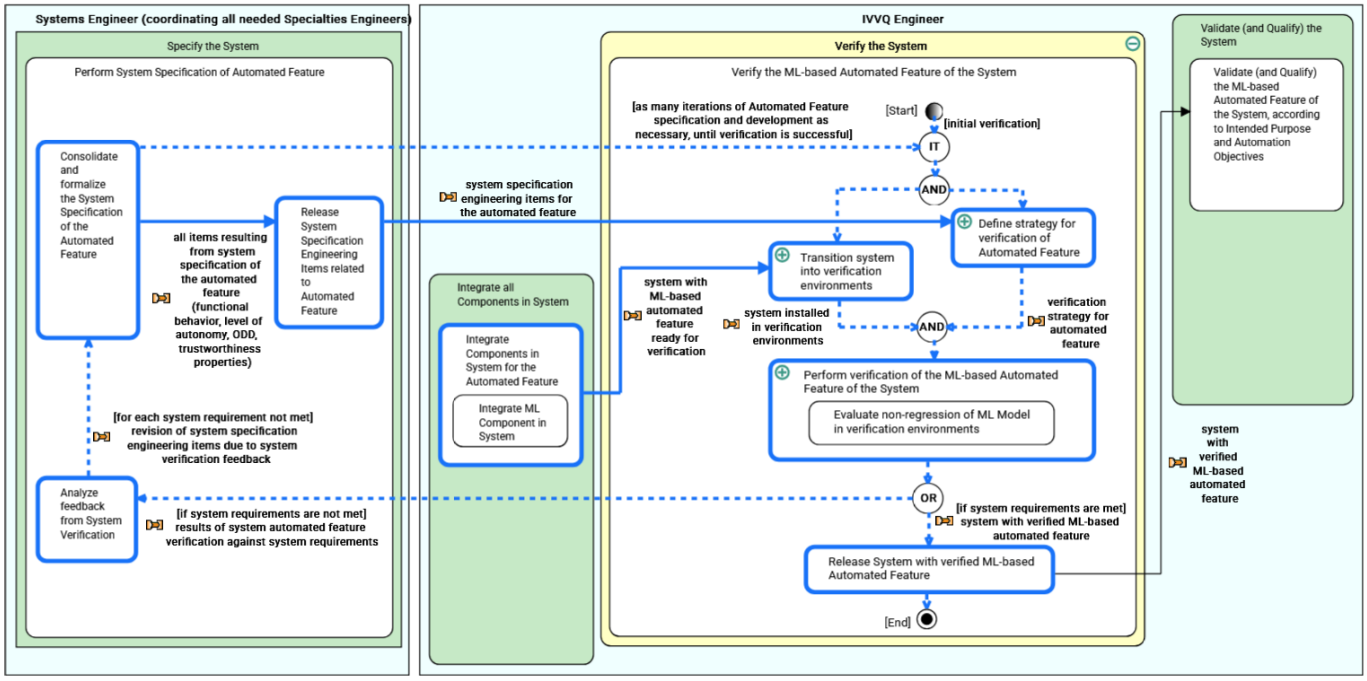


Figure 8. Verification Phase: verification of the ML-based feature of the system.

Testing an ML component aims to detect discrepancies between the actual and intended behaviors of ML models. Formal ML testing refers to any activity designed to reveal ML bugs, where an ML bug is defined as any imperfection in an ML item that causes a discrepancy between the model's output and the reference output. Examples of discrepancies could be due to a shift in the distribution of training and testing data, or an incorrect assessment of the suitability of the data for the task at hand; therefore, data is usually the cause of incorrect or unexpected errors.

This definition highlights three challenges to overcome. First, ML systems may have different types of "required conditions", *i.e.*, properties to verify. We can classify these as basic functional requirements (*e.g.*, correctness and model relevance) and non-functional requirements (*e.g.*, efficiency, robustness, fairness, interpretability). Different methods and metrics are required to verify such properties, so the selection of the best tools for verifying the component must be preceded by a definition of the required conditions: "What do we want to prove through testing?". Second, an ML bug may exist in the data, the learning program, or the framework. The testing strategy should address either the component itself or another "sub-component". This may make the testing more complex, since establishing a causal link between the bug and its source may be difficult, and defining a testing protocol allowing the distinction of independent and dependent variables is not trivial in an ML pipeline. Finally, testing activity may include several radically different approaches. These may include test input generation, test oracle identification, test adequacy evaluation, and bug triage. The selection of the approach must be based on a trade-off between the technical feasibility of performing such a test on the ML component and the required conditions initially formalized.

Quality control is an essential part of verifying and validating the ML component, and this can be achieved by estimating the success of the task solved by the component. Traditional metrics for regression problems include mean squared error (MSE) or mean absolute error (MAE), while classification problems can be evaluated using precision, accuracy, and recall. For classification problems, a confusion matrix depicting the distribution of true/false negatives/positives for each class is a practical tool for visualizing errors and allows most metrics to be computed (*e.g.*, precision, recall, sensitivity, specificity, F1 score, and ROC curve).

The most common evaluation protocol involves maintaining a hold-out validation set. This involves setting aside some of the data as the test set. The process involves training the model with the remaining data and tuning its parameters with the validation set, before finally evaluating its performance on the test set. The reason for splitting the data into three parts is to avoid information leaks. The main disadvantage of this method is that if a small amount of data is available, the validation and test sets will contain so few samples that tuning and evaluating the model will be ineffective. An alternative is k-fold cross-validation, which involves splitting the data into k partitions of equal size.

Another interesting approach is: Iterated k-fold validation with shuffling. This technique is useful when there are few data available and it is necessary to evaluate models as precisely as possible. Functional performance evaluation presents its own challenges. Selecting the most appropriate metrics to reflect the desired level of performance and choosing a suitable testing

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

protocol require careful consideration. However, the notion of quality control (QC) should go beyond simply estimating functional performance.

First, we note that the validation set is part of the ML algorithm design. The focus here is on the technical validity of the algorithm design, with few links to the operational constraints established in the specification phase. The influence of the training data is ignored at this stage, as traditional protocols do not necessarily take into account the informational value of the data points in each set. QC should encompass more than a simple evaluation of the ML algorithm. QC procedures should be formalized and deployed at each stage of the ML pipeline, with different objectives and verification strategies, but with one overarching objective: to ensure the quality of all processes involved in developing the ML component.

Although each domain has its own traditional ways of performing qualification (for example, data qualification has its own procedures), the link with the particularities and constraints of ML components is not always well established. Additionally, some aspects of verification and validation strategies are underestimated or not routinely considered in ML engineering. For example, data engineering information about the limits and constraints of the data should be reflected in the overall model evaluation strategy. The system in which the ML component is intended to operate must also provide its own set of constraints against which the component's compliance can be checked. This means that all parts of the ML pipeline should include specific QC procedures, and this information should be communicated to the relevant parts of the pipeline to inform the overall evaluation of the component's quality.

G. Maintenance and In-Service Support

Once the AI-based System has been validated and qualified, it can be deployed and declared in-service. Maintaining AI systems is a complex and evolving challenge. In many ways, this mirrors the rigorous and continuous effort. At its core, AI maintenance is not just about ensuring a model functions as intended at deployment; it is also about safeguarding its performance, reliability, and trustworthiness throughout its entire lifecycle. This is particularly important because AI systems are increasingly being integrated into high-stakes areas where failures can have severe or even catastrophic consequences. Maintaining AI systems is a complex and evolving challenge. In many ways, this mirrors the rigorous and continuous effort. At its core, AI maintenance is not just about ensuring a model functions as intended at deployment; it is also about safeguarding its performance, reliability, and trustworthiness throughout its entire lifecycle. This is particularly important because AI systems are increasingly being integrated into high-stakes areas where failures can have severe or even catastrophic consequences.

