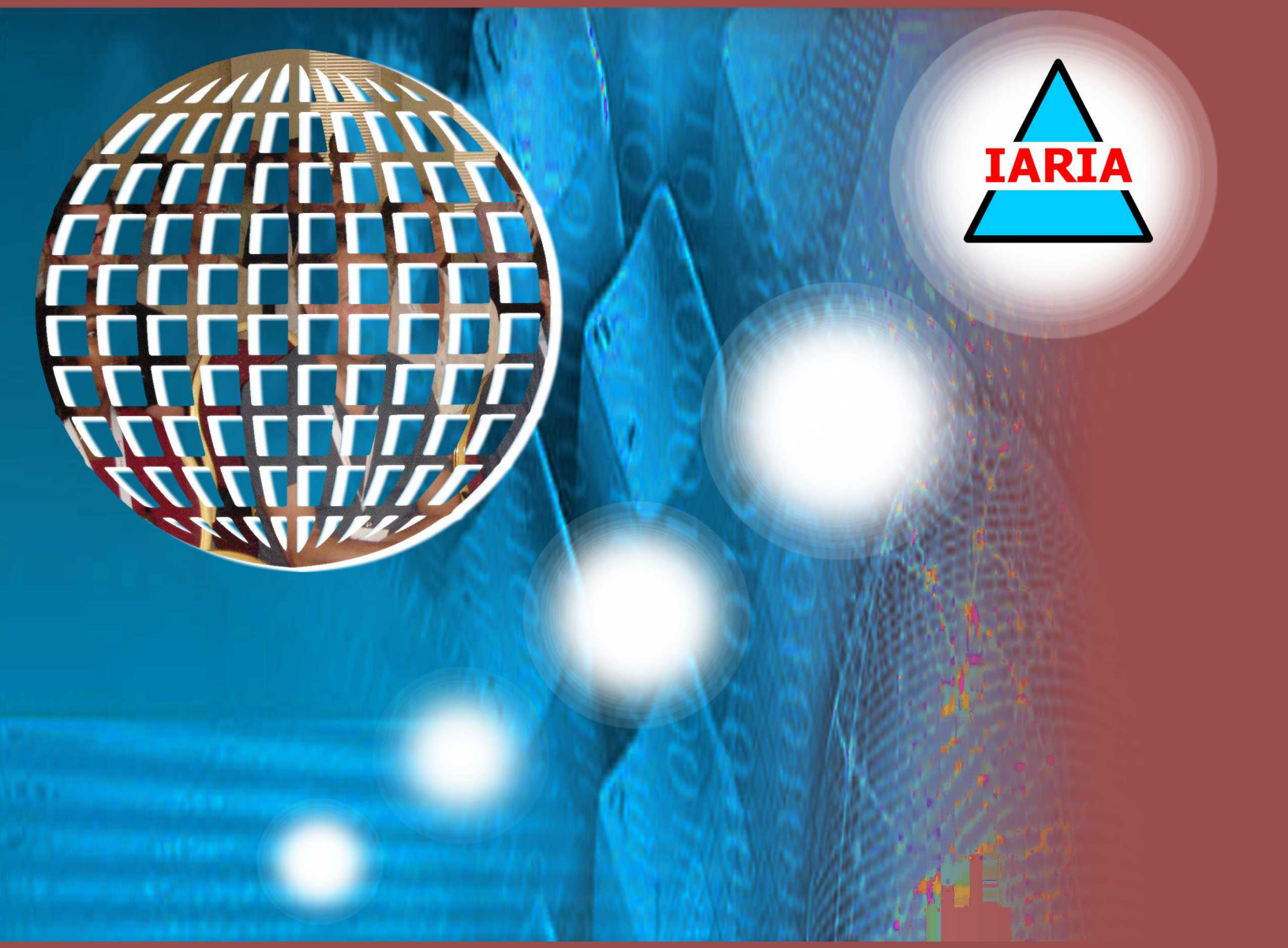
International Journal on Advances in Security



2019 vol. 12 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

Sponsored by IARIA
www.iaria.org

International Journal on Advances in Security
Volume 12, Number 3 & 4, 2019

Zhixiong Chen, Mercy College, USA
Clelia Colombo Vilarrasa, Autonomous University of Barcelona, Spain
Peter Cruickshank, Edinburgh Napier University Edinburgh, UK
Nora Cuppens, Institut Telecom / Telecom Bretagne, France
Glenn S. Dardick, Longwood University, USA
Vincenzo De Florio, University of Antwerp & IBBT, Belgium
Paul De Hert, Vrije Universiteit Brussels (LSTS) - Tilburg University (TILT), Belgium
Pierre de Leusse, AGH-UST, Poland
William Dougherty, Secern Consulting - Charlotte, USA
Raimund K. Ege, Northern Illinois University, USA
Laila El Aimani, Technicolor, Security & Content Protection Labs., Germany
El-Sayed M. El-Alfy, King Fahd University of Petroleum and Minerals, Saudi Arabia
Rainer Falk, Siemens AG - Corporate Technology, Germany
Shao-Ming Fei, Capital Normal University, Beijing, China
Eduardo B. Fernandez, Florida Atlantic University, USA
Anders Fongen, Norwegian Defense Research Establishment, Norway
Somchart Fugkeaw, Thai Digital ID Co., Ltd., Thailand
Steven Furnell, University of Plymouth, UK
Clemente Galdi, Universita' di Napoli "Federico II", Italy
Birgit Gersbeck-Schierholz, Leibniz Universität Hannover, Germany
Manuel Gil Pérez, University of Murcia, Spain
Karl M. Goeschka, Vienna University of Technology, Austria
Stefanos Gritzalis, University of the Aegean, Greece
Michael Grottke, University of Erlangen-Nuremberg, Germany
Ehud Gudes, Ben-Gurion University - Beer-Sheva, Israel
Indira R. Guzman, Trident University International, USA
Huong Ha, University of Newcastle, Singapore
Petr Hanáček, Brno University of Technology, Czech Republic
Gerhard Hancke, Royal Holloway / University of London, UK
Sami Harari, Institut des Sciences de l'Ingénieur de Toulon et du Var / Université du Sud Toulon Var, France
Daniel Harkins , Hewlett Packard Enterprise, USA
Ragib Hasan, University of Alabama at Birmingham, USA
Masahito Hayashi, Nagoya University, Japan
Michael Hobbs, Deakin University, Australia
Hans-Joachim Hof, Munich University of Applied Sciences, Germany
Neminath Hubballi, Infosys Labs Bangalore, India
Mariusz Jakubowski, Microsoft Research, USA
Ravi Jhawar, Università degli Studi di Milano, Italy
Dan Jiang, Philips Research Asia Shanghai, China
Georgios Kambourakis, University of the Aegean, Greece
Florian Kammueller, Middlesex University - London, UK
Sokratis K. Katsikas, University of Piraeus, Greece
Seah Boon Keong, MIMOS Berhad, Malaysia
Sylvia Kierkegaard, IAITL-International Association of IT Lawyers, Denmark
Hyunsung Kim, Kyungil University, Korea
Geir M. Køien, University of Agder, Norway
Ah-Lian Kor, Leeds Metropolitan University, UK
Evangelos Kranakis, Carleton University - Ottawa, Canada
Lam-for Kwok, City University of Hong Kong, Hong Kong
Jean-Francois Lalande, ENSI de Bourges, France
Gyungho Lee, Korea University, South Korea
Clement Leung, Hong Kong Baptist University, Kowloon, Hong Kong
Diego Liberati, Italian National Research Council, Italy

Giovanni Livraga, Università degli Studi di Milano, Italy
Gui Lu Long, Tsinghua University, China
Jia-Ning Luo, Ming Chuan University, Taiwan
Thomas Margoni, University of Western Ontario, Canada
Rivalino Matias Jr ., Federal University of Uberlandia, Brazil
Manuel Mazzara, UNU-IIST, Macau / Newcastle University, UK
Catherine Meadows, Naval Research Laboratory - Washington DC, USA
Carla Merkle Westphall, Federal University of Santa Catarina (UFSC), Brazil
Ajaz H. Mir, National Institute of Technology, Srinagar, India
Jose Manuel Moya, Technical University of Madrid, Spain
Leonardo Mostarda, Middlesex University, UK
Jogesh K. Muppala, The Hong Kong University of Science and Technology, Hong Kong
Syed Naqvi, CETIC (Centre d'Excellence en Technologies de l'Information et de la Communication),Belgium
Sarmistha Neogy, Jadavpur University, India
Mats Neovius, Åbo Akademi University, Finland
Jason R.C. Nurse, University of Oxford, UK
Peter Parycek, Donau-Universität Krems, Austria
Konstantinos Patsakis, Rovira i Virgili University, Spain
João Paulo Barraca, University of Aveiro, Portugal
Sergio Pozo Hidalgo, University of Seville, Spain
Yong Man Ro, KAIST (Korea advanced Institute of Science and Technology), Korea
Rodrigo Roman Castro, University of Malaga, Spain
Heiko Roßnagel, Fraunhofer Institute for Industrial Engineering IAO, Germany
Claus-Peter Rückemann, Leibniz Universität Hannover / Westfälische Wilhelms-Universität Münster / North-German Supercomputing Alliance, Germany
Antonio Ruiz Martinez, University of Murcia, Spain
Paul Sant, University of Bedfordshire, UK
Peter Schartner, University of Klagenfurt, Austria
Alireza Shameli Sendi, Ecole Polytechnique de Montreal, Canada
Dimitrios Serpanos, Univ. of Patras and ISI/RC ATHENA, Greece
Pedro Sousa, University of Minho, Portugal
George Spanoudakis, City University London, UK
Vladimir Stantchev, Institute of Information Systems, SRH University Berlin, Germany
Lars Strand, Nofas, Norway
Young-Joo Suh, Pohang University of Science and Technology (POSTECH), Korea
Jani Suomalainen, VTT Technical Research Centre of Finland, Finland
Enrico Thomae, Ruhr-University Bochum, Germany
Tony Thomas, Indian Institute of Information Technology and Management - Kerala, India
Panagiotis Trimintzios, ENISA, EU
Peter Tröger, Hasso Plattner Institute, University of Potsdam, Germany
Simon Tsang, Applied Communication Sciences, USA
Marco Vallini, Politecnico di Torino, Italy
Bruno Vavala, Carnegie Mellon University, USA
Mthulisi Velempini, North-West University, South Africa
Miroslav Velev, Aries Design Automation, USA
Salvador E. Venegas-Andraca, Tecnológico de Monterrey / Texia, SA de CV, Mexico
Szu-Chi Wang, National Cheng Kung University, Tainan City, Taiwan R.O.C.
Steffen Wendzel, Fraunhofer FKIE, Bonn, Germany
Piyi Yang, University of Shanghai for Science and Technology, P. R. China
Rong Yang, Western Kentucky University , USA
Hee Yong Youn, Sungkyunkwan University, Korea
Bruno Bogaz Zarpelao, State University of Londrina (UEL), Brazil
Wenbing Zhao, Cleveland State University, USA

International Journal on Advances in Security
Volume 12, Numbers 1 & 2, 2019

CONTENTS

pages: 153 - 163
Verified Metrics for Continuous Active Defence
George O. M. Yee, Aptusinnova Inc. and Carleton University, Canada

pages: 164 - 176
A Guideline on the Analysis of Stochastic Interdependencies in Critical Infrastructures
Sandra König, Austrian Institute of Technology, Austria
Thomas Grafenauer, Austrian Institute of Technology, Austria
Stefan Rass, Universität Klagenfurt, Austria
Stefan Schauer, Austrian Institute of Technology, Austria
Manuel Warum, Austrian Institute of Technology, Austria

pages: 177 - 193
An Advanced Approach for Choosing Security Patterns and Checking their Implementation
Sébastien Salva, University Clermont Auvergne, LIMOS laboratory, France
Loukmen Regainia, University Clermont Auvergne, LIMOS laboratory, France

pages: 194 - 202
Network Analysis of City Streets: Forecasting Burglary Risk in Small Areas
Maria Mahfoud, CWI National Research Institute for Mathematics and Computer Science, The Netherlands
Sandjai Bhulai, Vrije Universiteit Amsterdam, The Netherlands
Rob van der Mei, CWI National Research Institute for Mathematics and Computer Science, The Netherlands
Dimitry Erkin, CWI National Research Institute for Mathematics and Computer Science, The Netherlands
Elenna Dugundji, Vrije Universiteit Amsterdam, The Netherlands

pages: 203 - 222
Deploying Artificial Intelligence to Combat Disinformation Warfare: Identifying and Interdicting Disinformation Attacks Against Cloud-based Social Media Platforms
Barry Cartwright, Simon Fraser University, Canada
George Weir, University of Strathclyde, Scotland, UK
Richard Frank, Simon Fraser University, Canada
Karmvir Padda, Simon Fraser University, Canada

pages: 223 - 235
Reaching Grey Havens: Industrial Automotive Security Modeling with SAM
Markus Zoppelt, Nuremberg Institute of Technology, Germany
Ramin Tavakoli Kolagari, Nuremberg Institute of Technology, Germany

pages: 236 - 247
Enabling Financial Reports Transparency and Trustworthiness using Blockchain Technology
Van Thanh Le, Free University of Bolzano, Italy
Claus Pahl, Free University of Bolzano, Italy
Nabil El Ioini, Free University of Bolzano, Italy
Gianfranco D'Atri, Calabria University, Italy

Verified Metrics for Continuous Active Defence

George O. M. Yee
Computer Research Lab, Aptusinnova Inc., Ottawa, Canada
Dept. of Systems and Computer Engineering, Carleton University, Ottawa, Canada
email: george@aptusinnova.com, gmyee@sce.carleton.ca

Abstract—**As a sign of the times, headlines today are full of attacks against an organization's computing infrastructure, resulting in the theft of sensitive data. In response, the organization applies security measures (e.g., encryption) to secure its vulnerabilities. However, these measures are often only applied once, with the assumption that the organization is then protected and no further action is needed. Unfortunately, attackers continuously probe for vulnerabilities and change their attacks accordingly. This means that an organization must also continuously check for new vulnerabilities and secure them, to continuously and actively defend against the attacks. This paper derives metrics that characterize the security level of an organization at any point in time, based on the number of vulnerabilities secured and the effectiveness of the securing measures. The metrics are verified in terms of their soundness using the author's recently published procedure for deriving good security metrics. The paper then shows how an organization can apply the metrics for continuous active defence.**

Keywords- sensitive data; vulnerability; security level; verified metrics; continuous defence.

I. INTRODUCTION

This work extends Yee [1] by adding explanations and related work, elaborating the application areas, and including a new section on verifying the soundness of the proposed metrics.

Headlines today are full of news of attacks against computing infrastructure, resulting in sensitive data being compromised. These attacks have devastated the victim organizations. The losses have not only been financial (e.g., theft of credit card information), but perhaps more importantly, have damaged the organizations' reputation. The first half of 2019 had 3,800 publicly disclosed breaches with 4.1 billion records exposed, an increase of 54% in the number of reported breaches when compared to the first half of 2018 [2]. Here are just two of those breaches [2]:

- March 22 and 23, 2019, Capital One: The number of records breached was 106 million, including names, addresses, postal codes, phone numbers, email addresses, birthdates, and self-reported income. Also exposed in some cases were customer credit scores, credit limits, balances, and payment history. The breach affected about 100 million consumers in the United States and about 6 million in Canada. A hacker accessed the servers of a third-party cloud services company contracted by Capital One. The hacker hacked the servers on March 22 and 23, 2019 and has since been arrested. According to CNN Business, Capital One expected losses of $100 million to $150 million related to the hack, for expenses incurred in notifying affected customers, providing free credit monitoring, legal defense, and fixing the vulnerability.
- August 1, 2018 to March 30, 2019, American Medical Collection Agency: Here the number of records breached was over 20 million, including social security numbers, birthdates, payment card data, credit card information, and bank account information. American Medical Collection Agency collected overdue payments for medical labs. This long running breach exposed the records of the labs' customers including the above sensitive data. A cybersecurity firm found the breached information on the dark web. American Medical Collection Agency filed for bankruptcy in June 2019, citing IT costs, possible lawsuits, and the loss of business from its customers.

Hard hit data breach victims in 2018 [3] include toymaker Vtech Technologies (a cyberattack exposed the personal data of an estimated 6.4 million children worldwide), Under Armour (a cyberattack stole the personal data of 150 million users of its app), and major airlines such as Air Canada, British Airways, and Cathay Pacific (hackers made off with the personal data of a combined 9.8 million customers). The year 2017 [4] saw a total of 5,207 breaches and 7.89 billion information records compromised.