Robustness is a core concept in AI maintenance, referring to an AI system's ability to perform reliably in unexpected or adversarial conditions. Robustness is threatened by various factors during development and deployment. In development, the integrity of training data is crucial. Data can be compromised by noise from errors in data collection, annotation, or processing, or by data poisoning, where incorrect data is injected to degrade performance. This can lead to a model that performs well in testing but fails in real-world applications. Another threat is the backdoor attack, where an attacker embeds a hidden trigger in the model. When activated, the model's behavior can be manipulated without detection. These vulnerabilities are concerning in distributed learning environments, such as federated learning, where multiple parties contribute to the training process without full data visibility.

Once deployed, an AI model faces new challenges that can undermine its robustness. Adversarial examples, crafted inputs designed to deceive the model, exploit its sensitivity to small, often imperceptible, perturbations in the input data. An image recognition system might be misclassified due to a few pixels being altered in an image. Such vulnerabilities are dangerous in safety-critical applications like avionics, where a single misclassification can have life-or-death consequences. Another challenge in deployment is out-of-distribution generalization, the model's ability to handle inputs that differ from the data it was trained on. Real-world environments are dynamic, and data distributions can shift over time due to factors such as changing user behavior, sensor degradation, or evolving contextual factors. A model that performs well on its training data may struggle when faced with these shifts, leading to degraded performance or unexpected failures.

To address these challenges, AI maintenance has been proposed. It is a process akin to the maintenance of complex systems. An AI system requires regular monitoring and testing to remain reliable and maintain robustness. Inspection and diagnosis involve probing the model to identify vulnerabilities, anomalies, or degradation. Testing the model against adversarial examples is an example of monitoring for data drift. Soliciting feedback from users helps to understand the model's real-world behavior. Fixing and updating are next. This can be recalibrations or interventions like hardening the model against specific threats. Modules can be replaced if they are flawed. The cost and complexity of these activities depend on the identified issues and the requirements of the application.

An AI model inspector is a proactive framework that goes beyond the passive documentation provided by tools like model cards or data-sheets. Detect potential risks, such as back-doors, adversarial vulnerabilities, or data drift, and then take corrective actions to mitigate these risks. This could involve retraining the model on updated data, applying patches to address specific vulnerabilities, or even triggering a complete overhaul if the model's performance has degraded. The inspector framework represents a shift from reactive to proactive maintenance, where potential issues are identified and addressed before they lead to failures.

IV. Trustworthiness Attributes and Assessment

Trustworthiness is fundamental for the successful development and adoption of AI-based critical systems. Thus, trustworthiness assessment [76] can be defined as the process of evaluating and determining the level of trustworthiness of

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

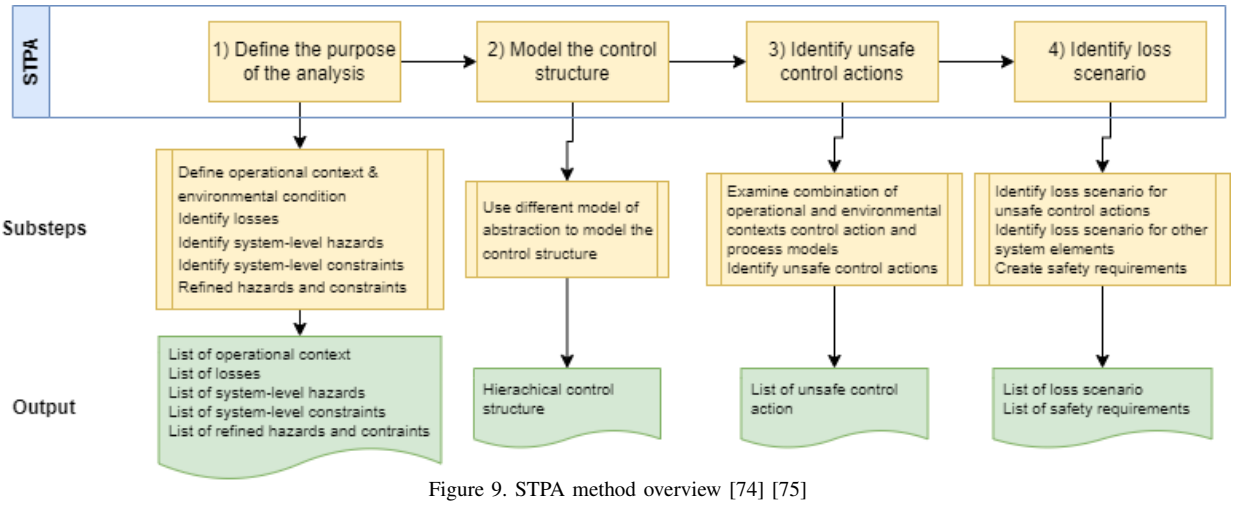


Figure 9. STPA method overview [74] [75]

a given characteristic, such as robustness [77] [78], accuracy, reliability [26], or effectiveness, in the context of AI systems engineering.

Nevertheless, it is very misleading to only judge how good an AI system is based on how accurate it is. It is also difficult to test and check the quality of software in the traditional way, and it is even difficult to measure test coverage at all. Trust and trustworthiness are complex, and so one of the main issues we face is to establish objective attributes such as accountability, accuracy, controllability, correctness, data quality, reliability, resilience, robustness, safety, security, transparency, explainability, fairness, privacy, and compliance with regulatory actors. We need to map these attributes onto the AI processes and its lifecycle and provide methods and tools to assess them. This highlights the importance of quality requirements, which are non-functional requirements and are particularly challenging in AI systems, although many of them can be considered in any critical system. Furthermore, this can also include risk and process considerations. The attributes and values for these requirements depend on things like how important the application is, what the AI system is used for, how it will be used, and the people involved. So, in some situations, some attributes may be more important than others, and new attributes may be added to the list [79]. Clear specifications of the non-functional requirements will help clarify these conflicts and can also encourage innovation that solves some of these conflicts, allowing us to fulfill more of them at the same time.

A. Risk analysis related to trustworthiness relationships between stakeholders

All interactions between the stakeholders (*e.g.*, engineers, operators, end-users, certification authorities, insurance companies, etc.) and the system are addressed by the trustworthiness relationships dimension.

Trustworthiness relationships must be established at each phase of the System lifecycle. They must also be maintained at each phase. This applies from engineering and design, until operation in a target environment. Indeed, during the engineering and design phases, engineers must be able to build trust on the system they will deliver to operators, which is an essential step in the process. Ultimately, operators must have confidence in the system features they will use.

The way in which trustworthiness relationships are established is dependent on the automation objectives that need to be achieved, as well as the environmental and human conditions that must be taken into account when operating in a trustworthy manner. As a result, trustworthiness relationships need to be analyzed from the viewpoint of each stakeholder involved in the automation Objectives, and defined and refined so that they can be supported by the system.

The dimension of trustworthiness relationships requires the application of an AI-specific risk analysis (see Figure 11). Indeed, due to the high level of uncertainty and unpredictability of the AI-based Automated Features outputs and behaviors, a new risk analysis approach related to the dimension of trustworthiness relationships is needed. Various techniques for hazard analysis such as Failure Modes and Effects Analysis (FMEA), Fault Tree Analysis (FTA), Hazard and Operability Analysis (HAZOP), System Theoretic Accident model and Processes (STAMP) and System Theoretic Process Analysis (STPA) are common. The STAMP framework is an accident causality model that provides a new paradigm for STPA.based system safety engineering.

In our conte)xt, the Confiance.ai research program has proposed a methodological process. The process relies on both the unified approach for trustworthiness assessment defined in our previous work [29] and the STPA method (see Figure 9), which is identified as relevant for analysis purposes.

STPA [75] is a system approach that considers potential dysfunctional system's characteristics and behaviors as a system control problem and not only as a problem of component failure. It does not replace traditional failure analysis approaches but complements them. In *Confiance.ai*, STPA is extended beyond its traditional safety analysis domain to trustworthiness characteristics/properties risks analysis and control, to be applied for each autonomy objective and feature defined in the

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

Concept name	Definition
Misuse	[ISO 21448: 2021] usage of the system by a human in a way not intended by the manufacturer or the service provider
Error	[ARP 4754 A] A mistake made by a crew member or maintenance person, or a mistake in the requirements, design or implementation (derived from AMC 25.1309). [ISO 26262: 2011] A discrepancy is when a value or condition is not the same as the true value. It could be a computed value. Or an observed value. Or a measured value. Or a theoretically correct value. Note 1: An error can arise as a result of unforeseen operating conditions or due to a fault within the system, subsystem or component being considered. Note 2: A fault can manifest itself as an error within the considered element and the error can ultimately cause a failure.
Failure	An occurrence, which affects the operation of a component, part or element such that it can no longer function as intended, (this includes both loss of function and malfunction). Note: errors may cause Failures, but are not considered Failures. (AMC 25.1309)
Hazard	Definition from STPA: A system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to an accident (loss). [ARP 4754 A] Extended for trustworthiness: A condition resulting from failures, external events, errors or a combination of these factors affecting trustworthiness.
Worst-case environmental condition	Environmental, non-controllable context
Risk	[ARP 4754 A] The level of severity of an occurrence is dependent on its frequency (probability).
Accident-Loss	Definition from STPA, extended for trustworthiness: An undesired or unplanned event that results in a loss, including loss of human life or human injury, property damage, environmental pollution, mission loss, trustworthiness loss etc. Definition from STAMP: An undesired or unplanned event that causes loss, damage, or injury [80].
Mitigation	Any means enabling risk reduction (occurrence likelihood and/or impact with barriers) at any step of the System of interest lifecycle (*e.g.*, specification, design, training...).
Loss/Damage	[STPA Handbook] extended for trustworthiness: A loss involves the loss of something of value to stakeholders. They may consider this to include anything from loss of human life or injury to property damage, environmental pollution, loss of mission, loss of reputation, loss or leak of sensitive information or loss of trustworthiness.

Figure 10. Safety Analysis concept definition [29]

ODD. Therefore, STPA can be applied to analyze and mitigate risk of trustworthiness loss.

Remind that STPA is a system-theoretic safety analysis method designed to identify and mitigate risks in complex systems by focusing on control structures and unsafe interactions rather than just component failures. It is particularly useful for autonomous, cyber-physical, or human-machine systems where traditional hazard analysis may fall short. The process is structured into four key steps (see Figure 9), each building on the previous one to ensure traceability from high-level losses to specific scenarios:

1) Identify accidents and hazards list, and associated unacceptable losses related to the System of Interest (either material losses or immaterial losses, *e.g.,* mission, trustworthiness loss. Identify also the boundary of the analysis and defines those losses and hazards that must be prevented as well as high-level operational and system constraints/requirements needed to prevent them.
2) Identify and model the control structures, starting from the operational environment, and refining through the abstraction layers of the system analysis (*i.e*, System level, Architectural level, AI Component level)
3) Identify Unsafe Control Actions (UCAs) leading to hazards and losses, and specify requirements and constraints to prevent them.
4) Identify scenarios leading to UCAs or hazards, and specify requirements to mitigate the risks."

To define the process of trustworthiness risk analysis, traditional Safety Analysis concepts of the aeronautical business domain have been considered (*e.g.*, failure). Other business domains (*e.g.*, railway, automotive...) could introduce their dedicated concepts (see Figure 10). The resulting process is described in Figure 11, as a pattern that needs to be iteratively applied at several steps of the end-to-end method.

The principle is that a specific Trustworthiness Engineering team, once it has analyzed the trustworthiness properties, has also to analyze the trustworthiness loss risks, according to the approach AI specific risk analysis approach. It has also to perform potential traditional risks analysis (*e.g.*, Safety failure modes analysis approaches). In addition to the system specification including Operational Design Domain (ODD) analysis, and trustworthiness assessment processes, a risk analysis process is essential to address and mitigate the risks related to AI technologies, based on add-hoc control-structures specifications.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

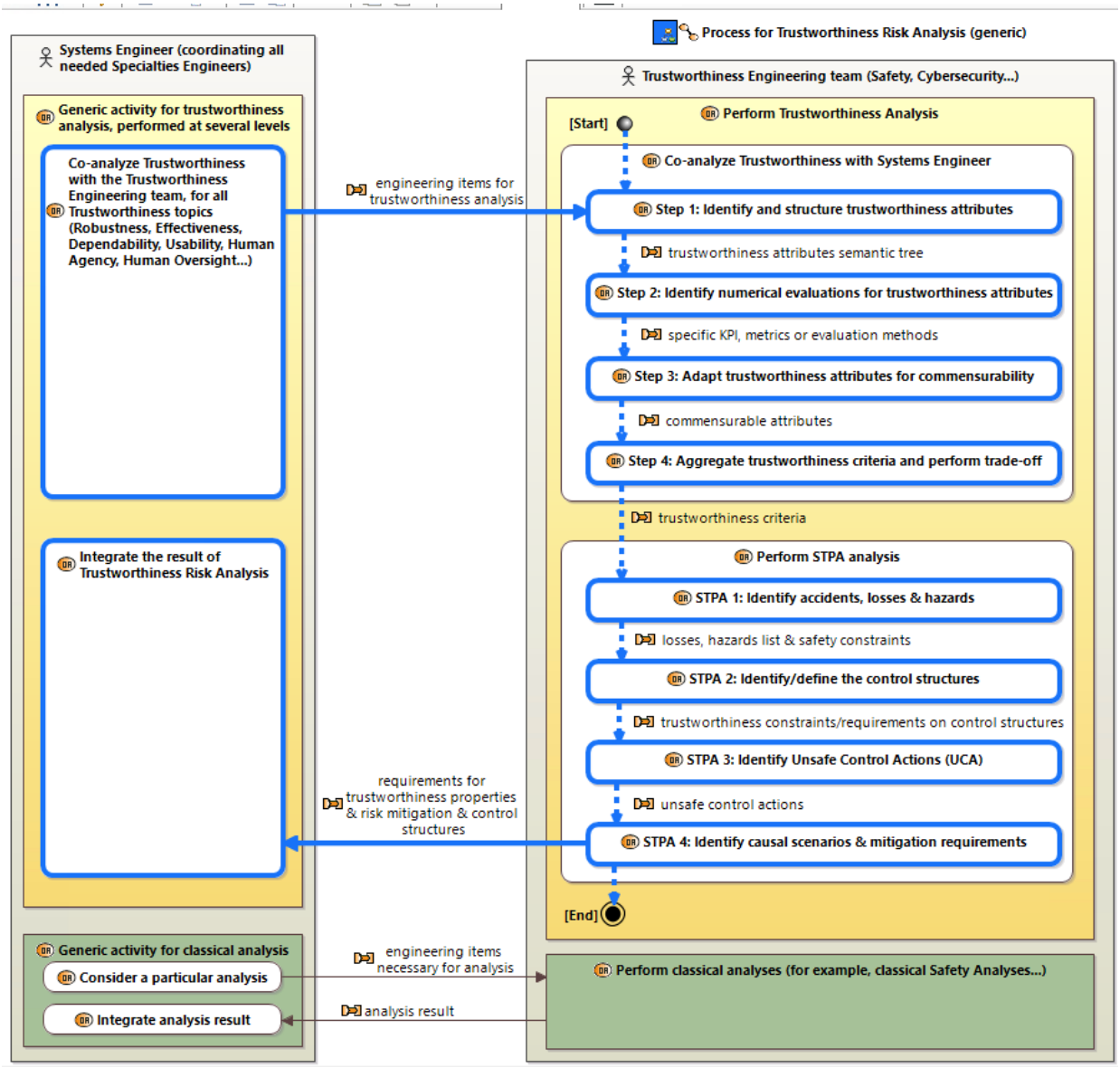


Figure 11. Trustworthiness Risk Analysis - Generic process

B. *Local trustworthiness assessment*

Thus, by leveraging system engineering best-practices, ML development workflows, and testing procedures, the end-to-end methodology ensures that trustworthiness attributes are embedded in every stage of the AI system life-cycle, from conception to maintenance. The Confiance.ai framework focuses on the following attributes:

- **Robustness** covering **Safety and Security**: High-risk systems must be as resilient as possible against errors, faults or inconsistencies that may occur within the system or the environment in which it operates, and also against attempts by unauthorized third parties to alter its use, outputs or performance by exploiting system vulnerabilities; furthermore, the technical solutions aiming to ensure the cybersecurity of high-risk AI systems must be appropriate to the risks and circumstances. Various perturbations (*i.e.*, variations in input data and operating conditions) should not be an issue for robust AI systems. Therefore, an AI-based system must meet rigorous safety and security requirements [81] (see Figure 12):
  - Safety analysis and certification based on standards.
  - Cybersecurity counter-measures, integrated on the AI pipeline.

  This requires :
  - Adversarial robustness, ensuring the system is not easily manipulable by adversarial attacks.
  - OOD Robustness (Out-Of Distribution), the system must generalize well across different environment and be trained on diverse datasets.
  - model monitoring, ensuring a continuous evaluation

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

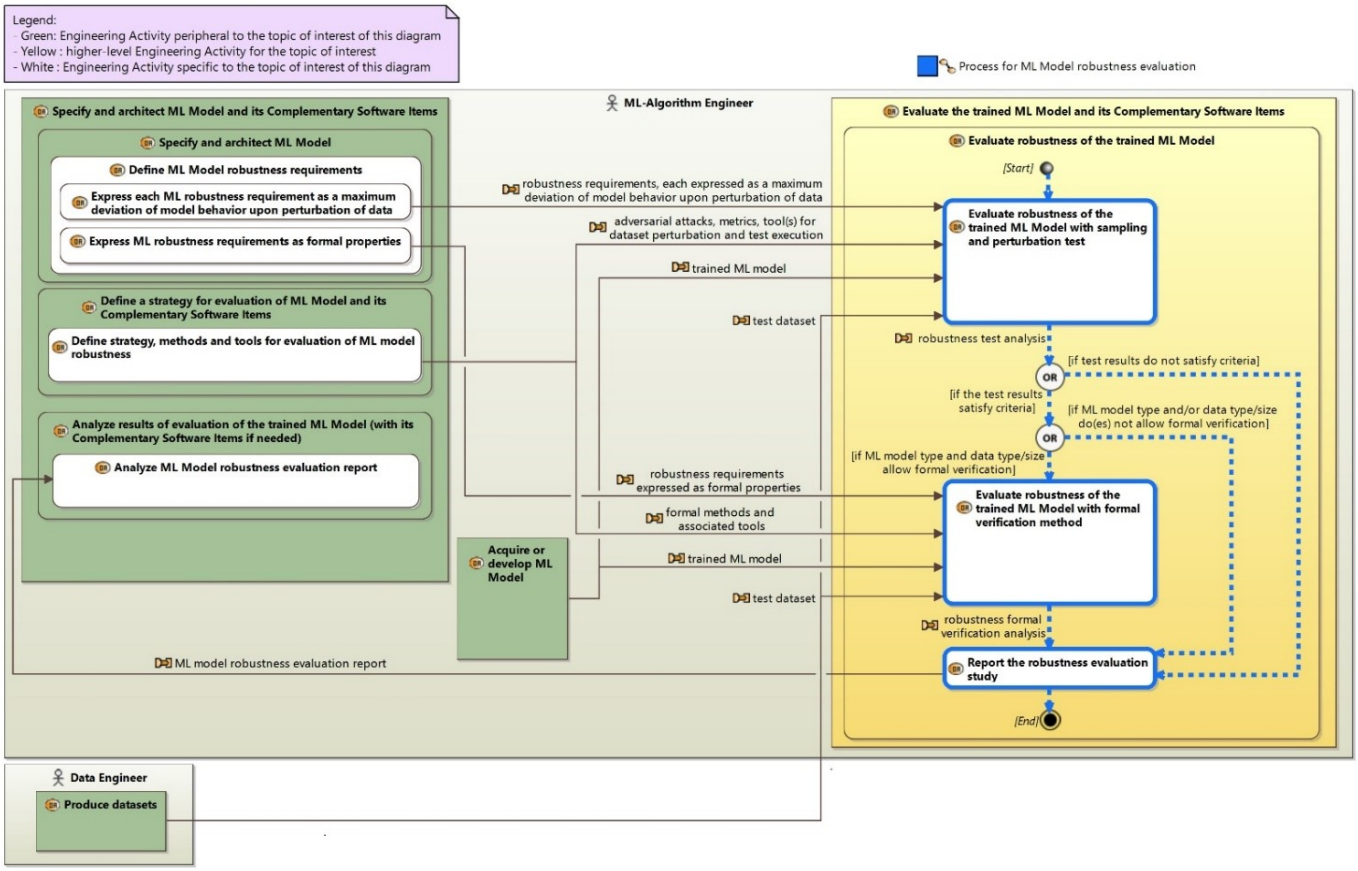


Figure 12. ML model robustness evaluation process considering two complementary strategies

of the AI models, to detect performance degradation.

To evaluate the robustness of ML model, the end-to-end methodology proposes a strategy made of two successive phases:

- Robustness test by sampling and perturbation (empirical evaluation).
- Formal verification of robustness (formal evaluation);

The test-based phase consists in comparing, on one hand, the behavior of the ML model fed with a perturbed dataset, and on the other hand, its nominal behavior.

The formal evaluation-based phase uses formal methods and tools to verify one or several mathematical properties (here, related to robustness) of the ML model. Ideally, such a property shall be formally verified on the whole Model, whatever the input data. However, practically, because of constraints on the formal verification tools, the property is formally verified only for given input data: it proves that the ML model is locally robust, at given points of the test dataset. Each phase implies a specific expression of ML model robustness requirements.

The first phase is relatively inexpensive, compared to the second one. By testing the model with different inputs and perturbations, information is obtained about the performance of the model in different scenarios and its ability to generalize well despite data perturbation. However, this type of evaluation has limited confidence because it only tests a subset of possible scenarios but it may not uncover all potential issues or weaknesses.

On the other hand, the second phase involves rigorous mathematical analysis of the model robustness: it is more expensive and time-consuming compared to the first phase. Formal verification provides a higher level of confidence in the model's performance because it is based on sound proofs. However, it is important to note that formal verification may not be possible for certain types of models due to their complexity or lack of formal specifications. In this sense, the adoption of a formal verification to evaluate the robustness of ML models depends on certain constraints such as the acceptability of formal proofs, the compatibility of the verification tools with the ML model algorithm, and the dimension of the data space.

The interest of starting with sampling and perturbation test is to quickly identify any major issues or weaknesses in the model. If the model performs well in this step, it can then be subjected to formal

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

verification to obtain a more expensive but also more reliable result.

This combination of approaches allows for the most comprehensive evaluation of the model robustness, considering both cost-effectiveness and confidence.