In response to attacks, such as the ones described above, organizations determine their computer system vulnerabilities and secure them using security measures. Typical measures include firewalls, intrusion detection systems, two-factor authentication, encryption, and training for employees on identifying and resisting social engineering. However, once the security measures have been implemented, organizations tend to believe that they are safe and that no further actions are needed. Unfortunately, attackers do not give up just because the organization has secured its known computer vulnerabilities. Rather, the attackers will continuously probe the organization's computer system for new vulnerabilities that they can exploit. This means that the organization must continuously analyze its computer system vulnerabilities and secure any new ones that it discovers. In order to do this effectively, it is useful to

have quantitative metrics of the security level at any particular point in time, based on the number of vulnerabilities secured and the effectiveness of the security measures, at that point in time. An acceptable security level can be set, so that if the security level falls below this acceptable level due to new vulnerabilities, the latter can be secured to bring the security level back to the acceptable level. This work derives such metrics and shows how to apply them for continuous active defence, i.e., continuous vulnerabilities evaluation and follow up. Further, this work verifies that the proposed metrics are sound, a term that will be defined below.

The objectives of this work are: i) derive straightforward, clear metrics of the resultant protection level obtained by an organization at any point in time, based on the use of security measures to secure vulnerabilities and based on the effectiveness of the measures, ii) show how these metrics can be calculated, iii) verify that these metrics are sound, and iv) show how the metrics can be applied for continuous active defence and discuss some application areas. We seek straightforward, easy to understand metrics since complicated, difficult to understand ones tend not to be used or tend to be misapplied. We base these metrics on securing vulnerabilities since this has been and continues to be the method organizations use to secure their computer infrastructure.

The rest of this paper is organized as follows. Section II discusses sensitive data, attacks, and vulnerabilities. Section III derives the metrics, shows how to calculate them, and presents various aspects of the metrics, including some of their strengths, weaknesses, and limitations. Section IV verifies that the metrics are sound. Section V explains how to apply the metrics for continuous active defence and presents some application areas. Section VI discusses related work. Section VII gives conclusions and future work.

II. Sensitive Data, Attacks, and Vulnerabilities

Sensitive data is data that needs protection and must not fall into the wrong hands. It includes private or personal information [5], which is information about an individual, can identify that individual, and is owned by that individual. For example, an individual's height, weight, or credit card number can all be used to identify the individual and are considered as personal information or personal sensitive data. Sensitive data also includes non-personal information that may compromise the competitiveness of the organization if divulged, such as trade secrets or proprietary algorithms and formulas. For government organizations, non-personal sensitive data may include information that is vital for the security of the country for which the government organization is responsible.

DEFINITION 1: *Sensitive data* (SD) is information that must be protected from unauthorized access in order to safeguard the privacy of an individual, the well-being or expected operation of an organization, or the well-being or expected functioning of an entity for which the organization has responsibility.

DEFINITION 2: An *attack* is any action carried out against an organization's computer system that, if successful, compromises the system or the SD held by the system.

An attack that compromises a computer system is Distributed Denial of Service (DDoS). One that compromises the SD held by the system is a Trojan horse attack in which malicious software (the Trojan) is planted inside the system to steal SD. Attacks can come from an organization's employees, in which case the attack is an *inside attack*. For example, a disgruntled employee secretly keeps a copy of a SD backup and sells it on the "dark web".

DEFINITION 3: A *vulnerability* of a computer system is any weakness in the system that can be targeted by an attack with some expectation of success. A vulnerability can be secured to become a *secured vulnerability* through the application of a security measure.

An example of a vulnerability is a communication channel that is used to convey sensitive data in the clear. This vulnerability can be targeted by a Man-in-the-Middle attack with reasonable success of stealing the sensitive data. This vulnerability can become a secured vulnerability by encrypting the sensitive data that the communication channel carries.

A computer system can undergo upgrades, downgrades, and other modifications over time that changes its number of secured and unsecured vulnerabilities. It is thus necessary to specify a time *t* when referring to vulnerabilities. Clearly, the number of secured and unsecured vulnerabilities of a computer system at time *t* is directly related to the security level of the system at time *t*. This idea is formalized in the next definition.

DEFINITION 4: A computer system's security level (SL) at time t, or SL(t), is the degree of protection from attacks that results from having q(t) secured vulnerabilities, and p(t) unsecured vulnerabilities, where the system has a total of N(t) = p(t)+q(t) secured and unsecured vulnerabilities. SL(t) is uniquely represented by the pair (p(t), q(t)).

Clearly SL(t) increases with increasing q(t) and decreases with increasing p(t). Figure 1 shows 3 SL(t) points on the (p(t), q(t)) plane for N(t)=100.

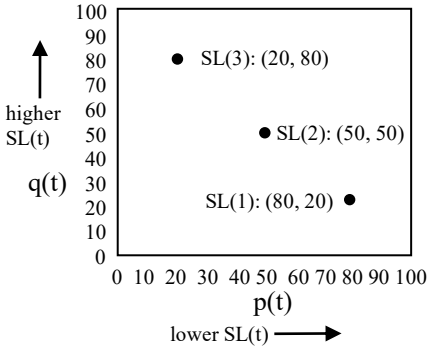


Figure 1. SL(t) points corresponding to a computer system with N(t)=100. SL(3) is higher security than SL(2), which is higher security than SL(1).

In Figure 1, the higher values of q(t) correspond to higher security levels, and the higher values of p(t) correspond to lower security levels.

Definition 4 requires p(t) to be known. Of course, it is next to impossible to determine all the vulnerabilities in a typical computer system, so the exact value of p(t) is most likely undeterminable. Thus, a value for p(t) can only be a "best effort" value, and consequently, a value for SL(t) is not the true value, but a "best effort" value. It is in this context that the values of p(t) and SL(t) are to be understood.

III. Metrics for Continuous Active Defence

While the pair (p(t), q(t)) uniquely represents SL(t), it cannot be used to calculate the value of SL(t), which would be useful in tracking the security of a system over time as its vulnerabilities change. In this section, we derive two metrics for the value of SL(t), one assuming that the measures securing vulnerabilities are totally reliable; the other with the measures only partly reliable. Both metrics are applied right after the vulnerabilities have been determined, and possibly before any of them have actually been secured. Determining vulnerabilities is discussed in Section III.C below.

A. Metric with Totally Reliable Securing Measures

We seek a metric STRM(t) (STRM is an acronym for "SL with Totally Reliable Measures") for a computer system's SL(t), where all securing measures are totally reliable. Suppose that p(t) and q(t) are as in Definition 4. Let $P_t(e)$ represent the probability of event e at time t. Let "exploit" mean a successful attack on a vulnerability. Let "all exploits" mean exploits on 1 or more vulnerabilities. Let $U_k(t)$ denote an unsecured vulnerability k at time t. We have

$$SL(t) = P_t(\text{no exploits}) = 1\text{-}P_t(\text{all exploits}) \quad (1)$$

However, the only exploitable vulnerabilities are the unsecured vulnerabilities since the securing measures are totally reliable. Therefore

$$P_t(\text{all exploits}) = \Sigma_k\, [P_t(\text{exploit of } U_k(t))]$$

by applying the additive rule for the union of probabilities, assuming that 2 or more exploits do not occur simultaneously. Let $u_k(t)$ be a real number with $0 < u_k(t) \leq p(t)$ and $\Sigma_k u_k(t) = p(t)$. Set

$$P_t(\text{exploit of } U_k(t)) \approx u_k(t)/(p(t)+q(t)) \quad (2)$$

By substitution using (2)

$$\begin{aligned} P_t(\text{all exploits}) &\approx \Sigma_k\, [u_k(t)/(p(t)+q(t))] \\ &= \Sigma_k u_k(t)/(p(t)+q(t)) \\ &= p(t)/(p(t)+q(t)) \quad (3) \end{aligned}$$

The condition $0 < u_k(t) \leq p(t)$ is needed to ensure that there is some probability for an unsecured vulnerability to be exploited. The condition $\Sigma_k u_k(t) = p(t)$ is necessary in order for $P_t(\text{all exploits}) \leq 1$. Expression (2) gives a way of assigning values for $P_t(\text{exploit of } U_k(t))$ based on a risk analysis [5]. However, expression (3) ensures that such assignment is not needed for calculating STRM(t). In other words, the fact that some vulnerabilities are more likely to be exploited than others does not affect the value of STRM(t).

Substituting (3) into (1) gives

$$\begin{aligned} SL(t) &\approx 1\text{-}[p(t)/(p(t)+q(t))] \\ &= q(t)/(p(t)+q(t)) \quad \text{if } p(t)+q(t) > 0 \\ &= 1 \quad \text{if } p(t)+q(t) = 0 \end{aligned}$$

(Note that mathematically, we cannot divide by 0.) We obtain STRM(t) by assigning as follows:

$$\begin{aligned} \mathbf{STRM(t)} &= \mathbf{q(t)/(p(t)+q(t))} \quad \textbf{if } \mathbf{p(t)+q(t) > 0} \quad (4) \\ &= \mathbf{1} \quad \textbf{if } \mathbf{p(t)+q(t) = 0} \quad (5) \end{aligned}$$

We see from (4) that $0 \leq STRM(t) \leq 1$ if $p(t)+q(t) > 0$ and has value 0 if q(t)=0 (the system has no secured vulnerabilities) and 1 if p(t)=0 (all of its vulnerabilities are secured). We see from (5) that STRM(t)=1 if p(t)+q(t)=0 (no vulnerabilities, which is unlikely). The values of the metric are therefore as expected.

B. Metric with Partially Reliable Securing Measures

Here, we seek a metric SPRM(t) (SPRM is an acronym for "SL with Partially Reliable Measures") for a computer system's SL(t) where the measures securing the vulnerabilities are only partially reliable.

Let $V_k(t)$ denote a secured vulnerability k at time t. The reliability $r_k(t)$ of the measure securing $V_k(t)$ can be defined as the probability that the measure remains operating from time zero to time t, given that it was operating at time zero [6]. The unreliability of the measure is then $1\text{-}r_k(t)$. We have the events

[exploit of $V_k(t)$] if and only if [$V_k(t)$ selected for exploit]
AND [measure securing $V_k(t)$ unreliable]

Since the two right-hand side events are independent,

$P_t(\text{exploit of } V_k(t)) = P_t(V_k(t) \text{ selected for exploit}) \times P_t(\text{measure securing } V_k(t) \text{ unreliable})$

Set $P_t(V_k(t) \text{ selected for exploit}) \approx 1/(p(t)+q(t))$ (6)

since attackers will have no preference to attack one secured vulnerability over another secured vulnerability (they should not even see them as vulnerabilities). Again, applying the additive rule for the union of probabilities,

$$\begin{aligned} P_t(\text{all } V_k(t) \text{ exploits}) &= \Sigma_k[P_t(V_k(t) \text{ selected for exploit}) \times P_t(\text{measure securing } V_k(t) \text{ unreliable})] \\ &= \Sigma_k\, [(1/(p(t)+q(t)))(1-r_k(t))] \\ &= [\Sigma_k(1-r_k(t)]/[p(t) + q(t)] \\ &= [q(t)-\Sigma_k r_k(t)]/[p(t) + q(t)] \\ &= [q(t)/(p(t)+q(t))]-\Sigma_k r_k(t)/(p(t) + q(t)) \quad (7) \end{aligned}$$

Now, since both $U_k(t)$ and $V_k(t)$ can be exploited,

$$\begin{aligned} P_t(\text{all exploits}) &= P_t(\text{all } U_k(t) \text{ exploits}) + P_t(\text{all } V_k(t) \text{ exploits}) \\ &\approx [p(t)/(p(t)+q(t))] + [q(t)/(p(t)+q(t))]- \Sigma_k r_k(t)/(p(t) + q(t)) \\ &= 1 - \Sigma_k r_k(t)/(p(t) + q(t)) \quad (8) \end{aligned}$$

by substitution using (3) and (7), where (3) is $P_t(\text{all } U_k(t)$

exploits). Finally, by substitution using (1) and (8),

$$\begin{aligned} SL(t) &\approx 1 - 1 + \Sigma_k r_k(t)/(p(t) + q(t)) \\ &= \Sigma_k r_k(t)/(p(t) + q(t)) \quad \text{if } p(t) \geq 0, q(t) > 0 \\ &= 1 \quad \text{if } p(t)+q(t) = 0 \\ &= 0 \quad \text{if } p(t)>0, q(t) = 0 \end{aligned}$$

We obtain SPRM(t) by assigning as follows:

$$\mathbf{SPRM(t) = \Sigma_k r_k(t)/(p(t)+q(t)) \quad if\ p(t) \geq 0, q(t) > 0} \quad (9)$$

$$\mathbf{= 1 \quad if\ p(t)+q(t) = 0} \quad (10)$$

$$\mathbf{= 0 \quad if\ p(t)>0, q(t)=0} \quad (11)$$

We see from (9) that $0 < SPRM(t) < 1$ for $p(t) \geq 0$, $q(t) > 0$ (all vulnerabilities may or may not be secured), and from (10) that $SPRM(t) = 1$ for $p(t)+q(t) = 0$ (no vulnerabilities, which is unlikely). We see from (11) that $SPRM(t) = 0$ for $p(t)>0$, $q(t) = 0$ (no secured vulnerabilities). We also see that for $r_k(t) = 1$, SPRM(t) is the same as STRM(t). The values of the metric are therefore as expected.

C. Calculating the Metrics

Calculating STRM(t) requires the values of p(t) and q(t) at a series of time points of interest. SPRM(t) requires the values of p(t), q(t), and the reliability value for each measure used to secure the vulnerabilities.

To obtain the values of p(t) and q(t), an organization may perform a threat analysis of vulnerabilities in the organization's computer system that could allow attacks to occur. Threat analysis or threat modeling is a method for systematically assessing and documenting the security risks associated with a system (Salter et al. [7]). Threat modeling involves understanding the adversary's goals in attacking the system based on the system's assets of interest. It is predicated on that fact that an adversary cannot attack a system without a way of supplying it with data or otherwise accessing it. In addition, an adversary will only attack a system if it has some assets of interest. The method of threat analysis given in [7] or any other method of threat analysis will yield the total number N(t) of vulnerabilities to attacks at time t. Once this number is known, the organization can select which vulnerabilities to secure and which security measures to use, based on a prioritization of the vulnerabilities and the amount of budget it has to spend. A way to optimally select which vulnerabilities to secure is described in [8]. Once vulnerabilities have been selected to be secured, we have q(t). Then $p(t) = N(t) - q(t)$. The threat analysis may be carried out by a project team consisting of the system's design manager, a security and privacy analyst, and a project leader acting as facilitator. In addition to having security expertise, the analyst must also be very familiar with the organization's computer system. Further discussion on threat analysis is outside the scope of this paper. More details on threat modeling can be found in [8]. Vulnerabilities may be prioritized using the method in [5], which describes prioritizing privacy risks.

The reliability values for hardware measures used to secure the selected vulnerabilities may be obtained from the hardware's manufacturers (e.g., hardware firewall). Reliability values for software and algorithmic measures are more difficult to obtain (e.g., encryption algorithm). For these, it may be necessary to estimate the reliability values based on the rate of progress of technology. For example, one could estimate the reliability of an encryption algorithm based on estimates of the computer resources that attackers have at their disposal. If they have access to a super computer, an older encryption algorithm may not be sufficiently reliable. One could also opt to be pessimistic and assign low reliability values, which would have the net effect of boosting security by securing more vulnerabilities, in order to meet a certain SL(t) level (see Section V). Reliability values for security measures represent a topic for future research.

It is important to note that at each time point where the metrics are calculated, the values of p(t) and q(t) are generated anew. Vulnerabilities secured previously with totally reliable measures would not appear again as vulnerabilities. On the other hand, vulnerabilities secured with only partially reliable measures should be identified again as vulnerabilities. Further, it is not necessary to have actually implemented the securing measures before calculating the metrics.

D. Graphing the Metrics

The metrics STRM(t) and SPRM(t) are both functions of p(t), q(t), and t. Figure 2 shows a 3-dimensional graph of these metrics with axes for STRM(t)/SPRM(t), p(t), and q(t). Time is not shown explicitly as an axis since we would need 4 dimensions, but is instead represented as time period displacements of the metrics' values.

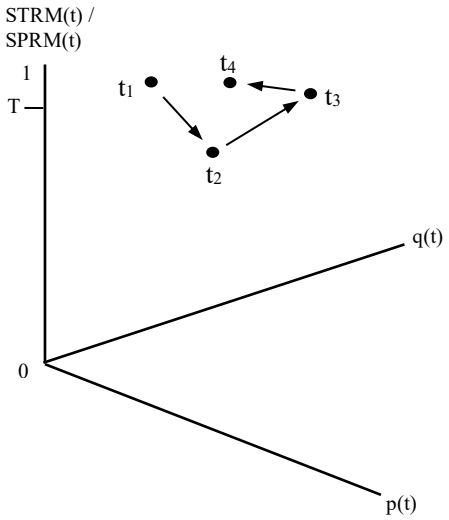


Figure 2. STRM(t)/SPRM(t) values at times $t_1 < t_2 < t_3 < t_4$.