- **Transparency, Explainability, Interpretability and Comprehensibility**. The principle of "having a human in the loop" is at the core of responsible and trustworthy AI. Transparency is vital for effective human control and oversight, including instructions for safe use and information about the level of accuracy, robustness, and cybersecurity of the critical AI system. This enables us to: (i) Properly understand the relevant capacities and limitations of the system and monitor its operations, including in view of detecting and addressing anomalies, dysfunctions and unexpected performance; (ii) Maintain awareness of the potential tendency to automatically rely or over-rely on the output produced by the system; (iii) Accurately interpret the system's output; and (iv) Decide not to use the system or otherwise disregard, override or reverse the system's output. Moreover, trustworthy AI should be transparent and its decisions should be interpretable where
  - Explainability deals with the capability to provide the human with relevant information on how an AI application is coming to its result.
  - Interpretability relates to the capability of an element representation (an object, a relation, a property...) to be associated with the mental model of a human being. It is a basic requirement for an explanation.
  - Comprehensibility refers to the capability of an element representation (an object, a relation, a property...) to be understood by a person according to its level of expertise or background knowledge.

  This requires:
  - Post-hoc explainability tools, to provide insights into model decisions.
  - model simplification strategies to enhance interpretability.
  - Human-in-the-loop validation to ensure AI decisions align with expert knowledge.

  There is a profusion of methods, tools, and solutions available, each with its own set of advantages, drawbacks, and trade-offs [84]. The many different approaches show how tricky it is to make sure that AI and machine learning models can explain their predictions and decisions. Choosing the right way to make models explainable is a technical and strategic decision. It depends on the unique needs and limits of the people it will be used by, the specific example it will be used for, and the wider situation in which the AI system will be used. What works for a medical diagnosis model may not work for the aeronautic domain, and what regulators expect can be very different from what end-users or business stakeholders expect. The Confiance.ai program provides a "Methodological Guideline for Explainability" (https://catalog.Confiance.ai/) which is designed to be a complete guide to help people use AI. It will explain why explainability is important, highlight the many available methods, and offer guidance on selecting the most suitable approach based on the specific situation.
- **Fairness and Bias Mitigation**. A key concern with data-driven AI (such as ML) is the amplification of biases. Therefore, we have first to take appropriate measures to detect, prevent and mitigate possible biases and to use high-quality datasets for training, validation, and testing, as the output of the AI system depends largely on the quality of the training data. The data must be relevant, sufficiently representative and, to the best extent possible, free of errors and complete. AI models should be free from discriminatory biases. This involves:
  - Bias detection and correction techniques, in the data processing and model training phases.
  - Regulatory alignment with fairness standards.

The end-to-end methodology integrates those attributes throughout the AI system life-cycle, namely in:

- **Operational Design Domain (ODD) definition**: Critical AI systems are subject to rigorous regulatory requirements, including conformity assessments and post-market surveillance. The EU AI Act establishes a risk classification system for AI systems based on their intended purpose. This means that the use for which an AI system is intended by the provider, including the specific context and conditions of use, determines its risk classification. This ensures that regulatory scrutiny aligns with the system's anticipated function and impact.
  - Define the operational boundaries where the AI system is expected to function reliably.
  - Establish clear environmental constraints for the AI-system's development.

  The ODD is a description of measurable foreseeable operating conditions within which a system/component shall operate. A traceability property shall be assured between the different levels of ODD (system, subsystem or component).
- **Systems Engineering**
  - Ensure AI system-level requirements are defined in alignment with overall system objectives.
  - Align AI-based system requirements with preexisting system engineering standards and certification guidelines.
- **Data Engineering and Data Quality Assessment**
  - Rely on a robust data pipeline to guarantee data integrity, consistency, and traceability across the engineering cycle.
  - Implement bias mitigation strategies at the data collection and processing stages.
  - Use adaptive data augmentation strategies to improve data diversity and model generalization to distribution shifts and operational scenarios.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

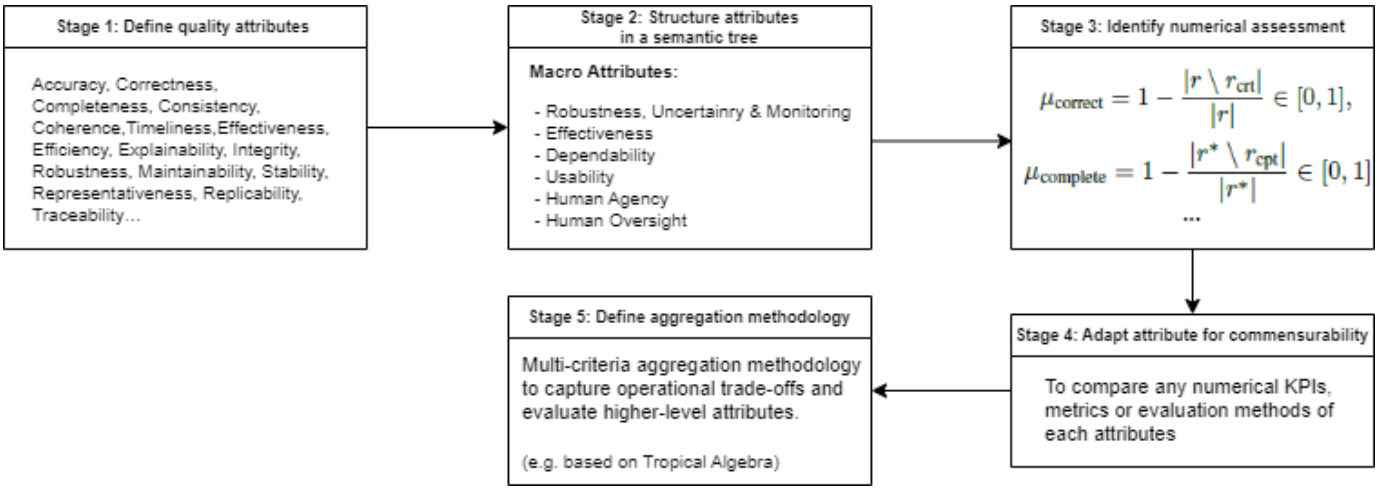


Figure 13. The unified approach based on MCDA [82] [83]

- **ML Algorithm Engineering**
  - Use ML robustness techniques, designed to handle perturbation and adversarial outputs.
  - Incorporate explanability techniques to have understandable decisions.
  - Apply Uncertainty quantification techniques to asses the model's confidence.
- **Verification and Validation**
  - Perform extensive simulation-based testing to asses performances under edge cases.

In addition, measuring how trustworthy AI systems are is tricky. The ideas behind them are complicated, the characteristics they produce are different, and you can't always compare them. The Confiance.ai program proposes an innovative way to measure trustworthiness using (max,+) algebra [85] based on a complete hierarchical model that brings together different properties, such as how strong, effective, dependable, easy to use and human agency, and human oversight) into a single assessment method. This offers advantages over traditional weighted averaging methods by better handling extreme values and preserving sensitivity to critical indicators, while maintaining sensitivity to critical indicators to provide detailed, understandable assessments of AI-based system trustworthiness.

C. Global Trustworthiness Assessment

As it is not straightforward to select the relevant attributes for assessing AI trustworthiness, given that the choice depends on the context of application. This context is modeled according to a number of elements, including the Operational Design Domain (ODD), the intended domain of use, the nature and roles of the stakeholders, and so on. The attributes may be quantitative, typically comprising numerical values derived from measurements or providing a comprehensive statistical overview of a phenomenon. Alternatively, they may be qualitative, based on the detailed analysis and interpretation of a limited number of samples. Then, the second activity mentioned above on the characterization of the trustworthiness evaluation is broken down into several activities, according to the Multi-Criteria Decision Aiding (MCDA) method [82] [83]. Those are:

1) *Define trustworthiness characteristics*. All the characteristics of the considered item are identified and described (*i.e.*, their name, properties).
2) *Structure attributes in a semantic tree*. Characteristics (*i.e.*, quality attributes) are organized in a tree, from the most general down to the leaf characteristics.
3) *Identify numerical evaluations*. Each characteristic is typed by a numerical value domain.
4) *Adapt attribute for commensurability*. Characteristics can follow different forms of distribution with different value domains. The purpose is to make them compatible so that they can be compared and operated together.
5) *Define the aggregation methodology*. MCDA enables one to explore several solutions, compare them, and to keep the best one.

Once the list of relevant attributes has been defined, the aggregation of several attributes remains complex due to issues of commensurability. This is because the attributes in question are not of the same unit; for example, combining "*oranges and apples*" is not a meaningful exercise. Furthermore, it is necessary to make compromises and arbitrate between the attributes. This means that the value of each attribute must be transformed into a scale that is consistent across all attributes and reflects the preferences of a stakeholder. Furthermore, the values assigned to the scales for the various criteria must be aggregated. These elements represent the primary stages of a problem-solving process that employs an Multi-Criteria Decision Aiding (MCDA) approach [85].

MCDA is a generic term for a collection of systematic approaches developed specifically to assist one or several decision makers in assessing or comparing alternatives based on multiple criteria [86]. The challenge lies in the fact that the decision-making criteria are often numerous, interdependent, and occasionally in conflict with each other. For instance, there

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

may be a conflict between effectiveness and other criteria, such as robustness, explainability, or affordability. The viewpoints are quantified through the use of attributes. Aggregation functions are frequently used to facilitate comparisons between alternatives evaluated on the basis of multiple, potentially conflicting criteria. This is achieved by synthesizing their respective performances into overall utility values. These functions must be sufficiently expressive to align with the preferences of the stakeholders involved, enabling the identification of the most preferred alternative or facilitating the negotiation of compromises among the criteria. It is important to note that improving one criterion may necessitate a trade-off in another.

V. Conclusion and future works

The Confiance.ai program has evolved since its kick-off in 2021, with a first year dedicated to covering the academic and industrial state of the art related to ML-based system design. Subsequent years (2022-2023) were dedicated to the accurate characterization of industrial use cases, the development and evaluation of technological components to address specific aspects of reliability, and the construction of an end-to-end method revisiting all stages of the engineering cycle for the design, integration, and evaluation of ML components [9]. The last year (2024) encompasses the evaluation of this end-to-end method, the completion and dissemination of key results, and the guarantee of their continuation and sustainability under the aegis of a new research initiative currently under construction. To facilitate the adoption of the tool-based methodology by industry, several implementations of the 2023 version have been carried out on use cases.

These experiments have demonstrated the importance of integrating diverse tools and methods to address expectations regarding trusted ownership, as illustrated by the following two examples: In a use case involving autonomous driving, the analysis of dataset diversity reveals a limited presence of night-time images, prompting the generation of synthetic night-time data. This data exhibits a "domain gap" and undergoes "domain adaptation" prior to integration into the model training data. These tools, instrumental in the construction of datasets, will also be reused in the supervision stage of the use case. In an aeronautical use case called LARD for "Landing Approach Runway Detection" [87] and represented Figure 14, a data quality supervision module is incorporated to consolidate the confidence score of an ML model (see Figure 14). In this example, local image quality estimators (*e.g.*, level of blur, brightness) are taken into account in the detection zone of the landing strip that is being detected. The combination of these indicators with the other indicators intrinsic to the model facilitates the establishment of a level of confidence for the system component. In addition to providing a numerical value, this implementation serves as a tool to facilitate the interpretation of model and data errors.

The Confiance.ai program is opening up two major outcomes to the community as a "digital common good". First, it provides a body of knowledge describing an end-to-end method of AI engineering. This makes it possible to characterize and qualify the trustworthiness of a data-driven AI system and integrate it into industrial products and services. Second, this method is applicable to any sector of activity. A catalog of developed and/or mature technological components to increase the level of trust in AI integrated into critical systems.

The Body of Knowledge (BoK) is one of the main outcomes because it provides access to a navigable version of this end-to-end methodology that covers the activities structuring the engineering cycle of a critical system based on ML (https://bok.Confiance.ai/). This compendium of expertise from multiple disciplines is a corpus that articulates the system level with the model and data levels in the engineering process. It is continuously updated and expanded and is expected to continue beyond the program. The content provided in the body of knowledge is structured with an end-to-end engineering method in mind and can be navigated through different roles in this process, namely through the field of application of different engineering profiles: These roles include, but are not limited to, the following: machine learning algorithm engineer, data engineer, embedded software engineer, IVVQ (Integration, Validation, Verification and Qualification) engineer or system engineer.

The following simplified high-level view of the BoK is presented as a gateway to the end-to-end method for engineering trustworthy ML-based systems.The body of knowledge presents the stages of the methodology, from operational analysis and specification of the function of the system that one wishes to automate through the use of ML technology, to verification/validation/qualification, including the development and implementation of the ML model. The navigation through each stage and according to each role facilitates the visualization of the activities, sub-activities and workflow to be carried out when developing a reliable ML-based system.This corpus is thus a compendium of expertise from multiple disciplines because it links the system level with the model and data levels in the engineering process.It is continuously updated and expanded, and this is planned beyond the program.

The catalog (https://catalog.Confiance.ai/) is a web application that allows users to consult the results of the Confiance.ai program. It employs filtering and search functions (sorting, categories, etc.) to facilitate navigation through the various results, which can be either documents or software. Results categorized as "documentary" are exclusively of a literary nature, including reports (studies or benchmarks), state of the art, doctoral theses or good practice guides. "Software" results are components intended to be run directly or through another application, such as a web application, a library, a plugin or a binary executable.

Acknowledgment

This work has been supported by the French government under the "France 2030" program, as part of the SystemX Technological Research Institute within the Confiance.ai Program (www.Confiance.ai) and the *CSIA* (Confiance dans les Systèmes d'IA) project.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

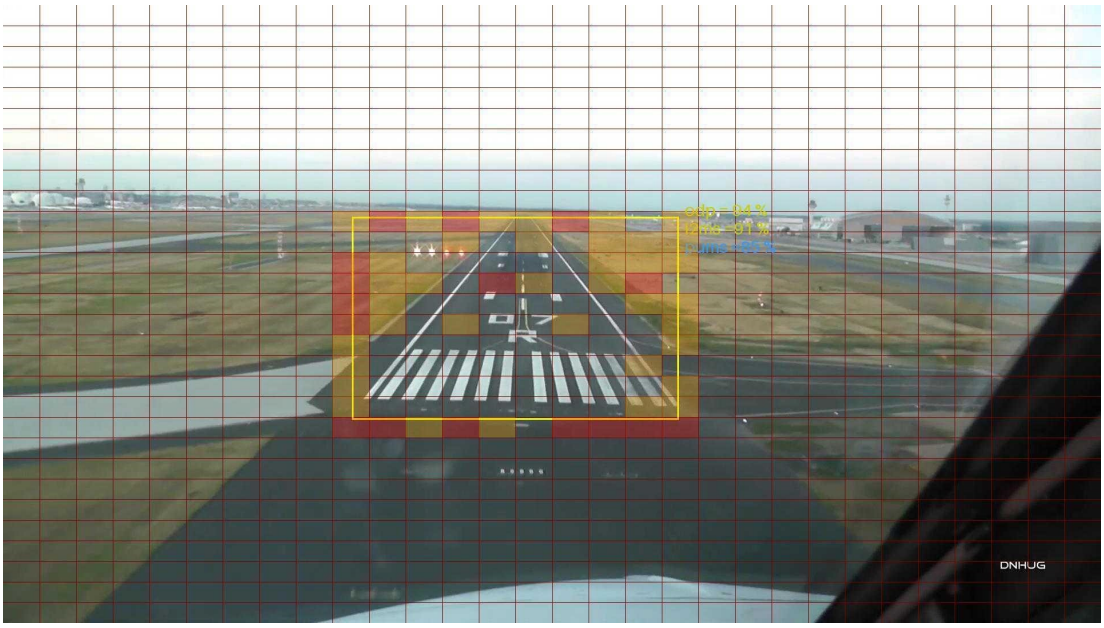


Figure 14. Example of the implementation of a supervision tool on the LARD [87] use-case

REFERENCES

[1] K. Quintero et al., "An end-to-end method for operationalizing trustwothiness in AI-based critical systems", in *15th International Conference on Performance, Safety and Robustness in Complex Systems and Applications PESARO 2025*, 2025.