Figure 2 shows 4 values of one of the metrics, labeled according to the times it was evaluated, namely t_1, t_2, t_3, and t_4 where $t_1 < t_2 < t_3 < t_4$. The intervals between these times may be 1 week or 1 month, for example. T is a threshold, below which the metric values should not drop (see Section V.A). At t_1, one of the metrics was evaluated producing the value

shown. At t_2, the metric was again evaluated, but this time the value was found to be much lower than at t_1, and in fact, the value dropped below T. The reason for this was that new vulnerabilities were found that had not been secured. The organization decides to secure the additional vulnerabilities. At t_3, another evaluation was carried out, and this time, the metric had improved, reaching above T. The organization finds some surplus money in its budget and decides to secure 2 other vulnerabilities. An evaluation of the metric at t_4 finds the value a little higher than at t_3, due to the 2 additional vulnerabilities secured. It is thus seen that the security level of a computer system changes over time, in accordance with the system's number of secured and unsecured vulnerabilities.

E. Strengths, Weaknesses, and Limitations

Some strengths of the metrics are: a) conceptually straightforward, and easily explainable to management, and b) flexible and powerful, i.e., they have many application areas, as described in Section V.

Some weaknesses are: a) threat modeling to determine the vulnerabilities is time consuming and subjective, and b) the SL will involve more factors than vulnerabilities and secured vulnerabilities. Moreover, as mentioned above, it is next to impossible to find all the p(t), so the SL determined by the metrics can never be the true SL. For weakness a), it may be possible to automate or semi-automate the threat modeling. Related works [18] and [28] are good starting points for further research. For weakness b), it may be argued that the metrics as presented are sufficient for their envisaged application when other sources of error are considered (e.g., it is difficult to tell where an attacker will strike or how he will strike), and that adding more factors would only make the metrics unnecessarily more cumbersome and time consuming to evaluate with little additional benefit. It is next to impossible to determine the true SL anyway.

Some mathematical limitations of the metrics follow. First of all, the metrics are only estimates of the security level, not the security level itself (and can never be the true SL due to unknowable p(t) as mentioned above). This was indicated in assigning the probabilities as approximate in expressions (2) and (6) above. Second, as noted in Section III.A, it makes no difference to the values of the metrics whether one unsecured vulnerability is more likely to be exploited than another. This means that the metrics are insensitive to one exploited vulnerability causing more damage than others, and may be due to the fact that the metrics are estimating the total security of the computer system, and therefore the total number of exploitable vulnerabilities is what's important, not whether a particularly damaging vulnerability is exploited. Third, we applied the additive rule for the union of probabilities above, requiring that 2 or more exploits do not occur simultaneously. This condition holds in general but if it is violated, the metrics will be inaccurate. This may not be very significant, since they are only estimates. An additional limitation may be that a secured vulnerability may not in reality be secured because the attacker has a secret way of defeating the securing measure. However, this additional limitation is true of other security methods as well.

IV. Verification of Soundness

This section examines the soundness (as defined below) of the proposed metrics using a procedure in this author's previously published paper [9]. In that paper, this author pointed out some flaws that can unintentionally be included in the definition of security metrics, leading to invalid conclusions. The flaws can be found in a number of existing security metrics that were presented in [9]. This author then proposed a procedure that can be used to design "good" or sound security metrics that would be free of the flaws. As it turns out, the procedure can also be used to check existing security metrics to verify that they are sound.

Consider the metric *number of viruses detected and eliminated at a firewall*. The purpose of this metric is to assess the effectiveness of a firewall at filtering out viruses, which impacts the organization's level of security. Unfortunately, this metric says nothing about the viruses that were not detected and got through. If 50 viruses were detected and eliminated but 100 got through, basing the firewall's effectiveness solely on the 50 viruses that were detected and not on the 100 that got through would falsely inflate the firewall's effectiveness and the level of security. Thus, this metric fails its purpose. Another often-used security metric is *time spent on a security-related task*, such as software patching or security incident investigation. The purpose of this metric is to gauge the level of security, assuming that more time spent means higher security. This metric may be useful for project management, to make sure that there is sufficient time to complete the project, but it is practically useless as an indicator of security. The assumption is wrong: more time spent does not necessarily mean better security. For example, the extra time may have been due to inefficient procedures or work processes. Thus, this metric also fails its purpose. To avoid problematic metrics such as the foregoing, this author proposed the following procedure [9] for designing good or sound security metrics.

A. Steps for Designing Sound Security Metrics (SDSSM)

1. **Definition:** Define the quantity to be measured, i.e. the candidate metric. Check that this quantity is meaningful, objective, and unbiased as a measure of the component or components of the security level of "something", where that "something" could be the organization, the organization's computer system, or even a software product. Check also that this quantity can be obtained with undue hardship or costs. If the quantity passes all these checks, proceed to Step 2. Otherwise, repeat this step to obtain a new quantity. Note that the quantity can only measure a *component* or *components* of the security level since the actual security level has many components, such as the number of unsecured vulnerabilities, security flaws in software, disgruntled employees, and so on. An example quantity is *number of software security patches issued in a month*, which is a component of the security level of the software.
2. **Sufficiency:** Verify that the quantity is a sufficient measure of the component or components of the security level (as in necessary and sufficient conditions for

something to be true, see [10]). It is enough to verify sufficiency since there are usually many ways to measure a component, so necessity will not apply in most cases. Verify sufficiency by asking and answering the questions in Table I. For the quantity to be sufficient, the answers to questions 1, 2, and 3 must be "yes", "yes", and "no" respectively. If the quantity is found to be sufficient, proceed to Step 3. Otherwise, repeat from Step 1 to obtain a new quantity. For example, the quantity *time spent on a security-related task* is not a sufficient estimator since spending more time does not mean that the security level will be consistently higher (or lower), as discussed above. Thus, the answer is "no" to question 1. Since this answer must be "yes" for sufficiency, this quantity is not sufficient.

TABLE I. QUESTIONS FOR DETERMINING SUFFICIENCY

No.	Question
1	If the quantity goes up, do you believe that the security level consistently goes up (or down)?
2	Does the quantity have a direct impact on the security level?
3	Are there any aspects missing from the definition of the quantity that are needed for it be effective as a measure of the component or components of the security level?

3. **Divisibility:** Verify if the quantity is divisible into other constituent quantities, or is expressible mathematically in terms of other constituent quantities. If not, proceed to step 4. Otherwise, formulate a mathematical expression that equates the quantity to the constituent quantities, and proceed to Step 4. For example, the quantity *number of software security patches issued in a month* is not further divisible, whereas the quantity *outstanding vulnerabilities after threat analysis each month* may be divided into and equated to *the number of non-secured vulnerabilities from last month* plus *the number of new vulnerabilities found during threat analysis*.
4. **Progression:** Verify that the quantity has the "progression property", that when evaluated over a sufficiently large time period, from past to future, the quantity progresses to an acceptable target level that corresponds to an acceptable or maximal security level. If the quantity has this property, proceed to Step 5. Otherwise, repeat from Step 1 to obtain a new quantity. For example, in the case of *number of software security patches issued in a month,* suppose that this metric is evaluated at the first of the month for the last month. Suppose that the target level for the quantity is zero. Thus, over a sufficiently large number of months in which patches are issued, there are corresponding increases in the security level of the software toward some level. The security level of the software increases with each patch issued until at some point, there is consistently no new patch issued (target zero reached). At this point, the security level of the software is maximal (but not necessarily maximized since there may still be undiscovered security bugs). The quantity has progressed to its target level with corresponding maximal security.
5. **Reproducibility:** Verify that the quantity is reproducible by third-party verifiers. This means that the latter may evaluate the quantity or arrive at its value using the same inputs or procedure and obtain the same result. If the quantity is reproducible, stop. The quantity is now considered a sound security metric. Otherwise, repeat from Step 1 to obtain a new quantity. For example, if the quantity is *number of software security patches issued in a month*, a third-party verifier would add up the software security patches issued for a particular month, and find the same number as the organization that is using the metric. If the quantity is *outstanding vulnerabilities after threat analysis each month*, which we know is equated to *the number of non-secured vulnerabilities from last month* plus *the number of new vulnerabilities found during threat analysis*, the third-party verifier would do the latter addition and verify that the total is the same as obtained by the organization using the metric.

Procedure SDSSM can be used not only to design a sound security metric but also to verify if an existing security metric is sound. This verification is carried out by checking if the metric satisfies each of the steps in SDSSM except for STEP 3, which is not a condition to be checked. STEP 3 is only used when designing a security metric, in order to allow the metric to take on a clearer form. This verification of soundness is captured in the following definition.

DEFINITION 5: A security metric is *sound* if it satisfies every step in SDSSM, excluding STEP 3.

We now apply definition 5 to verify that the metrics proposed in Section III are sound. These metrics are:

$$\mathbf{STRM(t) = q(t)/(p(t)+q(t))} \quad \mathbf{if\ p(t)+q(t) > 0}$$
$$\mathbf{= 1} \quad \mathbf{if\ p(t)+q(t) = 0}$$

$$\mathbf{SPRM(t) = \Sigma_k r_k(t)/(p(t)+q(t))} \quad \mathbf{if\ p(t) \geq 0,\ q(t) > 0}$$
$$\mathbf{= 1} \quad \mathbf{if\ p(t)+q(t) = 0}$$
$$\mathbf{= 0} \quad \mathbf{if\ p(t)>0,\ q(t)=0}$$

It suffices to check that these metrics satisfy the conditions in each step of SDSSM, as follows.

STEP 1: Definition. The security of the computer system is directly related to the number of secured vulnerabilities in the system: the higher this number, the higher the security, and the lower this number, the lower the security. Consequently, since both metrics express the security level in terms of the proportion of secured vulnerabilities to total vulnerabilities, both metrics are clearly meaningful for assessing the security level (note that the numerator in SPRM(t) is really the number of secured vulnerabilities as a fractional or real number). The metrics are objective since secured vulnerabilities relate directly to the security of the system. They are unbiased since their values, based on secured and unsecured vulnerabilities, cannot be overstated or understated. Finally, one can evaluate these metrics without undue hardship or cost by doing a vulnerability or threat analysis, deciding which vulnerabilities to secure, and using the reliabilities of the

securing measures where available. Thus, these metrics are considered to have passed Step 1 and we proceed to Step 2.

STEP 2: Sufficiency. We answer the questions in Table I. The first question asks if the security would consistently go up (or down) if the quantity (metric) goes up. Clearly if the value of STRM(t) goes up, the number of secured vulnerabilities must consistently go up since the denominator is a constant. In other words, the security consistently goes up. The same can be said of SPRM(t), since its numerator is the number of secured vulnerabilities as a real number. So, the answer to the first question is "yes" for both metrics. The second question asks if the quantity has a direct impact on the security level. The answer is again "yes" for both metrics, since the higher their values, the higher the security level, and the lower their values, the lower the security level. Finally, the third question asks if the quantity is missing any components or aspects that are needed for it to be effective. The answer here is "no" for both metrics, since they are ready to be used "as is" for effectively assessing the security level. The answers to the three questions conform to the answers required for sufficiency. We declare the metrics sufficient and proceed to Step 4, since STEP 3 is not needed for verifying soundness.

STEP 4: Progression. Suppose that vulnerabilities are determined (through a threat analysis) and one of the metrics (STRM(t) if no reliability values are available, SPRM(t) otherwise) is re-calculated at regular time intervals, e.g., monthly. Suppose also that Company A's management has agreed on a goal of 95% for the metric, at which level the computer system is considered "safe", i.e. management is willing to live with the risks arising from the remaining non-secured vulnerabilities. With this goal in mind, management will want to secure vulnerabilities at each opportunity until the metric attains 95%. This doesn't mean that the metric will increase monotonically, since it is possible that a particular threat analysis identifies so many new vulnerabilities that the metric is actually lower than when it was last calculated. However, the metric will eventually reach 95%, given that management wants to secure new vulnerabilities until this goal is reached, which is all we mean by having the progression property. Since this analysis applies to both metrics, we can consider them as having passed Step 4 and proceed to Step 5.

STEP 5: Reproducibility. Given the expression for STRM(t), anyone will calculate the same value for it given the same values for p(t) and q(t). Similarly, given the expression for SPRM(t), anyone will calculate the same value for it given the same values for the reliabilities, p(t), and q(t). Thus, the metrics are reproducible.

Thus, according to Definition 5, the metrics STRM(t) and SPRM(t) are sound.

To show that the application of SDSSM can find that a metric is not sound, consider its application to the flawed metric mentioned above, namely the metric *number of viruses detected and eliminated at a firewall.* Applying SDSSM to this metric leads to it failing STEP 1 Definition, since it is biased towards overstating the firewall's effectiveness. Thus, according to Definition 5, this metric is not sound. Note that the metric *time spent on a security-related task* would also be found by SDSSM as not sound since it failed STEP 2 Sufficiency, as indicated in the description of SDSSM above.

V. Application Areas

In this section, we present some applications for the metrics. In Section V.A, we discuss how they can be used for continuous active defence of a computer system. In Section V.B, we present other application areas, such as critical infrastructure and defence.

A. Continuous Active Defence

Attackers do not attack once, and finding that you are well protected, go away. Rather, they continuously probe your defences in order to find new vulnerabilities to exploit. It is thus necessary to continuously evaluate the computer system's vulnerabilities using threat modeling, and add additional security by securing new vulnerabilities when necessary. We call this "Continuous Active Defence" or CAD. How do we know when it is necessary to add more security? This is where the metrics can be applied. Continuous Active Defence involves the following steps:

1. Decide on a threshold for SL(t) below which the values of the metrics should not drop.
2. Decide on the frequency with which to perform threat modeling, e.g., every week, every month, exceptions.
3. Begin Continuous Active Defence by carrying out the threat modeling at the frequency decided above. After each threat modeling exercise, calculate either STRM(t) (if reliability data is not available) or SPRM(t) (if reliability data is available). If the value of the metric falls below T (see Figure 2), secure additional vulnerabilities until the value is above T.
4. If there has been a change to the system, such as new equipment or new software, do an immediate threat analysis, calculate one of the metrics, and add security if necessary based on T. Then, proceed with the frequency for threat modeling decided above.

The value of T and the frequency of threat modeling can be determined by the same threat analysis team mentioned above. The values would depend on the following:

- The potential value of the sensitive data – the more valuable the data is to a thief, a malicious entity, or a competitor, the higher the threshold and frequency should be.
- The damages to the organization that would result, if the sensitive data were compromised – of course, the higher the damages, the higher the threshold and frequency.
- The current and likely future attack climate – consider the volume of attacks and the nature of the victims, say over the last 6 months; if the organization's sector or industry has sustained a large number of recent attacks, then the threshold and frequency need to be higher.
- Consider also potential attacks by nation states as a result of the political climate; attacks by individual hacktivist groups such as Anonymous or WikiLeaks may also warrant attention.

In general, a computer system should be as secure as possible. Therefore, T above 80% and a frequency of weekly would not be uncommon. However, whatever the threshold and frequency, the organization must find them acceptable after considering the above factors. The financial budget available for securing vulnerabilities also plays an important role here, since higher thresholds call for securing more vulnerabilities, which means more financial resources will be needed.

B. Other CAD Application Areas

CAD may also be applied to a specific type of vulnerabilities. An example of this application is dealing with inside attacks. If the organization is particularly susceptible to inside attacks, it can decide to apply CAD to vulnerabilities that can be exploited for inside attacks. In this case, some of the vulnerabilities may be weaknesses of the organization itself, e.g., ineffective screening of job applicants, and the securing measures may not be technological, e.g., having an ombudsman for employee concerns. A list of questions that can be used to identify vulnerabilities to inside attack is given in [8].

CAD may be applied to a specific subset of vulnerabilities that the organization deems are crucial to its mission. For example, a cloud service provider would deem the protection of clients' data crucial to its mission. It can choose to apply CAD to vulnerabilities that are specific to its data storage capabilities, and also apply CAD to its computer system as a whole.

CAD may also be applied to code level vulnerabilities. In this case, the frequency of application will depend on how often the code is changed, due to patching and the addition or deletion of functionality. The threat modeling would have to be tailored to code and would be more of a code inspection exercise.

Finally, CAD may be applied to protect critical infrastructure and defence systems. The power grid is an example of critical infrastructure. The development of the metrics only considers vulnerabilities and reliabilities, which are also found in critical infrastructure and defence systems. However, the threat analyses would involve different types of threats, and the securing measures, would of course, need to be appropriate for the vulnerability. For example, the vulnerability of transformer sabotage in a power grid may need to be secured by the use of intrusion alarms. As another example, the vulnerability of a retaliatory missile site being preemptively destroyed may need to be secured by putting the missile on a mobile platform. The application of CAD to protect these areas is a subject of future research.