[2] European Commission, *Proposal for a Regulation of the European Parliament and of the Council laying down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act) and Amending Certain Union Legislative Acts*, 2021.

[3] M. Felderer and R. Ramler, "Quality Assurance for AI-Based Systems: Overview and Challenges (Introduction to Interactive Session)", in *International Conference on Software Quality*, Springer, 2021, pp. 33–42.

[0] ISO/IEC 25024:2015, *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of data quality*, 2015.

[4] H. Liu et al., "Trustworthy AI: A computational perspective", *ACM Transactions on Intelligent Systems and Technology*, vol. 14, pp. 1–59, 2022.

[5] HLEG, *A definition of AI: Main capabilities and scientific disciplines*, Definition developed for the purpose of the deliverables of the High-Level Expert Group on AI, 2018.

[6] A. Horneman et al., *AI Engineering: 11 Foundational Practices-Recommendations for Decision Makers from Experts in Software Engineering, Cybersecurity, and applied Artificial Intelligence White Paper DM19-0624, 06.06*. Carnegie Mellon University, Software Engineering Institute (SEI), 2019.

[7] M. Gonzalez et al., "Introducing RUM: A Methodological Contribution for Engineering Trustworthy AI Components in Industrial Systems", in *Proceedings of the AAAI Symposium Series*, vol. 7, 2025, pp. 153–160.

[8] M. Adedjouma et al., "Engineering dependable AI systems", in *17th IEEE Annual System of Systems Engineering Conference (SOSE)*, 2022, pp. 458–463.

[9] A. Awadid et al., "Ensuring the reliability of AI systems through methodological processes", in *2024 IEEE 24th International Conference on Software Quality, Reliability and Security (QRS)*, IEEE, 2024, pp. 139–146.

[10] A. Avizienis et al., "Basic concepts and taxonomy of dependable and secure computing", *IEEE Transactions on Dependable and Secure Computing*, vol. 1, pp. 11–33, 2004.

[11] J. Cho et al., "Stram: Measuring the trustworthiness of computer-based systems", *ACM Computing Surveys (CSUR)*, vol. 51, pp. 1–47, 2019.

[12] NSTC, *The National Artificial Intelligence Research and Development Strategic Plan: 2019 Update*. National Science and Technology Council (US), 2019.

[13] E. Schmidt et al., "National security commission on artificial intelligence (AI)", National Security Commission on Artificial Intelligence, Tech. Rep., 2021.

[14] UNESCO, *Recommendation on the Ethics of Artificial Intelligence*, 2022. Accessed: Nov. 8, 2024. [Online]. Available: https://unesdoc.unesco.org/ark:/48223/pf0000381137.

[15] OECD, *Recommendation of the Council on Artificial Intelligence*, Legal Instruments, May 2019. Accessed: Nov. 3, 2024. [Online]. Available: https://legalinstruments.oecd.org/en/instruments/OECD-LEGAL-0449.

[16] OCDE, *G7 Hiroshima Process on Generative Artificial Intelligence (AI)*, 2023. DOI: https://doi.org/https://doi.org/10.1787/bf3c0c60-en. [Online]. Available: https://www.oecd-ilibrary.org/content/publication/bf3c0c60-en.

[17] EASA, *Concept Paper First Usable Guidance for Level 1 Machine Learning Applications*, 2021. [Online]. Available: https://www.easa.europa.eu/en/easa-concept-paper-first-usable-guidance-level-1-machine-learning-applications-proposed-issue-01pdf.

[18] P. Ala-Pietilä et al., *The Assessment List for Trustworthy Artificial Intelligence (ALTAI)*. European Commission, 2020.

[19] M. Mock et al., *Management system support for trustworthy artificial intelligence*, 2021.

[20] ISO/IEC DIS 42001, *Information technology — Artificial intelligence — Management system*, 2022.

[21] B. Stanton et al., "Trust and artificial intelligence", *NIST preprint*, vol. 10, 2021.

[22] IEEE 7000, *IEEE Standard Model Process for Addressing Ethical Concerns during System Design*, 2021.

[23] ETSI, *Securing Artificial Intelligence (SAI); Mitigation Strategy Report*, 2021.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

[24] J. Mattioli et al., "Empowering the trustworthiness of ML-based critical systems through engineering activities", *arXiv preprint arXiv:2209.15438*, 2022.

[25] B. Braunschweig et al., "The wall of safety for AI: Approaches in the confiance.ai program", in *Workshop on Artificial Intelligence Safety (SAFEAI)*, 2022.

[26] J. Mattioli et al., "AI engineering to deploy reliable AI in industry", in *2023 Fifth International Conference on Transdisciplinary AI (TransAI)*, 2023, pp. 228–231.

[27] R. Gelin, "Confiance.ai program software engineering for a trustworthy AI", in *Producing Artificial Intelligent Systems: The Roles of Benchmarking, Standardisation and Certification*, Springer, 2024, pp. 11–29.

[28] A. Awadid et al., "AI Systems Trustworthiness Assessment: State of the Art", in *Workshop on Model-based System Engineering and Artificial Intelligence-MBSE-AI Integration 2024*, 2024.

[29] A. Awadid et al., "Ensuring the reliability of AI systems through methodological processes", in *2024 IEEE 24th International Conference on Software Quality, Reliability and Security (QRS)*, 2024, pp. 139–146.

[30] A. Awadid, B. Robert, and B. Langlois, "Mbse to support engineering of trustworthy AI-based critical systems", in *12th International Conference on Model-Based Software and Systems Engineering*, 2024.

[31] A. Awadid et al., "Towards engineering processes to guide the development of trustworthy ML systems", in *2024 IEEE International Symposium on Systems Engineering (ISSE)*, IEEE, 2024, pp. 1–6.

[32] V. Liubchenko, "Specific aspects of software development process for ai/ml-based systems", in *2022 IEEE 17th International Conference on Computer Sciences and Information Technologies (CSIT)*, IEEE, 2022, pp. 470–473.

[33] P. Koopman, F. Fratrik, et al., "How many operational design domains, objects, and events?", *Safeai@ aaai*, vol. 4, no. 4, 2019.

[34] F. Pedregosa et al., "Scikit-learn: Machine learning in python", *the Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[35] J. Mattioli et al., "Information quality: The cornerstone for ai-based industry 4.0", *Procedia Computer Science*, vol. 201, pp. 453–460, 2022.

[36] L. Mattioli et al., "Data curation matters: Model collapse and spurious shift performance prediction from training on uncurated text embeddings", *arXiv preprint arXiv:2506.17989*, 2025.

[37] M. Mazumder et al., "Dataperf: Benchmarks for data-centric AI development", *arXiv preprint arXiv:2207.10062*, 2022.

[38] J. Jakubik et al., "Data-centric artificial intelligence", *arXiv preprint arXiv:2212.11854*, 2022.

[39] M. Jarrahi, A. Memariani, and S. Guha, "The principles of data-centric AI (DCAI)", *arXiv preprint arXiv:2211.14611*, 2022.

[40] G. Mountrakis and B. Xi, "Assessing reference dataset representativeness through confidence metrics based on information density", *ISPRS journal of photogrammetry and remote sensing*, vol. 78, pp. 129–147, 2013.

[41] H. Delseny et al., "White paper machine learning in certified systems", *arXiv preprint arXiv:2103.10529*, 2021.

[42] Z. Gong et al., "Diversity in machine learning", *IEEE Access*, vol. 7, pp. 64 323–64 350, 2019.

[43] M. Dereziński, "Fast determinantal point processes via distortion-free intermediate sampling", in *Conference on Learning Theory*, PMLR, 2019, pp. 1029–1049.

[44] Z. Gong et al., "Diversity-promoting deep structural metric learning for remote sensing scene classification", *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, pp. 371–390, 2017.

[45] C. Zhang et al., "Active mini-batch sampling using repulsive point processes", in *Proceedings of the AAAI conference on Artificial Intelligence*, vol. 33, 2019, pp. 5741–5748.

[46] Y. LeCun et al., "Gradient-based learning applied to document recognition", *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, 1998.

[47] V. Gurupur and M. Shelleh, "Machine learning analysis for data incompleteness (MADI): Analyzing the data completeness of patient records using a random variable approach to predict the incompleteness of electronic health records", *IEEE Access*, vol. 9, pp. 95 994–96 001, 2021.

[48] C. Holden et al., "The electronic health record system and hospital length of stay in patients admitted with hip fracture", *Am J Research Nurs [Internet]*, pp. 1–5, 2015.

[49] Y. Teo et al., "Benchmarking quantum tomography completeness and fidelity with machine learning", *New Journal of Physics*, vol. 23, p. 103 021, 2021.

[50] C. Tran, *Evolutionary machine learning for classification with incomplete data*, 2018.

[51] B. Schouten et al., "Indicators for the representativeness of survey response", *Survey Methodology*, vol. 35, pp. 101–113, 2009.

[52] M. Berkesewicz, "A two-step procedure to measure representativeness of internet data sources", *International Statistical Review*, vol. 85, pp. 473–493, 2017.

[53] M. Blatchford et al., "Determining representative sample size for validation of continuous, large continental remote sensing data", *International Journal of Applied Earth Observation and Geoinformation*, vol. 94, p. 102 235, 2021.

[54] M. Qi et al., "Quantifying representativeness in randomized clinical trials using machine learning fairness metrics", *JAMIA open*, vol. 3, ooab077, 2021.

[55] Y. Geifman and R. El-Yaniv, "Selective classification for deep neural networks", *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[56] Y. Geifman and R. El-Yaniv, "Selectivenet: A deep neural network with an integrated reject option", in *International Conference on Machine Learning*, 2019, pp. 2151–2159.

[57] A. Krizhevsky, G. Hinton, et al., *Learning multiple layers of features from tiny images*, 2009.

[58] K. Pei et al., "Deepxplore: Automated whitebox testing of deep learning systems", in *proceedings of the 26th Symposium on Operating Systems Principles*, 2017, pp. 1–18.

[59] Y. Tian et al., "Deeptest: Automated testing of deep-neural-network-driven autonomous cars", in *Proceedings of the 40th International Conference on Software Engineering*, 2018, pp. 303–314.

[60] A. Odena et al., "Tensorfuzz: Debugging neural networks with coverage-guided fuzzing", in *International Conference on Machine Learning*, PMLR, 2019, pp. 4901–4911.

[61] F. Adjed et al., "Coupling algebraic topology theory, formal methods and safety requirements toward a new coverage metric for artificial intelligence models", *Neural Computing and Applications*, vol. 34, pp. 1–16, 2022.

[62] J. Bolte et al., "Towards corner case detection for autonomous driving", in *IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 438–445.

[63] J. Breitenstein et al., "Systematization of corner cases for visual perception in automated driving", in *IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1257–1264.

[64] J. Breitenstein et al., "Corner cases for visual perception in automated driving: Some guidance on detection approaches", *arXiv preprint arXiv:2102.05897*, 2021.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org

[65] T. Ouyang et al., "Corner case data description and detection", in *IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*, 2021, pp. 19–26.

[66] T. Ouyang et al., "Improved surprise adequacy tools for corner case data description and detection", *Applied Sciences*, vol. 11, p. 6826, 2021.

[67] W. Wu et al., "Deep validation: Toward detecting real-world corner cases for deep neural networks", in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, IEEE, 2019, pp. 125–137.

[68] F. Heidecker et al., "An application-driven conceptualization of corner cases for perception in highly automated driving", in *IEEE Intelligent Vehicles Symposium (IV)*, 2021, pp. 644–651.

[69] F. Heidecker, M. Bieshaar, and B. Sick, *Towards corner case identification in cyclists' trajectories*, 2019.

[70] A. Le Coz et al., "Leveraging generative models to characterize the failure conditions of image classifiers", in *The IJCAI-ECAI-22 Workshop on Artificial Intelligence Safety (AISafety 2022)*, 2022.

[71] A. Awadid et al., "A methodological framework for supporting the operational analysis of ML-based systems", in *Models and Methods for Systems Engineering*, Springer, 2025, pp. 129–141.

[72] A. Fakhouri et al., "Ml model coverage assessment by topological data analysis exploration", in *Proceedings of the AAAI Symposium Series*, vol. 4, 2024, pp. 32–39.

[73] E. Breck et al., "The ml test score: A rubric for ml production readiness and technical debt reduction", in *IEEE International Conference on Big Data (Big Data)*, 2017, pp. 1123–1132.

[74] J. Thomas, "Systems theoretic process-analysis STPA", *Availble online at: http://psas. scripts. mit. edu/home/wp-content/uploads/2016/01*, 2016.

[75] J. Berger, *STPA guide*. VTT Technical Research Centre of Finland, 2024.

[76] B. Braunschweig et al., "AITA: AI trustworthiness assessment: AAAI spring symposium 2023", *AI and Ethics*, vol. 4, pp. 1–3, 2024.

[77] K. Kapusta et al., "Protecting ownership rights of ml models using watermarking in the light of adversarial attacks", *AI and Ethics*, vol. 4 - 1, pp. 95–103, 2024.

[78] M. Lansari et al., "A Black-Box Watermarking Modulation for Object Detection Models", in *Proceedings of the AAAI Symposium Series*, vol. 4, 2024, pp. 60–67.

[79] J. Mattioli et al., "An overview of key trustworthiness attributes and KPIs for trusted ML-based systems engineering", *AI and Ethics*, vol. 4 - 1, pp. 15–25, 2024.

[80] A. Dakwat and E. Villani, "System safety assessment based on STPA and model checking", *Safety science*, vol. 109, pp. 130–143, 2018.

[81] A. Awadid and B. Robert, "On assessing ML model robustness: A methodological framework", in *Symposium on Scaling AI Assessments*, 2025.

[82] J. Mattioli et al., "Towards a holistic approach for AI trustworthiness assessment based upon aids for multi-criteria aggregation", in *SafeAI 2023-The AAAI's Workshop on Artificial Intelligence Safety*, vol. 3381, 2023.

[83] J. Mattioli et al., "A Brief Overview of Key Quality Metrics for Knowledge Graph Solution. Illustration on Digital NOTAMs", in *Proceedings of the AAAI Symposium Series*, vol. 7, 2025, pp. 206–213.

[84] S. Naveed et al., "An overview of the empirical evaluation of explainable AI (XAI): A comprehensive guideline for user-centered evaluation in xAI", *Applied Sciences*, vol. 14, p. 11 288, 2024.

[85] J. Mattioli et al., "Leveraging tropical algebra to assess trustworthy AI", in *Proceedings of the AAAI Fall Symposium Series*, vol. 4, 2024, pp. 81–88.

[86] C. Labreuche, "A general framework for explaining the results of a multi-attribute preference model", *Artificial Intelligence*, vol. 175, pp. 1410–1448, 2011.

[87] M. Ducoffe et al., "Lard–landing approach runway detection–dataset for vision based landing", *arXiv preprint arXiv:2304.09938*, 2023.

2025, © Copyright by authors, Published under agreement with IARIA - www.iaria.org