C. Where CAD May and May Not Be Applied

Fundamentally, CAD may be applied to organizations and systems that have the following elements:

a) Possess "something" that attackers want
b) Vulnerabilities that change over time and that attackers can attack to access the "something"
c) Measures (or controls) that can be used to secure the vulnerabilities from attack

An examination of the above CAD application areas will find these elements present in each area. Organizations or systems that are missing any of these elements are therefore not suitable for the application of CAD. An example of such a "system" may be an expensive bicycle. In this case, it is the bicycle itself that thieves (attackers) want. Its vulnerability is that it can be stolen if the bicycle is not suitably secured. The measure that can be used to secure the bicycle is a strong lock. However, the bicycle's vulnerability to being stolen is not changing over time. This vulnerability will be the same always, even if the bicycle becomes less attractive to thieves over time. This bicycle is not a suitable system for the application of CAD.

VI. Related Work

Related work found in the literature includes attack surface metrics, risk and vulnerabilities assessment, vulnerabilities classification, threat analysis, an "other" category, and this author's previous work. We discuss each of these categories in turn, starting with attack surface metrics.

A system's attack surface is related to a SL; it is proportional to the inverse of a SL since the lower the attack surface, the higher the SL. Manadhata and Wing [11] formalize the concept of a system's attack surface and propose an attack surface metric for systematically measuring the attack surface. They claim that their metric does not depend on the software system's implementation language and can be used on systems of all sizes. They further provide demonstrations of the metric and have conducted empirical studies to validate it. Stuckman and Purtilo [12] present a framework for formalizing code-level attack surface metrics and describe activities that can be carried out during application deployment to reduce the application's attack surface. They also describe a tool for determining the attack surface of a web application, together with a method for evaluating an attack surface metric over a number of known vulnerabilities. Munaiah and Meneely [13] propose function and file level attack surface metrics that allow fine-grained risk assessment. They claim that their metrics are flexible in terms of granularity, perform better than comparable metrics in the literature, and are tunable to specific products to better assess risk.

In terms of risk and vulnerabilities assessment, Islam et al. [14] present a risk assessment framework that starts with a threat analysis followed by a risk assessment to estimate the threat level and the impact level. This leads to an estimate of a security level for formulating high-level security requirements. The security level is qualitative, such as "low", "medium", and "high". Vanciu et al. [15] compare an architectural-level approach with a code-level approach in terms of the effectiveness of finding security vulnerabilities. Wang et al. [16] discuss their work on temporal metrics for software vulnerabilities based on the Common Vulnerability Scoring System (CVSS) 2.0. They use a mathematical model to calculate the severity and risk of a vulnerability, which is time dependent as in this work. Gawron et al. [17] investigate the detection of vulnerabilities in computer systems and computer networks. They use a logical representation of

preconditions and post conditions of vulnerabilities, with the aim of providing security advisories and enhanced diagnostics for the system. Wu and Wang [18] present a dashboard for assessing enterprise level vulnerabilities that incorporates a multi-layer tree-based model to describe the vulnerability topology. Vulnerability information is gathered from enterprise resources for display automatically. Farnan and Nurse [19] describe a structured approach to assessing low-level infrastructure vulnerability in networks. The approach emphasizes a controls-based evaluation rather than a vulnerability-based evaluation. Instead of looking for vulnerabilities in infrastructure, they assume that the network is insecure, and determine its vulnerability based on the controls that have or have not been implemented. Neuhaus et al. [20] present an investigation into predicting vulnerable software components. Using a tool that mines existing vulnerability databases and version archives, mapping past vulnerabilities to current software components, they were able to come up with a predictor that correctly identifies about half of all vulnerable components, with two thirds of the predictions being correct. Roumani et al. [21] consider the modeling of vulnerabilities using time series. According to these researchers, time series models provide a good fit to vulnerability datasets and can be used for vulnerability prediction. They also suggest that the level of the time series is the best estimator for prediction. Li et al. [22] present VulPecker, a tool for automatically detecting whether source code contains a particular vulnerability. Pang et al. [23] propose a technique based on a deep neural network to predict vulnerable software components. They claim that their technique can predict vulnerable Java classes in Android applications with high accuracy. Anand et al. [24] propose a model for classifying security patterns according to the type of vulnerability they address, claiming that their model helps software developers to select an appropriate security pattern once they know the type of vulnerability they would like to remove. The authors also claim that their classification scheme identifies missing security patterns, when no patterns can be found for particular vulnerabilities. Salfer and Eckert [25] consider the attack surface and vulnerability assessment of automotive electronic control units (ECUs). They propose a method and metric for assessing the attack surface and predicting the effort for a code injection exploit using ECU development data. They also provide an application of their method and metric to a graph-based security assessment.

With regard to vulnerabilities classification, Spanos et al. [26] look at ways to improve CVSS. They propose a new vulnerability scoring system called the Weighted Impact Vulnerability Scoring System (WIVSS) that incorporates the different impact of vulnerability characteristics. In addition, the MITRE Corporation [27] maintains the Common Vulnerability and Exposures (CVE) list of vulnerabilities and exposures, standardized to facilitate information sharing.

In terms of threat analysis, Schaad and Borozdin [28] present an approach for automated threat analysis of software architecture diagrams. Their work gives an example of automated threat analysis. Sokolowski and Banks [29] describe the implementation of an agent-based simulation model designed to capture insider threat behavior, given a set of assumptions governing agent behavior that pre-disposes an agent to becoming a threat. Sanzgiri and Dasgupta [30] present a taxonomy and classification of insider threat detection techniques based on strategies used for detection. Manzoor et al. [31] claim that contemporary cloud threat analysis approaches fail to include variants of identified vulnerabilities in their analyses. They target achieving a holistic cloud threat analysis procedure by designing a multi-layer cloud model, employing Petri Nets to comprehensively profile the operational behavior of the services in cloud operations. They use this model to identify threats within and across different operational layers. They further claim that their approach also looks at the variants of potential vulnerabilities to infer the cloud attack surface. Valani [32] looks at Secure DevOps threat modeling and concludes that maintaining speed to support business needs is difficult due to the fact that the threat modeling is too slow. He proposes the use of a lightweight threat modeling approach that uses a correlation matrix created from common lists and application abstractions, that is quicker and can be applied where detailed threat modeling is unnecessary.

The following publications fall into the other category. Kotenko and Doynikova [33] investigate the selection of countermeasures for ongoing network attacks. They suggest a selection technique based on the countermeasure model in open standards. The technique incorporates a level of countermeasure effectiveness that is related to the reliability of measures securing vulnerabilities, used in the SPRM(t) metric proposed in this work. Ganin et al. [34] present a review of probabilistic and risk-based decision-making techniques applied to cyber systems. They propose a decision-analysis-based approach that quantifies threat, vulnerability, and consequences through a set of criteria designed to assess the overall utility of cybersecurity management alternatives. Pendleton et al. [35] provide a systematic survey of systems security metrics. Based on this survey, they propose that an overall system security metric can be represented by the following dimensions of metrics: vulnerabilities, defenses, attacks, and situations. The situation dimension is focused on the current security state of a given system at a particular point in time, in order to account for dynamics related to system security states, including the level of vulnerabilities, attacks, and system defenses.

This author's directly related work includes [36], [8], and [1] where [8] is an expanded version of [36]. Yee [1] improves on [36] and [8] by a) adding time dependency, together with the notion that an organization's security level needs to be continuously evaluated, b) adding a new metric incorporating the reliability of the securing measures, and c) adding a description of new application areas. This work extends Yee [1] by adding the material mentioned at the start of Section I.

VII. Conclusion and Future Work

Since attackers continuously probe for new vulnerabilities to exploit, an organization cannot afford to assess its computer system's vulnerabilities once, secure some of the vulnerabilities, and then do nothing further.

Rather, the organization needs to assess and secure its vulnerabilities on a continuous basis, i.e., perform CAD. This work has proposed two conceptually clear SL metrics, verified as sound, that can be used to evaluate a computer system's security level at any point in time for CAD. One metric assumes that the measures securing vulnerabilities are totally reliable; the other considers the measures to be only partially reliable. CAD may be applied to specific types of vulnerabilities (e.g., vulnerabilities to insider attack), groupings of vulnerabilities that require special attention, specific application areas such as critical infrastructure and defence, and even at the code level. CAD may not be applied to areas that are missing any of the elements listed in Section V.C.

There are many security metrics in the literature, as seen in Section VI. The metrics in this work have the advantages of being easy to understand, and easy to calculate, which may be needed to convince management to provide the necessary resources required for CAD.

Future work includes formulations of other security metrics, the application of security metrics to critical infrastructure and defence, improving the methods for threat modeling, and exploring how this work may complement work in the literature and in the standardization community.

REFERENCES

[1] G. Yee, "Metrics for continuous active defence," Proc. Twelfth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2018), pp. 92-98, 2018.

[2] D. Rafter, "2019 Data Breaches: 4 Billion Records Breached So Far," Norton, retrieved December, 2019 from: https://us.norton.com/internetsecurity-emerging-threats-2019-data-breaches.html

[3] Identity Force, "2018 Data Breaches – The Worst of Last Year," retrieved November, 2019 from: https://www.identityforce.com/blog/2018-data-breaches

[4] Dark Reading, "2017 Smashed world's records for most data breaches, exposed information," retrieved November, 2019 from: https://www.darkreading.com/attacks-breaches/2017-smashed-worlds-records-for-most-data-breaches-exposed-information/d/d-id/1330987?elq_mid=83109&elq_cid=1734282&_mc=NL_DR_EDT_DR_weekly_20180208&cid=NL_DR_EDT_DR_weekly_20180208&elqTrackId=700ff20d23ce4d3f984a1cfd31cb11f6&elq=5c10e9117ca04ba0ad984c11a7dfa14b&elqaid=83109&elqat=1&elqCampaignId=29666

[5] G. Yee, "Visualization and prioritization of privacy risks in software systems," International Journal on Advances in Security, issn 1942-2636, vol. 10, no. 1&2, pp. 14-25, 2017.

[6] ITEM Software Inc.,"Reliability prediction basics", retrieved November, 2019 from: http://www.reliabilityeducation.com/ReliabilityPredictionBasics.pdf

[7] C. Salter, O. Saydjari, B. Schneier, and J. Wallner, "Towards a secure system engineering methodology," Proc. New Security Paradigms Workshop, pp. 2-10, 1998.

[8] G. Yee, "Optimal security protection for sensitive data," International Journal on Advances in Security, vol. 11, no. 1&2, pp. 80-90, 2018.

[9] G. Yee, "Designing good security metrics," Proceedings of the 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Milwaukee, WI, USA, pp. 580-585, July 15-19, 2019.

[10] N. Swartz, "The concepts of necessary conditions and sufficient conditions," Department of Philosophy, Simon Fraser University, 1997, retrieved November, 2019 from: https://www.sfu.ca/~swartz/conditions1.htm

[11] P. K. Manadhata and J. M. Wing, "An attack surface metric," IEEE Transactions on Software Engineering, vol. 37, no. 3, pp. 371-386, May/June, 2011.

[12] J. Stuckman and J. Purtilo, "Comparing and applying attack surface metrics," Proceedings of the 4th International Workshop on Security Measurements and Metrics (MetriSec '12), pp. 3-6, Sept. 2012.

[13] N. Munaiah and A. Meneely, "Beyond the attack surface," Proceedings of the 2016 ACM Workshop on Software Protection (SPRO '16), pp. 3-14, October 2016.

[14] M. Islam, A. Lautenbach, C. Sandberg, and T. Olovsson, "A risk assessment framework for automotive embedded systems," Proc. 2nd ACM International Workshop on Cyber-Physical System Security (CPSS '16), pp. 3-14, 2016.

[15] R. Vanciu, E. Khalaj, and M. Abi-Antoun, "Comparative evaluation of architectural and code-level approaches for finding security vulnerabilities," Proceedings of the 2014 ACM Workshop on Security Information Workers (SIW '14), pp. 27-34, Nov. 2014.

[16] J. A. Wang, F. Zhang, and M. Xia, "Temporal metrics for software vulnerabilities," retrieved: November, 2019. http://www.cs.wayne.edu/fengwei/paper/wang-csiirw08.pdf

[17] M. Gawron, A. Amirkhanyan, F. Cheng, and C. Meinel, "Automatic vulnerability detection for weakness visualization and advisory creation," Proc. 8th International Conference on Security of Information and Networks (SIN '15), pp. 229-236, 2015.

[18] B. Wu and A. Wang, "A multi-layer tree model for enterprise vulnerability management," Proceedings of the 2011 Conference on Information Technology Education (SIGITE '11), pp. 257-262, October 2011.

[19] O. Farnan and J. Nurse, "Exploring a controls-based assessment of infrastructure vulnerability," Proc. International Conference on Risks and Security of Internet and Systems (CRiSIS 2015), pp. 144-159, 2015.

[20] S. Neuhaus, T. Zimmermann, C. Holler, and A. Zeller, "Predicting vulnerable software components," Proc. 14th ACM Conference on Computer and Communications Security (CCS '07), pp. 529-540, 2007.

[21] Y. Roumani, J. Nwankpa, and Y. Roumani, "Time series modeling of vulnerabilities," Computers and Security, Vol. 51 Issue C, pp. 32-40, June 2015.

[22] Z. Li, D. Zou, S. Xu, H. Jin, H. Qi, and J. Hu, "VulPecker: an automated vulnerability detection system based on code similarity analysis," Proceedings of the 32nd Annual Conference on Computer Security Applications (ACSAC'16), pp. 201-213, Dec. 2016.

[23] Y. Pang, X. Xue, and H. Wang, "Predicting vulnerable software components through deep neural network," Proceedings of the 2017 International Conference on Deep Learning Technologies (ICDLT'17), pp. 6-10, June 2017.

[24] P. Anand, J. Ryoo, and R. Kazman, "Vulnerability-based security pattern categorization in search of missing patterns," Proceedings of the 2014 Ninth International Conference on Availability, Reliability and Security (ARES), pp. 476-483, Sept. 2014.

[25] M. Salfer and C. Eckert, "Attack surface and vulnerability assessment of automotive electronic control units," Proceedings of the 12th International Conference on Security and Cryptography (SECRYPT 2015), pp. 317-326, 2015.

[26] G. Spanos, A. Sioziou, and L. Angelis, "WIVSS: A new methodology for scoring information system vulnerabilities," Proc. 17th Panhellenic Conference on Informatics, pp. 83-90,

2013.

[27] MITRE, “Common vulnerabilities and exposures”, retrieved November, 2019 from: https://cve.mitre.org/

[28] A. Schaad and M. Borozdin, “TAM2: Automated threat analysis,” Proc. 27th Annual ACM Symposium on Applied Computing (SAC ’12), pp. 1103-1108, 2012.

[29] J. Sokolowski and C. Banks, “An agent-based approach to modeling insider threat,” Proc. Symposium on Agent-Directed Simulation (ADS '15), pp. 36-41, 2015.

[30] A. Sanzgiri and D. Dasgupta, “Classification of insider threat detection techniques,” Proc. 11th Annual Cyber and Information Security Research Conference (CISRC ’16), article no. 25, pp. 1-4, 2016.

[31] S. Manzoor, H. Zhang, and N. Suri, “Threat modeling and analysis for the cloud ecosystem,” Proceedings of the 2018 IEEE International Conference on Cloud Engineering, pp. 278-281, 2018.

[32] A. Valani, “Rethinking Secure DevOps threat modeling: the need for a dual velocity approach,” Proceedings of the 2018 IEEE Secure Development Conference (SecDev), pp. 136, 2018.

[33] I. Kotenko and E. Doynikova, “Dynamical calculation of security metrics for countermeasure selection in computer networks,” Proc. 2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, pp. 558-565, 2016.

[34] A. Ganin, P. Quach, M. Panwar, Z. A. Collier, J. M. Keisler, D. Marchese, and I. Linkov, “Multicriteria decision framework for cybersecurity risk assessment and management,” Risk Analysis, pp. 1-17, 2017.

[35] M. Pendleton, R. Garcia-Lebron, J.-H. Cho, and S. Xu, “A Survey on Systems Security Metrics,” ACM Computing Surveys (CSUR), Vol. 49, Issue 4, Article No. 62, pp. 1-35, February 2017.

[36] G. Yee, “Assessing security protection for sensitive data,” Proc. Eleventh International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2017), pp. 111-116, 2017.

A Guideline on the Analysis of Stochastic Interdependencies in Critical Infrastructures

Sandra König, Thomas Grafenauer,
Stefan Schauer, Manuel Warum
Austrian Institute of Technology GmbH
Digital Safety & Security Department
Vienna, Austria
{sandra.koenig, thomas.grafenauer,
stefan.schauer, manuel.warum}@ait.ac.at

Stefan Rass
Universität Klagenfurt
Institute of Applied Informatics
Klagenfurt, Austria
stefan.rass@aau.at

***Abstract*—Protecting Critical Infrastructures (CIs) requires decisions made about systems with complex dynamics, which rarely admits accurate descriptions or precise predictions. For this reason, simulation models are often probabilistic, embodying known (physical) laws to the extent possible, but generally adding a random element to account for unexpected events that security management is primarily concerned with. One such complex element is the interplay between different components of a CI, i.e., the dynamic *inside* a CI, which is more than just the sum of the mutual dependencies. Another important factor is the interplay *between* CIs, which might be understood between two specific CIs but less well known when it comes to mutual impacts. Simulations can help assessing these dependencies, but only to the extent as they are accurately specifiable. This work addresses the practical issues of using a probabilistic model to simulate cascading effects in interdependent CIs by proposing methods to allow for specifications carrying subjective uncertainty. The description follows a running example of a fictitious water provider, where a stochastic simulation model of incident propagation is embedded into its existing risk management process. Our exposition runs up to the final question of the decision maker about where to take action and how to prioritize assets regarding their need for protection, but also their role in impact propagation. The final picture delivered by the method outlined here is meant as a support for risk management decisions, containing possible concrete scenarios in an aggregate form. The value for a decision maker is the revelation of previously unseen influences and impacts besides the known causes and threats being subject of the risk management. This paper demonstrates how a stochastic model of dependencies between CIs can be integrated in a standard risk management process and illustrates each step for the case of a fictitious water provider.**

Keywords-critical infrastructure; dependencies; risk management; water supply

I. Introduction

Critical Infrastructures (CIs) are essential pillars of today's society that relies on availability of water, power, health care and transportation but also on the availability of food. Due of the high impact of a failure of even one of these infrastructures on society, a lot of research focuses on investigation of CIs. Of particular interest are the various interdependencies between CIs as these increase in number and type. For example, a hospital nowadays does not only depend on electricity, water and food supply, and a well functioning transportation system but also on information and control systems for intensive care or surgery. Even worse, interdependencies may amplify consequences of an reduced availability of one CI due to cascading effects. Such effects need to be taken into account when conducting a risk analysis [1]. While the local protection of a CI is possible using respective domain knowledge, securing the compound of several interacting CIs requires cross-domain expertise that is hardly available to the expert(s) in charge. Thus, to understand wide-range impacts cascading over several interdependent CIs, simulations are an indispensable tool to discover scenarios that require an orchestrated defence involving a collaboration of security officers in several distinct CI. Our model meets this need by letting each domain expert describe its own local CI, and leaving the interdependencies between two CIs as a matter of two domain experts agreeing on how their individual CIs interact with and depend on one another.

Since risk analysis is only one step in a more comprehensive risk management process, we here illustrate how such an advanced risk analysis can be integrated in an existing risk management process with the aim of yielding more accurate results. Our analysis uses the simulation tool described in [2]. The paper gives a step-by-step description of how to integrate a mathematical model into risk management processes and illustrates the procedure with an example. Practical aspects such as the use of expert opinions are discussed.

Related Work

Interdependencies between CIs have increased during the past decade and turned formerly loosely dependent CIs into a complex and highly interconnected network of CIs. The increasing complexity gave rise to numerous models of the dynamics inside a CI and between CIs. Early methods to describe those dynamics include Hierarchical Holographic Modeling (HHM) [3], followed by a multi-graph model for random failures [4] or input-output models [5]. However, most of these methods do not pay enough attention to nowadays interdependencies that yield to manifold effects of a single incident. The unpredictability of consequences shifted the focus towards stochastic models. While Markov models are

popular due to their simple structure, the applicability is often limited due to the exponentially growing state space. Models trying to cope with this issue allow for memory [6] but are challenging to put to practice due to their high complexity. The Interdependent Markov Chain (IDMC) model describes cascading failures in interdependent infrastructures in power systems [7], where every infrastructure is described by one discrete-time Markov chain where interdependencies between these chains are represented by dependent transition probabilities. A stochastic model allowing different degrees of failure while still being relatively simple to implement has been introduced in [8]. This paper extends previous work in the filed of cascading effects in interconnected networks [8], [9] but is also similar to approaches in IT security such as [10].

Incidents such as the disruption of electric power in California in 2001 [11], a power outage in Italy in 2003 [12] or failure of the nuclear plant in Fukushima, Japan, have demonstrated that interdependencies between various systems exist of which even experts were not aware. Intentional attacks such as the hacking of the Ukrainian power grid in 2015 [13], the Stuxnet worm [14] or the WannaCry ransomware (that hit hospitals particularly hard [15]) demonstrated the vulnerability of CIs due to the growing digitalization. Awareness of vulnerability due to cyberattacks has increased after recent attacks such as botnets [16] that nowadays also focus on critical information infrastructures [17]. Despite these well-known events, data is sparse and not sufficient to enable statistical analysis. Instead, simulation of such events and discussions with experts are needed to investigate consequences of incidents. The incidents of interest are both natural events (such as natural disasters) and man-made events, including unwanted interventions like cyberattacks or human error. Especially cyberattacks have recently moved into the center of attention and the EU Directive 2016/1148 on cyber security (also called the NIS-directive) describes regulations to increase the protection of CIs [18]. Simulating the consequences of an incident only requires knowledge about propagation dynamics and not about the type of incident (i.e., about the trigger) but even this information is typically difficult to get due to missing experience. Simulation methods are available to some extent, e.g., [19], and allow comparison of different models for specific situations. Motivated by the consequences of recent incidents, there is a growing interest in resilience of critical infrastructures [20]. An overview on models on interdependent CIs is presented in [21], while [22] gives an extensive review and comparison of different models of cascading effects in power systems. A general review on interdependencies between infrastructures with a focus on the different types of dependencies is given in [23]. The amount of research focusing on water supply and water providers seems to be more limited. A study focusing on security weaknesses of Industrial Control Systems (ICSs) and Supervisory Control and Data Acquisition (SCADA) systems and how to find good practices for water providers can be found in [24]. So far, there is only limited research focusing on incidents affecting a water provider. The impact of an Advanced Persistent Threat (APT) on a water provider has been investigated in [25] and [26].

In the following we demonstrate how to take interdependencies into account when analysing a critical infrastructure. The method is described for any critical infrastructure, as defined according to the European Commission [27].

> 'Critical infrastructure' means an asset, system or part thereof located in Member States which is essential for the maintenance of vital societal functions, health, safety, security, economic or social well-being of people, and the disruption or destruction of which would have a significant impact in a Member State as a result of the failure to maintain those functions.

The approach is illustrated focussing on a water provider.

Paper Outline

Since this article aims at illustrating the use of a theoretical model in practice, the remainder of this paper illustrates the various steps in detail for a example CI. Section II describes step by step how to integrate the analysis of stochastic dependencies between CIs in a risk management process. Section III then illustrates how the risk management process can be implemented for a fictitious water provider. Section IV provides concluding remarks and points out some directions of future work.

II. Incorporating Stochastic Dependencies in a Risk Management Process

This section illustrates how stochastic interdependencies between CIs can be incorporated in a standard risk management process. The process follows the ISO 31000 framework [28] and provides a step by step guide whose application will be illustrated by the analysis of a fictitious water provider in the following section. Our focus lies on the risk analysis (Step C below) as this is where the interdependencies between CIs have the biggest effect.

The upcoming analysis is based on modelling a CI as a set of interdependent *assets* (i.e., the relevant components of the CI) as a directed graph whose nodes represent the assets and edges represent the dependencies between them. Each asset carries the following information:

- *Criticality*: How important is the asset for the overall functionality of the CI?
- *Dependencies*: How critical is the asset for the functionality of other related assets?
- *Status indicator*: What is the level of functionality of each asset?

As for the last question, we here use different *states* on the scale $\{1, 2, 3\}$ to express increasing degrees of affection, ranging from status "working" (represented by value 1) up to the worst case status "outage" (represented by value 3), where the intermediate status level corresponds to limited functionality. More granular scales are possible but these three states are enough for illustration purposes.

Remark 1: It is important to note that we use the general term "asset" here, as in the standard risk management literature, when investigating risks *within* a CI. When interested in dependencies *between* CIs, e.g., the interplay between power providers, hospitals and water suppliers, a high-level perspective may see each CI as an asset and apply the model of stochastic dependencies. In this work, we focus on a single CI's situation and will thus hereafter use the term asset to describe a component of a CI.

A. Establishing the context

As a very first step it is necessary to understand the situation at hand. This includes both a description of how the water provider works internally as well as a deeper understanding of the overall context, i.e., dependencies to other CIs that provide input or require input for their part to ensure smooth operation. Dependencies between CIs are manifold and require a thorough analysis. In particular, dependencies are in no way limited to visible physical and known cyber connections but the analysis should also take into account logical interdependencies between different parts, as in the case of a control system.

A useful way to obtain an overview of the situation (and to discover potential missing dependencies) is visualization through a graph. To this end, a full list of components of the infrastructure as well as a full list of providers they depend on is represented in a network model. In a large network it may be useful (or even necessary) to classify dependencies according to their properties and only assign values to every class of connection, such as assigning one of the types "water", "communication" or "electricity" to every connection. This allows for distinction of different relationships but at the same time avoids an excessive amount of assessments.

Finally, context and focus of the risk management process are determined, defining which parts of the organization is covered by the analysis (which assets are relevant) and which criteria are used to evaluate the significance of risks, but also answering organizational questions such responsibilities and resources for the upcoming analysis.

B. Risk Identification

Next, it is important to identify the relevant risks. It is useful to distinguish the terms "threat", "vulnerability" and "risk" in the following: a *threat* is any factor or condition that can impact the correct functionality or security of a system. A *vulnerability* is any condition or property of a system that can lead to affection by a threat. A *risk* is the coincidence of threat and vulnerability. A security incident is then the physical event of a threat that hits some vulnerability and thereby causes an impact on the system. Quantitatively, risk is understood as the product of impact and the likelihood for the impact to occur.

In order to get a comprehensive overview, several sources need to be taken into account. General technical vulnerabilities are collected in databases such as the National Vulnerability Database (NVD) [29] while specific vulnerabilities of software and hardware components may be detected by use of automated vulnerability scanners such as Nessus [30] or OpenVAS [31]. Historic data help identify threats specific to the CI and discussions with experts form the field help understand which of these threats are actually risks. This step yields a list of the relevant risks that are analysed in the following.

C. Risk Analysis

This step is about getting a deeper understanding of the risks identified in the last step. Which assets are directly affected by each risk? What are the indirect consequences? How likely is each risk to occur?

Reports on past incidents are a good source to enhance understanding of risks. However, such data is not always available, either due to the rare occurrence or due to the fact that only necessary information is reported.

Consequences of an Incident

This step aims at estimating the consequences of an incident, i.e. of a realisation of each of the risks considered. To this end we apply the stochastic model introduced in [8] that also allows simulation. The simulation assumes that a certain incident has just occurred and directly affects one or more assets by putting them from functional into affected or even outage state. Based on the dependency information the status of related (dependent) assets is updated accordingly, where each asset may individually undergo different status changes, depending on the importance of the other asset (e.g., a mild affection may occur if the failed asset provides only a small part of the supply, or a severe affection may occur if an asset vitally depends on another yet failed asset). This reveals *cascading effects*, i.e., indirect impacts of a realisation of a risk scenario that are not evident at first sight. State transitions are supposed to happen probabilistically to cover cases of deterministic dependencies (e.g., a pump is fully relying on a continuous electricity supply) as well as probabilistic dependencies (e.g., water shortage can temporarily be overcome by backup water reservoirs). Application of the model [8] asks experts to provide assessments for the identified risks and to discuss the consequences of a realisation of each of these risks. While the model describes the propagation mechanisms and provides an estimate of the overall impact of an incident on a CI it requires knowledge about the effect of a failure of one single component on the ones directly depending on it. The core duty of the modeling then boils down into two major tasks:

1) Enumerate all assets and identify interdependencies between them as detailed as possible. In the following, we use the arrow notation $A \rightarrow B$ to denote a dependency of asset B on asset A (cf. Figure 1, e.g., where the pump B depends on the water A). A directed graph may be used to illustrate the situation.
2) Based on this information, specify probabilities for state changes in a dependent asset B, if the provider asset A is not working properly (i.e., is in state 2 or 3).

The first step typically collects information that is known and available to the CI operator. The challenging part is specification of the transition probabilities in the second step. This is a general issue in any probabilistic model (i.e., not specific to the one applied here), together with the occurring costs to the CI operator in terms of human resources.

The sought transition probabilities describe how likely it is that limited functionality or a complete shutdown of one component affects the dependent components. While historic data may provide some information on these transitions such data is rarely (publicly) available so that expert knowledge is the only remaining source. Despite the human resource cost to the CI this source is of high value since experienced employee often have a profound knowledge that is not available in written form. Still, experts may find it hard to provide precise estimates of likelihoods but rather have an idea about what is likely to happen next (based on past incidents). Aware of this problem, we avoid asking for precise numerical values but rather look for an assessment on a qualitative scale, as recommended in risk management (e.g., by the German Federal Office for Information Security (BSI) [32]). One way to address this problem is to ask for a qualitative prediction combined with a statement on the confidence of this estimate [1]. According to this approach we ask experts to answer the following two questions for each transmission probability t_{ij} that needs to be estimated.

> If the provider is in state i, how likely is it that this will put the dependent asset into state j?

Since this is usually hard to answer, we replace it by the following two simpler questions.

1) "If the provider is in state i, what is the most likely state j that the dependent asset will be in after this incident?"
2) "How certain are you about this prediction?"

The answers are to be chosen from a set of predefined values, namely from the set of states $\{1, \ldots, k\}$ for 1) and a set of possible confidence levels for 2). Even if the expert is unsure about the consequences, he typically still has a reasonable idea about the intensity of the consequences, i.e., if the expected consequences will be "close" to the predicted value. Because of this, the method assumes that in the case of uncertain assignments similar values as the predicted one are also likely to occur. In case an expert does not feel competent enough to make a prediction it is assumed that all possible values occur with the same likelihood. This intuition can be formalized and yields a probability distribution over the set of all possible states as shown in Table I.

TABLE I. Distribution over the CI's possible next state based on the expert's assignment

prediction	totally sure	somewhat unsure	totally unsure
1	(1,0,0)	(2/3, 1/3, 0)	(1/3,1/3,1/3)
2	(0,1,0)	(1/4, 2/4, 1/4)	(1/3,1/3,1/3)
3	(0,0,1)	(0, 2/3, 1/3)	(1/3,1/3,1/3)

Exact predictions may be difficult to provide even for very experienced people working in the respective domain. In order to capture this fact, the model can be extended to allow experts to be mistaken with a small probability ε even when they are sure about their prediction. We discussed this problem and a possible solution in [1] but will not go into detail here to keep the focus on the overall risk management process.

The input to the simulation is a network graph of connected assets of a critical infrastructure where each of these assets is in one specific state representing its functionality level. This graph resembles the picture in Figure 1 but additionally augments each node with matrices indicating the status change probabilities for each dependency. Dependencies may change over time, i.e., a short-term outage of a providers it typically less severe than a lasting reduced availability. This is particularly true for power supply (that is the backbone of every CI) where emergency power supply should prevent damage for a limited time period. The simulation will thus need a state transition probability matrix *per dependency* and *per time frame*.

The simulation prototype [2] distinguishes short, medium and long-term dependencies. For each of these time frames the probabilities $t_{ij} = \Pr(B \text{ is in state } j | A \text{ switches into state } i)$ describe the transition regimes. While the stochastic model allows a recovery (i.e., switching back into a better status), this is not yet implemented in the current version of the prototype.

The simulation starts when a risk scenario becomes reality, imitating probabilistic transitions (as for a Markov chain), and stops after a predefined time horizon (with the rationale that far-fetching forecasts become increasingly unreliable and hoping that after a decent amount of time some countermeasures can be taken). Each simulation run yields a time series of state transitions for all assets of the CI.

Given a number of simulations of a risk scenario (that may contain several risk, see [1]), the final states per assets can be averaged to get an estimate of the likelihood that this part of the CI is affected. For visualization it is helpful to apply color codes, ranging from green (symbolizing a working state) to red (symbolizing an outage), alerting about the criticality of the current condition. Numerically, the simulation results may be summarized as a table that lists the number of components which are on average in any of the possible states. The `OMNeT++` tool is used here to support the visualization and execution of our simulation, as described in [2]. Various additional outputs are possible, such as plots of time-lines relating to a single simulation run. This would display the times when a CI asset changes its state, and would show the temporal development of the cascading impacts.

Remark 2: It must be kept in mind that the simulation does *not* provide information about the likelihood for an incident to occur but starts from a given scenario that is assumed to have happened. The simulation then yields information on the likelihood of the consequences of this scenario.

D. Risk Evaluation

Risk evaluation is concerned with prioritizing the risks identified in Step II-B according to the criteria chosen in Step II-A. Classical risk management approaches use a cost-benefit analysis to decide on which risks are treated first. More

advanced approaches based on game theory allow optimizing several goals simultaneously, thus also take into account non-monetary factors such as reputation or employees satisfaction [33].

As part of the overall risk management process, risk evaluation takes the results of the risk analysis step II-C into account. Based on this, it is possible to compare the different risks in terms of the impact they have on the CI (according to the considered goals). Ordering risks is doable in manifold ways, such as taking expectations (i.e., risk = impact $\times$ likelihood) or by lexicographic order on impacts. This is our choice in the following, based on the HyRiM stochastic ordering [34]. This approach corresponds to minimization of the likelihood of the worst possible damage.

E. Risk Treatment

Risk Treatment classically focuses on ways to mitigate risks by means of improving existing controls or implementing new controls in order to either reduce the likelihood of occurrence or the magnitude of the consequences and the selection of the controls to be implemented is often subjective. Advanced approaches apply game theory to find (a mix of) optimal controls [35] by considering various defence actions and selecting an optimal selection of these.

Based on the risk evaluation step II-D, risk treatment evaluates mitigation actions tailored to the specific risk scenario. In order to achieve that, it is necessary to have information on the *reasons* for a failure (root cause analysis), which simulations can deliver (we will illustrate this in Section III-E). Based on this information, we can proceed to find precautions and defence actions that provide optimal protection. Once these are implemented, risks can be reassessed (following the same steps but using updated assessment reflecting the new situation) to measure the effectiveness of the treatment.

Another way to treat the analysed risks is to identify mitigation strategies. Since resources are limited, an important task is selecting from a set of potential strategies. A method to solve this optimization problem is to consider the different risks as strategies of an attacker (nature in this case) and let the operator of the CI defend his system. At first sight, this might be an inappropriate model for two reasons. First, game theory assumes *rational* players that are able to predict the (best) responses of the other players to their actions. Still game theory is able to provide reasonable results when applying a zero sum game, as this type of game assumes that the attacker want to cause maximal damage. When the defender plays his optimal attack for this worst case, he will only be better off if the attacker deviates from his optimal strategy due to the characterization of a Nash equilibrium. The identified solution may not be as efficient as if we knew the attackers intentions but they yield an upper bound to the expected damage. The second issue with game theoretic models is the traditional assumption that payoffs are real valued. This assumption can however be generalized under very mild assumptions [36] such payoffs may be random. Thus the impact can be estimated through a simulation model as described above and payoffs of the game are the estimated distributions over all possible states. A Nash equilibrium can be computed numerically, e.g., in R [37]. Application of this game theoretic setting to CIs are illustrated in [35], [38].

III. Analysing a Water Provider

This section demonstrates how to put the process described in Section II into practice for a specific CI. In the following, we analyse risks faced by a fictitious water provider following the steps laid out above. All assessments and estimates presented in this paper are illustrative only, since any such data is sensitive and hence protected. However, the data used is based on discussions with experts of the field to be as realistic as possible. The main goal is to illustrate how to analyse the consequences of a risk scenario affecting a CI that is part of an entire network of interdependent CIs, i.e., depends on some CIs and in turn provides important input to other CIs.

In the following, we consider a European water provider of average size providing its services to a town with several hundred thousands inhabitants as well as some municipalities in the surrounding area. The water provider is responsible for planning, building and maintaining the entire water network, but his main focus is on the availability of the drinking water. In order to ensure a sustainable water quality, the provider supports water processing and sewage cleaning by using an ICS. The considered use case assumes various sources for water, namely a mountain spring, a river and a well. The two latter use pumps to lift the water above ground level. The water is further treated at the water plant to increase the quality (e.g., through removal of undesired chemicals or adding of minerals). Transportation paths are short due to the geography of the landscape and the number of necessary lines is correspondingly low. Several reservoirs are available to ensure water supply in case of high demand, e.g., to extinguish fire.

The water provider relies on the transportation system, in particular on roads, e.g., to be able to perform maintenance and manual checks, and as many other CI it crucially depends on electricity. In this scenario, we consider an internal power plant that contributes approximately 30% of the required energy while the remaining power comes from external providers. Redundancy in the system and an emergency power supply help to mitigate the criticality of this dependency. In case of a reduction or an interruption of electricity, the utility provider is still able to guarantee supply with drinking water up to three days due the precautionary measures.

On the other hand, the water provider is important for numerous other CIs in the region. It supplies drinking water to hospitals, schools and grocery stores but also cooling water for both hospitals and industrial companies. The actual relevance of each of these connections can only be assessed by the CIs depending on it, which requires discussions with the corresponding experts and thus goes beyond the scope of this use case that focuses on the risk management for a water utility. The remainder of this section presents an analysis of the effects stemming from the realisation of a risk using

a qualitative risk assessment performed by experts from the water domain.

A. Establishing the context

The main input to context establishment is discussion with experts. In detail, we discussed the importance of each asset for the functionality of water supply which yielded a list as given in Table II. Besides the elements required for production (river and well pump), purification (water plant) and storage (water reservoir) this list also contains essential elements of water distribution, namely two parts of a distribution network. Functionality of this network is essential for the delivery of water to dependent CIs. In order to at least indicate the complexity of this distribution network we composed it of two different parts where the second part is located at a higher altitude than the first part such that a pump is required for transportation. Both the river pump and the well pump are abstract nodes that are supposed to describe the behaviour of all pumps of the corresponding type (if more than one exist). This abstraction allows us to model situations where a significant number of pumps fail so that it affects availability of water.

TABLE II. List of Infrastructures and Providers

Number	Object
1	Water Plant
2	Mountain Spring
3	Well
4	Well Pump
5	Water Reservoir
6	River
7	River Pump
8	Power Grid
9	Communication
10	SCADA server
11	Distribution network 1
12	Distribution Pump
13	Distribution network 2

Besides the assets themselves, emergency systems and back-ups need to be accounted for in the modeling process, e.g., we assume the existence of emergency power stations as required by (Austrian) law. Such components are indirectly taken into account when assessing the dependencies (in particular when setting up the transmission regime; shortage in power supply only affects other parts after a certain time) rather than being assets themselves.

The situation of the fictitious water provider is displayed in Figure 1 showing all relevant interdependencies between the assets. Initially, we assume that the everything is working smoothly, i.e., every asset is in states 1, until an incident happens. In our small example, we refrain from categorising the connections but rather assess every single connection.

B. Risk Identification

In order to identify the relevant risks we discussed potential threats with experts. The most significant ones for a water provider turned out to be the following ones.

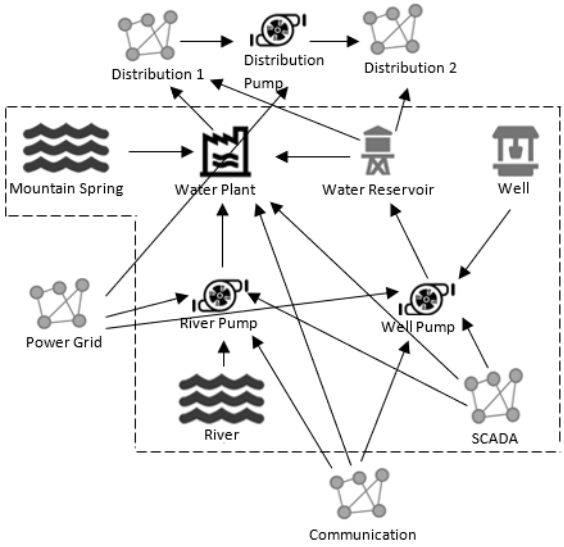


Fig. 1. Visualization of water use case

- R_1: flooding
- R_2: extreme weather conditions
- R_3: leakage of hazardous material (water contamination)
- R_4: cyberattack (e.g., on SCADA server)

Risk R_2 indicates the general vulnerability of water supply on the weather. For the upcoming analysis however, we need to be more specific in order to analyse the risk in the next step. Short-term heavy rain is not as a severe problem for a water provider (it certainly is for other CIs), since the main source of the water provider is groundwater. While it may cause smaller damage to the infrastructure, it will not interrupt water supply which is the core interest of our fictitious water provider. Some experts see flash floods an underestimated hazard and raise awareness [39]. Currently, droughts are considered more problematic for a water utility, in particular because they are likely to become more frequent in near future. Thus, we analyse the risk R_2 to be a *heat wave* in the following.

C. Risk Analysis

Knowing the relevant risks, it is necessary to understand each of these in detail. First, we identify the likelihood of each asset to be directly affected by a realisation of each of the risks. A flooding affects single sites (such as a well) but is typically not critical for the overall functionality for the water supply. Further, a realisation of risk R_1 may yield a limited operation of wells and springs as water may be contaminated by particles (germs, bacteria and others) induced by the flood. Depending on the degree of contamination, water may be boiled to make it drinkable or needs to be purified technically, which is a costly and time-consuming process. Recent floods such as the flooding in central Europe in 2002 [40] and 2013 [41], [42] indicate an increased likelihood of occurrence [43]. Concerning a realisation of R_2 we here focus on an extraordinarily dry period. Various water sources may dry up, in particular rivers or wells, but we assume that at least some sources

like ground water remain available. A drought implies also an increased need for water and thus yields a peak in consumption which in turn challenges the infrastructure. Peaks will cause additional costs for the provider but are not considered here any further since this does not affect other parts of the system. The effect of leakage of hazardous material strongly depends on the circumstances of the leakage and to some extent on the material. The crucial factor is the extent of the leakage as this changes the impact significantly. A bounded contamination is not a severe issue as long as the water network is close-meshed (i.e., there is enough redundancy in the network). However, if groundwater or a large number of wells are affected, water purification may take several months. For our use case, we assume a realisation of R_3 that is a limited spreading, affecting mainly the river and only with small likelihood also affects the mountain spring or the well. Since contamination seriously affects the quality of drinking water [44], the probability that it switches to the worst possible state 3 is high. A realisation of R_4 is most challenging to investigate because data is typically (and luckily) sparse. Recent incidents in ports [45]–[47] and in particular prominent attacks such as Wannacry [15] or NotPetya [48] allow a heuristic estimation of consequences. In order to perform simulations it is necessary to make some basic assumptions about the spreading process (e.g., does it spread via email or not).

Understanding a risk includes identifying those assets that are directly affected by a realisation of the risk and estimating the expected impact. Tables III, IV, V and VI give the estimated probability distributions over the various states for short, medium and long-term of those assets that are directly affected (those not directly affected are omitted and the corresponding likelihoods set to zero). Note that the tables only contain the necessary information, i.e., the chance that the asset is not affected by the considered risk (stays in state 1) is such that the sum of the corresponding row is one.

TABLE III. Direct Impact of R_1 on Assets

Asset	Impact	Short	Medium	Long
Mountain Spring	limitation	0.3	0.4	0.5
	failure	0.2	0.3	0.4
Well	limitation	0.2	0.3	0.4
	failure	0.1	0.2	0.3

TABLE IV. Direct Impact of R_2 on Assets

Asset	Impact	Short	Medium	Long
Mountain Spring	limitation	0.1	0.2	0.3
	failure	0.0	0.2	0.3
Well	limitation	0.2	0.3	0.4
	failure	0.0	0.1	0.2
River	limitation	0.6	0.4	0.2
	failure	0.2	0.4	0.6

Identification of indirect consequences of a risk on each of the assets is supported by the simulation, as described in Section II-C and will be demonstrated in detail in the next paragraph.

TABLE V. Direct Impact of R_3 on Assets

Asset	Impact	Short	Medium	Long
Mountain Spring	limitation	0.0	0.0	1/3
	failure	1.0	1.0	2/3
Well	limitation	0.0	0.0	1/3
	failure	1.0	1.0	2/3
River	limitation	0.0	0.0	1/3
	failure	1.0	1.0	2/3

TABLE VI. Direct Impact of R_4 on Assets

Asset	Impact	Short	Medium	Long
Communication	limitation	0	$1/3$	$1/3$
	failure	1	$2/3$	$2/3$
SCADA	limitation	$1/2$	$1/3$	$1/3$
	failure	$1/4$	$1/3$	$2/3$

Finally, the likelihood of occurrence is estimated for each risk to add to the picture. Additionally, the likelihood of failure and impairment of the CI due to a realisation of each risk are rated as “negligible”, “low”, “medium”, “high” or “very high” by experts. Where available, public reports and statistics may complement such subjective assessment and yield refined estimates. In case of a cyberattack the estimates highly depend on the assumptions about the attacker, e.g., whether he plans a highly sophisticated APT or a more general malware attack. The values for this use case are given in Table VII.

TABLE VII. Overall Likelihood Assessment for Risks

Risk	Occurrence	Failure	Impairment
R_1: flooding	medium	negligible	negligible
R_2: heat wave	medium	negligible	medium
R_3: contamination	low	negligible	medium
R_4: cyberattack	medium	low	medium

Consequences of an Incident

The consequences of a realisation of a risk are estimated based on a simulation model, as described in Section II-C. The dependencies between the assets are assessed for three different time horizons, taking into account the dynamic nature of CIs. For example, if emergency power supply is available, the likelihood for a pump to switch to the outage state 3 given the electricity goes off is zero for the first couple of hours, and changes to 1 if the emergency generator runs out of fuel, unless the original power supply has recovered in the meantime. The assessments given in Tables VIII, IX and X are based on several discussion with domain experts. Impact was measured on the three-tier scale “negligible” (state 1), “medium” (state 2) and “high” (state 3) while the experts’ confidence in the provided prediction is described as “totally sure”, “somewhat unsure” or “totally unsure”. Note that these assessments need to be made for each specific connection and do neither contain information about potential substitutes (e.g., if several pumps are available) nor take into account the option of repair or recovery. The assessment is solely concerned about the nature of this specific dependency between the two assets.

The assessments from Tables VIII, IX and X are mapped to the corresponding transition matrices as described in Table

TABLE VIII. Short-term impact assessment

Link	Problem	Prediction	Confidence
River Pump → Water Plant	limitation	negligible	totally sure
	failure	negligible	totally sure
Mountain Spring → Water Plant	limitation	negligible	totally sure
	failure	negligible	totally sure
Communication → Water Plant	limitation	medium	somewhat unsure
	failure	negligible	totally sure
Water Reservoir → Water Plant	limitation	negligible	totally sure
	failure	negligible	totally sure
SCADA → Water Plant	limitation	high	somewhat unsure
	failure	high	somewhat unsure
Well → Well Pump	limitation	negligible	totally sure
	failure	negligible	somewhat unsure
Communication → Well Pump	limitation	medium	somewhat unsure
	failure	negligible	totally sure
Power Grid → Well Pump	limitation	negligible	totally sure
	failure	negligible	totally sure
SCADA → Well Pump	limitation	medium	somewhat unsure
	failure	high	somewhat unsure
River → River Pump	limitation	negligible	totally sure
	failure	negligible	somewhat unsure
Communication → River Pump	limitation	medium	somewhat unsure
	failure	negligible	totally sure
Power Grid → River Pump	limitation	negligible	totally sure
	failure	negligible	totally sure
SCADA → River Pump	limitation	medium	somewhat unsure
	failure	high	somewhat unsure
Well Pump → Water Reservoir	limitation	negligible	totally sure
	failure	negligible	totally sure
Water Plant → Distribution 1	limitation	negligible	totally sure
	failure	negligible	totally sure
Water Reservoir → Distribution 1	limitation	negligible	totally sure
	failure	negligible	totally sure
Water Reservoir → Distribution 2	limitation	negligible	totally sure
	failure	negligible	totally sure
Distribution Pump → Distribution 2	limitation	negligible	totally sure
	failure	negligible	somewhat unsure
Power Grid → Distribution Pump	limitation	negligible	totally sure
	failure	negligible	totally sure
Distribution 1 → Distribution Pump	limitation	negligible	totally sure
	failure	negligible	somewhat unsure

TABLE IX. Medium-term impact assessment

Link	Problem	Prediction	Confidence
Pump → Water Plant	limitation	negligible	totally sure
	failure	negligible	somewhat unsure
Mountain Spring → Water Plant	limitation	negligible	totally sure
	failure	negligible	somewhat unsure
Communication → Water Plant	limitation	negligible	totally sure
	failure	negligible	totally sure
Water Reservoir → Water Plant	limitation	negligible	totally sure
	failure	negligible	somewhat unsure
SCADA → Water Plant	limitation	medium	somewhat unsure
	failure	medium	somewhat unsure
Well → Well Pump	limitation	medium	somewhat unsure
	failure	high	somewhat unsure
Communication → Well Pump	limitation	negligible	totally sure
	failure	negligible	totally sure
Power Grid → Well Pump	limitation	negligible	totally sure
	failure	negligible	totally sure
SCADA → Well Pump	limitation	medium	totally sure
	failure	medium	totally sure
River → River Pump	limitation	medium	somewhat unsure
	failure	high	somewhat unsure
Communication → River Pump	limitation	negligible	totally sure
	failure	negligible	totally sure
Power Grid → River Pump	limitation	negligible	totally sure
	failure	negligible	totally sure
SCADA → River Pump	limitation	medium	totally sure
	failure	medium	totally sure
Well Pump → Water Reservoir	limitation	negligible	totally sure
	failure	negligible	somewhat unsure
Water Plant → Distribution 1	limitation	negligible	totally sure
	failure	negligible	somewhat unsure
Water Reservoir → Distribution 1	limitation	negligible	totally sure
	failure	negligible	somewhat unsure
Water Reservoir → Distribution 2	limitation	negligible	totally sure
	failure	negligible	somewhat unsure
Distribution Pump → Distribution 2	limitation	negligible	somewhat unsure
	failure	high	somewhat unsure
Power Grid → Distribution Pump	limitation	medium	somewhat unsure
	failure	negligible	somewhat unsure
Distribution 1 → Distribution Pump	limitation	medium	somewhat unsure
	failure	negligible	somewhat unsure

I, e.g., yielding transition regimes as shown in Table XI corresponding to the first few rows of Table VIII. The fact that dependencies may change over time is here captured by the three different time horizons "short", "medium" and "long". The simulation requires definitions for all three time periods for each dependency. Table XII shows the definitions applied in this use case. Time starts at zero and the numbers given in each column are the upper bound of the corresponding range, e.g., for the connection River Pump → Water Plant short-term is a duration up to 12 hours, medium-term is between 12 an 48 hours and long is between 48 and 72 hours (when the simulation stops). For consistency, the lower limits are excluded and the upper ones are included, e.g., medium-term means $t \in (12, 48]$. The time definitions for the impacts are all equal and set to 1 hour for short-term, 12 hours for medium-term and 48 hours for long-term.

With this information the simulation can be run to estimate the impact of a realisation of the identified risks. Figure 2 shows the results of one run of the simulation (for a realisation of a contamination). The colouring of the nodes represents the state of the assets where dark grey indicates failure (state 3), medium grey indicates limited availability (state 2) and light grey means normal operation (state 1). Since the simulation contains stochastic elements, it is necessary to run a large number of repetitions. With the parameters specified above, we run the simulation 1000 times and estimate the probability distribution over the possible states for each asset. For a realisation of R_1, we considered the scenario of a flooding that lasts 4 days. The empirical probability distributions over the possible states for each asset are shown in Table XIII. For a realisation of R_2, we considered the scenario of heat wave lasting 7 weeks. The empirical probability distributions over the possible states for each asset are shown in Table XIV. For a realisation of R_3, we considered the scenario of a contamination that lasts 7 weeks. The empirical probability distributions over the possible states for each asset are shown in Table XV. For a realisation of R_4, we considered the scenario of a cyberattack that lasts 2 hours. The empirical probability distributions over the possible states for each asset are shown in Table XIII.

Note that the impact scale is influenced by the interests of the CI provider. The degree of damage could be measured in

TABLE X. Long-term impact assessment

Link	Problem	Prediction	Confidence
Pump → Water Plant	limitation	negligible	totally sure
	failure	medium	somewhat unsure
Mountain Spring → Water Plant	limitation	negligible	totally sure
	failure	medium	somewhat unsure
Communication → Water Plant	limitation	negligible	totally sure
	failure	negligible	totally sure
Water Reservoir → Water Plant	limitation	negligible	totally sure
	failure	medium	somewhat unsure
SCADA → Water Plant	limitation	medium	somewhat unsure
	failure	medium	somewhat unsure
Well → Well Pump	limitation	medium	somewhat unsure
	failure	high	totally sure
Communication → Well Pump	limitation	negligible	totally sure
	failure	negligible	totally sure
Power Grid → Well Pump	limitation	negligible	totally sure
	failure	high	totally sure
SCADA → Well Pump	limitation	medium	totally sure
	failure	medium	totally sure
River → River Pump	limitation	medium	somewhat unsure
	failure	high	totally sure
Communication → River Pump	limitation	negligible	totally sure
	failure	negligible	totally sure
Power Grid → River Pump	limitation	negligible	totally sure
	failure	high	totally sure
SCADA → River Pump	limitation	medium	totally sure
	failure	medium	totally sure
Well Pump → Water Reservoir	limitation	negligible	totally sure
	failure	medium	somewhat unsure
Water Plant → Distribution 1	limitation	negligible	somewhat unsure
	failure	medium	somewhat unsure
Water Reservoir → Distribution 1	limitation	negligible	totally sure
	failure	negligible	somewhat unsure
Water Reservoir → Distribution 2	limitation	negligible	totally sure
	failure	negligible	somewhat unsure
Distribution Pump → Distribution 2	limitation	medium	somewhat unsure
	failure	high	totally sure
Power Grid → Distribution Pump	limitation	medium	somewhat unsure
	failure	high	somewhat unsure
Distribution 1 → Distribution Pump	limitation	medium	somewhat unsure
	failure	high	totally sure

TABLE XI. Short-term transition probabilities

$$\mathrm{P}^{\text{short}}_{\text{river pump}\rightarrow\text{water plant}} = \begin{array}{c} \\ s_{rp}{=}1 \\ s_{rp}{=}2 \\ s_{rp}{=}3 \end{array} \begin{array}{c} \begin{array}{ccc} s_{wp}{=}1 & s_{wp}{=}2 & s_{wp}{=}3 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \end{array}$$

$$\mathrm{P}^{\text{short}}_{\text{mountain spring}\rightarrow\text{water plant}} = \begin{array}{c} \\ s_{ms}{=}1 \\ s_{ms}{=}2 \\ s_{ms}{=}3 \end{array} \begin{array}{c} \begin{array}{ccc} s_{wp}{=}1 & s_{wp}{=}2 & s_{wp}{=}3 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \end{array}$$

$$\mathrm{P}^{\text{short}}_{\text{communication}\rightarrow\text{water plant}} = \begin{array}{c} \\ s_{com}{=}1 \\ s_{com}{=}2 \\ s_{com}{=}3 \end{array} \begin{array}{c} \begin{array}{ccc} s_{wp}{=}1 & s_{wp}{=}2 & s_{wp}{=}3 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 1/4 & 2/4 & 1/4 \\ 1 & 0 & 0 \end{pmatrix} \end{array}$$

TABLE XII. Time assessments for short-, medium- and long-term

Link	Short	Medium	Long
River Pump → Water Plant	12 hours	48 hours	72 hours
Mountain Spring → Water Plant	12 hours	48 hours	72 hours
Communication → Water Plant	12 hours	48 hours	72 hours
Water Reservoir → Water Plant	12 hours	48 hours	72 hours
SCADA → Water Plant	1 hour	2 hours	6 hours
Well → Well Pump	12 hours	48 hours	72 hours
Communication → Well Pump	12 hours	48 hours	72 hours
Power Grid → Well Pump	8 hours	48 hours	72 hours
SCADA → Well Pump	1 hour	2 hours	6 hours
River → River Pump	12 hours	48 hours	72 hours
Communication → River Pump	12 hours	48 hours	72 hours
Power Grid → River Pump	8 hours	48 hours	72 hours
SCADA → River Pump	1 hour	2 hours	6 hours
Well Pump → Water Reservoir	12 hours	48 hours	72 hours
Water Plant → Distribution1	12 hours	48 hours	72 hours
Water → Distribution1	12 hours	48 hours	72 hours
Water Reservoir → Distribution2	12 hours	48 hours	72 hours
Distribution Pump → Distribution2	24 hours	48 hours	72 hours
Power Grid → Distribution Pump	8 hours	48 hours	72 hours
Distribution1 → Distribution Pump	12 hours	48 hours	72 hours

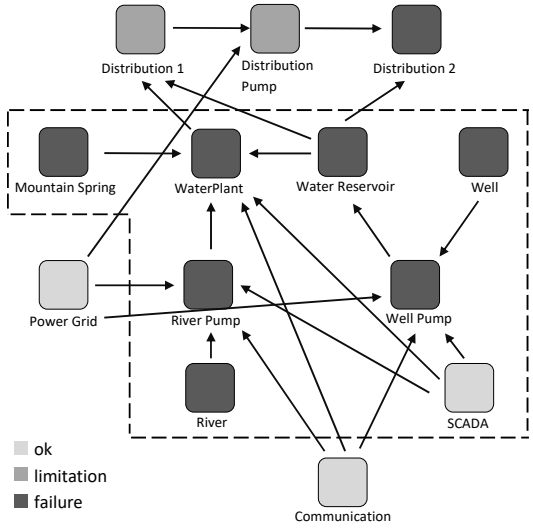


Fig. 2. Results of one run simulating scenario R_3

TABLE XIII. Estimated Impact of R_1

	state 1	state 2	state 3
Water Plant	0.384	0.422	0.194
Mountain Spring	0.011	0.325	0.664
Well	0.099	0.380	0.521
Well Pump	0.122	0.200	0.678
Water Reservoir	0.427	0.398	0.175
River	1.000	0.000	0.000
River Pump	1.000	0.000	0.000
Power Grid	1.000	0.000	0.000
Communication	1.000	0.000	0.000
SCADA server	1.000	0.000	0.000
Distribution network 1	0.672	0.290	0.038
Distribution Pump	0.690	0.148	0.162
Distribution network 2	0.670	0.137	0.193

terms of the number of affected customers, the time needed reassure availability of drinking water, the amount of resources necessary to overcome a shortage (in terms of money or person

TABLE XIV. Estimated Impact of R_2 on Assets

	state 1	state 2	state 3
Water Plant	0.144	0.547	0.309
Mountain Spring	0.228	0.344	0.428
Well	0.191	0.547	0.262
Well Pump	0.227	0.268	0.505
Water Reservoir	0.584	0.286	0.130
River	0.013	0.188	0.799
River Pump	0.023	0.093	0.884
Power Grid	1.000	0.000	0.000
Communication	1.000	0.000	0.000
SCADA server	1.000	0.000	0.000
Distribution network 1	0.524	0.394	0.082
Distribution Pump	0.551	0.196	0.253
Distribution network 2	0.565	0.150	0.285

TABLE XV. Estimated Impact of R_3 on Assets

	state 1	state 2	state 3
Water Plant	0.032	0.492	0.476
Mountain Spring	0.000	0.000	1.000
Well	0.000	0.000	1.000
Well Pump	0.000	0.000	1.000
Water Reservoir	0.145	0.600	0.255
River	0.000	0.000	1.000
River Pump	0.000	0.000	1.000
Power Grid	1.000	0.000	0.000
Communication	1.000	0.000	0.000
SCADA server	1.000	0.000	0.000
Distribution network 1	0.361	0.525	0.114
Distribution Pump	0.388	0.262	0.350
Distribution network 2	0.406	0.185	0.409

TABLE XVI. Estimated Impact of R_4 on Assets

	state 1	state 2	state 3
Water Plant	0.070	0.305	0.625
Mountain Spring	1.000	0.000	0.000
Well	1.000	0.000	0.000
Well Pump	0.230	0.553	0.217
Water Reservoir	0.811	0.139	0.050
River	1.000	0.000	0.000
River Pump	0.230	0.565	0.205
Power Grid	1.000	0.000	0.000
Communication	0.000	0.000	1.000
SCADA server	0.230	0.487	0.283
Distribution network 1	0.346	0.507	0.147
Distribution Pump	0.385	0.250	0.365
Distribution network 2	0.432	0.149	0.419

hours), the reputation damage due to the incident (e.g., in case of insufficient protection against cyberattacks) or many more. Further, assessing a criticality level to each asset is possible to represent the importance of each asset for the overall process. Such criticality levels may also have different meaning for individual scenarios; e.g., if a pump or water tower fails for one day, this is more critical than a water contamination, since in the latter case, households can be advised to boil the water before drinking it, whereas a failure of the pump may completely cut off the household from water supply.

D. Risk Evaluation

In order to evaluate the different risks, consequences of the risks are compared. As we use an ordinal scale to measure the impact of each risk, the stochastic ordering mentioned in Section II-D simplifies to a lexicographic ordering with following interpretation. If one risk has a lower likelihood of worst case impact (state 3) than the other, we prefer this one; in case these are equal, the likelihoods of the second worst impact (state 2) are compared, and so on (and we randomly decide on the ordering of two risks with identical distributions).

Risk evaluation focuses on an asset of special interest (e.g., due to its vital importance to the CI) and is here illustrated with a focus on the water plant, comparing the impacts the all four identified risks on this asset. Figures 3, 4, 5 and 6 show the estimated impacts of the corresponding risks on the water plant.

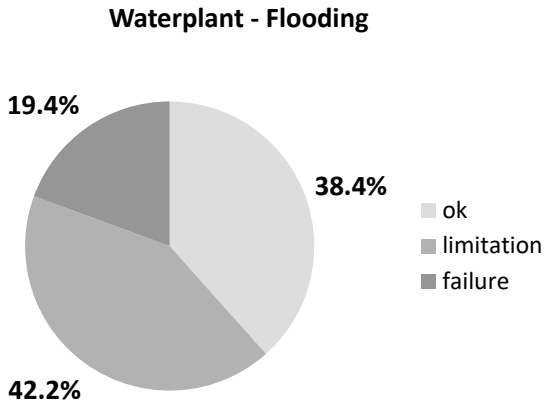


Fig. 3. Estimated Impact of R_1 on Water Plant

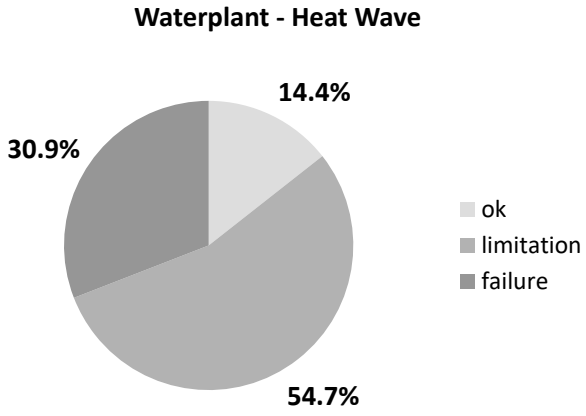


Fig. 4. Estimated Impact of R_2 on Water Plant

Comparing the probabilities we find that R_4 has the biggest chance of failure (62.5%), followed by R_3 (47.6%), R_2 (30.9%), and R_1 (19.4%). Thus, we think of R_4 as being the most dangerous one and R_1 as the one with the smallest chance of causing the most severe problems, i.e., we can write

$$R_1 \leq R_2 \leq R_3 \leq R_4.$$

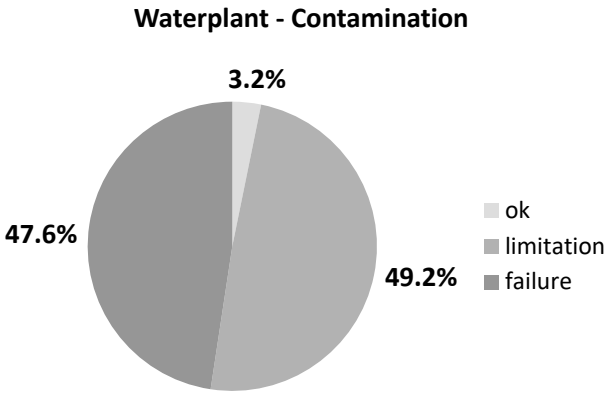


Fig. 5. Estimated Impact of R_3 on Water Plant

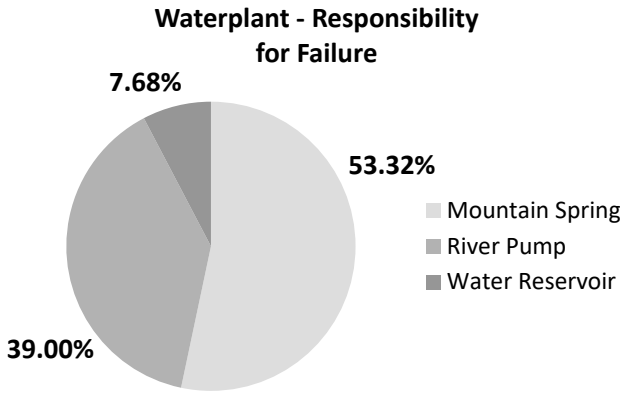


Fig. 7. Triggering factors for failure of the water plant during a contamination

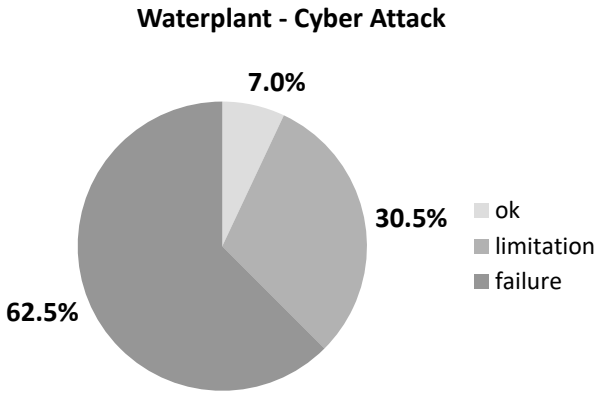


Fig. 6. Estimated Impact of R_4 on Water Plant

E. Risk Treatment

As the last step of the risk management process, risk treatment deals with the understanding of the relevant risks and their effects on the CI can (and should) be used to identify ways to mitigate the risks. This is what risk treatment is about as the last step in the risk management process. Several methods may be used here to identify controls that need to be implemented or improved but in any case it is helpful to know the *trigger* of the failure. Figure 7 shows a pie chart illustrating which assets were responsible for failure of the water plant during a contamination (i.e., a realisation of R_3).

The most frequent cause for failure of the water plant is clearly the mountain spring which is not surprising during a contamination if it happens near the mountain spring. This problem is not easily fixed (purification is an expensive and long-lasting process). In such a case it is faster to substitute water, e.g., through an agreement with other water providers to help out in such a critical situation. However, the analysis provides more information, it shows that also limited operation of the river pump as well as the water reservoir may lead to a failure of the water plant. While both problems are very likely to be also due to the contamination, it might sometimes be a bit easier to fix these indirect issues and to reduce the overall likelihood of failure due to a risk. Similarly, Figure 8 shows the different triggers for a limitation of the water plant in case of a contamination. The triggers are the same as in the case of failure but the mountain spring is now clearly the main source of problem while the water reservoir does not significantly trigger a limitation of the water plant.

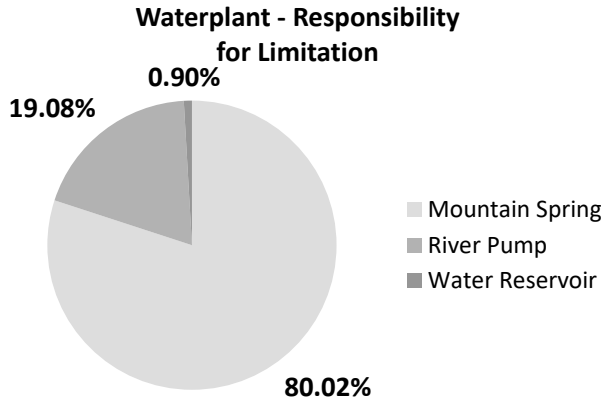


Fig. 8. Triggering factors for limitation of the Water Plant during a contamination

For a practitioner, the simulation's outcome is like a heat map, directly pointing out the most vulnerable spots in the network of critical infrastructures, which each CI domain expert can be informed about afterwards (see Figure 2 for an example scenario, from which each CI security officer can instantly see the degree of affection due to an incident). Towards a more fine-grained understanding of the affection's extent, the domain expert can continue by asking how likely an affection is to be medium or severe, which the pie charts

(see Figures 3 to 6) directly tell. Given this knowledge about the local affection, the expert may then strive for a root cause analysis, which the pie chart in Figures 7 and 8 help with: here, the domain expert gets the information of who is the most relevant "neighboring" CI that has the strongest impact on one's own CI. That is, if some CI C has relations on two other CIs A and B, an incident at A may be more or less severe than an incident happening at B. For example, Figure 7 explains that a failure of the water plant in a contamination scenario is most likely due to a problem with the mountain spring, or possibly also due to a problem with the river pump, but least likely, the cause is found at the water reservoir. This can be a guidance for fixing the problem in practice. Similarly, Figure 8 would advise the expert, upon an incident, to first look at the mountain spring as an external trigger of the local issue, but only in rare cases, the water reservoir will have caused the trouble.

IV. Conclusion

Applying simulation is a straightforward proposal to extend the understanding of how security incidents propagate through and affect one or more critical infrastructures. Setting up proper simulation models, and using the information that these deliver is, however, a different story with its own challenges. This work used a hypothetical water provider in the background to describe a step by step method of

(i) how to specify interdependencies between critical infrastructures in a way that allows domain experts to include their subjective uncertainty,
(ii) how the data and specification for a concrete simulation model (chosen here for illustration) could look like, and
(iii) how the simulation model's results could be compiled into a digestible form to ease decision making by revealing previously unexpected roles of assets in incident propagation and loss estimation.

Open issues includes accuracy assessments of such a simulation (relative to reported incidents in real life), but equally important, a study of usability from domain experts perspectives. Having an accurate model is not enough, unless people outside the scientific realm and concerned with the practical things feel capable of using it. The "tutorial" style of this work shall be a step towards bridging this gap.

References

[1] S. König, T. Grafenauer, S. Rass, and S. Schauer, "Practical risk analysis in interdependent critical infrastructures - a How-To," in *The Twelfth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE) 2018*. IARIA XPS Press, 2018, pp. 150–157.

[2] T. Grafenauer, S. König, S. Rass, and S. Schauer, "A simulation tool for cascading effects in interdependent critical infrastructures," in *International Workshop on Security Engineering for Cloud Computing (IWSECC 2018), collocated with the 13th International Conference on Availability, Reliability and Security (ARES 2018)*, 2018, (in press).

[3] Y. Y. Haimes, "Hierarchical Holographic Modeling," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 9, pp. 606–617, 1981.

[4] N. K. Svendsen and S. D. Wolthusen, "Analysis and statistical properties of critical infrastructure interdependency multiflow models," in *2007 IEEE SMC Information Assurance and Security Workshop*, June 2007, pp. 247–254.

[5] R. Setola, S. D. Porcellinis, and M. Sforna, "Critical infrastructure dependency assessment using the input-output inoperability model," *International Journal of Critical Infrastructure Protection (IJCIP)*, vol. 2, pp. 170–178, 2009.

[6] S.-J. Wu and M. T. Chu, "Markov chains with memory, tensor formulation, and the dynamics of power iteration," vol. 303, pp. 226–239, 2017.

[7] M. Rahnamay-Naeini and M. M. Hayat, "Cascading failures in interdependent infrastructures: An interdependent markov-chain approach," *IEEE Transactions on Smart Grid*.

[8] S. König and S. Rass, "Stochastic dependencies between critical infrastructures," in *SECURWARE 2017: The Eleventh International Conference on Emerging Security Information, Systems and Technologies*. IARIA, 2017, pp. 106–110.

[9] S. König, S. Schauer, and S. Rass, "A stochastic framework for prediction of malware spreading in heterogeneous networks," in *Secure IT Systems: 21st Nordic Conference, NordSec 2016, Oulu, Finland, November 2-4, 2016. Proceedings*, B. B. Brumley and J. Röning, Eds. Springer International Publishing, 2016, pp. 67–81.

[10] A. V. Uzunov, E. B. Fernandez, and K. Falkner, "Securing distributed systems using patterns: A survey," *Computers & Security*, vol. 31, no. 5, pp. 681–703, 2012.

[11] S. Fletcher, "Electric power interruptions curtail California oil and gas production," *Oil Gas Journal*, 2001.

[12] M. Schmidthaler and J. Reichl, "Economic Valuation of Electricity Supply Security: Ad-hoc Cost Assessment Tool for Power Outages," *ELECTRA*, no. 276, pp. 10–15, 2014.

[13] J. Condliffe, "Ukraine's Power Grid Gets Hacked Again, a Worrying Sign for Infrastructure Attacks," 2016. [Online]. Available: https://www.technologyreview.com/s/603262/ukraines-power-grid-gets-hacked-again-a-worrying-sign-for-infrastructure-attacks/

[14] D. Kushner, "The real story of stuxnet," *IEEE Spectrum*, vol. 3, no. 50, pp. 48–53, 2013.

[15] National Audit Office, "Investigation: Wannacry cyber attack and the nhs," 2017. [Online]. Available: https://www.nao.org.uk/wp-content/uploads/2017/10/Investigation-WannaCry-cyber-attack-and-the-NHS.pdf

[16] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[17] Z. Bederna and T. Szadeczky, "Cyber espionage through botnets," *Security Journal*, Sep 2019.

[18] European Parliament, "Directive (EU) 2016/1148 of the European Parliament and of the Council: concerning measures for a high common level of security of network and information systems across the Union," *Official Journal of the European Union*, 2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016L1148&from=EN

[19] S. Rinaldi, J. Peerenboom, and T. Kelly, "Identifying, understanding, and analyzing critical infrastructure interdependencies," *IEEE Control Systems Magazine*, pp. 11–25, 2001.

[20] A. Gouglidis, B. Green, J. Busby, M. Rouncefield, D. Hutchison, and S. Schauer, *Threat awareness for critical infrastructures resilience*. IEEE, 9 2016.

[21] S. M. Rinaldi, "Modeling and simulating critical infrastructures and their interdependencies," in *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*. IEEE, 2004, p. 8 pp.

[22] H. Guo, C. Zheng, H. H.-C. Iu, and T. Fernando, "A critical review of cascading failure analysis and modeling of power system," *Renewable and Sustainable Energy Reviews*, vol. 80, pp. 9–22, dec 2017.

[23] S. Saidi, L. Kattan, P. Jayasinghe, P. Hettiaratchi, and J. Taron, "Integrated infrastructure systems—A review," *Sustainable Cities and Society*, vol. 36, pp. 1–11, Jan 2018.

[24] E. Luiijf, M. Ali, and A. Zielstra, "Assessing and improving SCADA security in the Dutch drinking water sector," *International Journal of Critical Infrastructure Protection*, vol. 4, no. 3-4, pp. 124–134, 2011.

[25] A. Alshawish, M. A. Abid, H. de Meer, S. Schauer, S. König, A. Gouglidis, and D. Hutchison, "G-dps: A game-theoretical decision-making framework for physical surveillance games," in *Game Theory for Security and Risk Management: From Theory to Practice*. Cham: Springer International Publishing, 2018, pp. 129–156.

[26] A. Gouglidis, S. König, B. Green, K. Rossegger, and D. Hutchison, "Protecting water utility networks from advanced persistent threats: A case study," in *Game Theory for Security and Risk Management: From*

Theory to Practice. Cham: Springer International Publishing, 2018, pp. 313–333.

[27] European Comission, "COUNCIL DIRECTIVE 2008/114/EC of 8 December 2008 on the identification and designation of European critical infrastructures and the assessment of the need to improve their protection," *Official Journal of the European Union*, no. L345, pp. 75–82, 2008.

[28] ISO International Organization for Standardization, *ISO 31000:2018 Risk management - Guidelines*. Geneva, Switzerland: ISO International Organization for Standardization, 2018.

[29] National Institute of Standarts and Technology (NIST), "National Vulnerability Database (NVD)," https://nvd.nist.gov/, accessed: 2019-12-05.

[30] "Nessus vulnerability scanner," https://www.tenable.com/products/nessus-vulnerability-scanner, accessed: 2019-08-22.

[31] "OpenVAS: Open Vulnerability Assessment System," http://www.openvas.org/, accessed: 2019-08-22.

[32] I. Münch, "Wege zur Risikobewertung," in *DACH Security 2012*, P. Schartner and J. Taeger, Eds. syssec, 2012, pp. 326–337.

[33] S. Schauer, "A risk management approach for highly interconnected networks," in *Game Theory for Security and Risk Management*. Springer International Publishing, 2018, pp. 285–311.

[34] S. Rass, S. König, and S. Schauer, "Decisions with uncertain consequences—a total ordering on loss-distributions," vol. 11, no. 12, p. e0168583.

[35] ——, "Defending against advanced persistent threats using game-theory," vol. 12, no. 1, p. e0168675. [Online]. Available: http://dx.plos.org/10.1371/journal.pone.0168675

[36] S. Rass, S. König, and S. Schauer, "Uncertainty in games: Using probability-distributions as payoffs," in *Lecture Notes in Computer Science*. Springer International Publishing, 2015, pp. 346–357.

[37] S. Rass and S. König, "Package 'HyRiM': Multicriteria risk management using zero-sum games with vector-valued payoffs that are probability distributions," 2019. [Online]. Available: https://cran.r-project.org/web/packages/HyRiM/index.html

[38] S. König, "Choosing ways to increase resilience in critical infrastructures," in *Proceedings of the 16th ISCRAM Conference – Valencia, Spain May 2019*, Valencia, 2019, pp. 1245–1251.

[39] Zurich, "Flash floods: The underestimated natural hazard," Zurich Insurance Company Ltd, Tech. Rep., 2017, accessed: 2019-08-22.

[40] J. H. Christensen and O. B. Christensen, "Severe summertime flooding in europe," *Nature*, vol. 421, no. 6925, pp. 805–806, 2003.

[41] A. H. Thieken, T. Bessel, S. Kienzler, H. Kreibich, M. Müller, S. Pisi, and K. Schröter, "The flood of june 2013 in Germany: how much do we know about its impacts?" *Natural Hazards and Earth System Sciences*, vol. 16, no. 6, pp. 1519–1540, 2016.

[42] K. Schröter, M. Kunz, F. Elmer, B. Mühr, and B. Merz, "What made the June 2013 flood in Germany an exceptional event? a hydro-meteorological evolution," *Hydrology and Earth System Sciences*, vol. 19, pp. 309–327, 2015.

[43] Z. W. Kundzewicz, U. Ulbrich, T. brücher, D. Graczyk, A. Krüger, G. C. Leckebusch, L. Menzel, I. Pińskwar, M. Radziejewski, and M. Szwed, "Summer floods in central europe – climate change track?" *Natural Hazards*, vol. 36, no. 1-2, pp. 165–189, 2005.

[44] J. A. Brown and W. P. Darby, "Predicting the probability of contamination at groundwater based public drinking supplies," vol. 11, pp. 1077–1082, 1988.

[45] C. Cimpanu. (2018) Port of San Diego suffers cyber-attack, second port in a week after Barcelona. [Online]. Available: https://www.zdnet.com/article/port-of-san-diego-suffers-cyber-attack-second-port-in-a-week-after-barcelona/

[46] (2018) 2018 highlights: Major cyber attacks reported in maritime industry. [Online]. Available: https://safety4sea.com/cm-2018-highlights-major-cyber-attacks-reported-in-maritime-industry/

[47] World Maritime News. (2018) COSCO shipping lines falls victim to cyber attack. [Online]. Available: https://worldmaritimenews.com/archives/257665/cosco-shipping-lines-falls-victim-to-cyber-attack/

[48] A. Greenerg. The untold story of NotPetya, the most devastating cyberattck in history. Accessed: 2019-08-23. [Online]. Available: https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/

An Advanced Approach for Choosing Security Patterns and Checking their Implementation

Sébastien Salva*, Loukmen Regainia†
LIMOS - UMR CNRS 6158
University Clermont Auvergne, France
*Email: * sebastien.salva@uca.fr, † loukmen.regainia@uca.fr*

***Abstract*—This paper tackles the problems of generating concrete test cases for testing whether an application is vulnerable to attacks, and of checking whether security solutions are correctly implemented. The approach proposed in the paper aims at guiding developers towards the implementation of secure applications, from the threat modelling stage up to the testing one. This approach relies on a knowledge base integrating varied security data, e.g., attacks, attack steps, and security patterns that are generic and re-usable solutions to design secure applications. The first stage of the approach consists in assisting developers in the design of Attack Defense Trees expressing the attacker possibilities to compromise an application and the defenses that may be implemented. These defenses are given under the form of security pattern combinations. In the second stage, these trees are used to guide developers in the test case generation. After the test case execution, test verdicts show whether an application is vulnerable to the threats modelled by an ADTree. The last stage of the approach checks whether behavioural properties of security patterns hold in the application traces collected while the test case execution. These properties are formalised with LTL properties, which are generated from the knowledge base. Developers do not have to write LTL properties not to be expert in formal models. We experimented the approach on 10 Web applications to evaluate its testing effectiveness and its performance.**

Keywords-Security Pattern; Security Testing; Attack-Defense Tree; Test Case Generation.

I. Introduction

Today's developers are no longer just expected to code and build applications. They also have to ensure that applications meet minimum reliability guarantees and security requirements. Unfortunately, choosing security solutions or testing software security are not known to be simple or effortless activities. Developers are indeed overloaded of new trends, frameworks, security issues, documents, etc. Furthermore, they sometimes lack skills and experience for choosing security solutions or writing concrete test cases. They need to be guided on how to design or implement secure applications and test them, in order to contribute in a solid quality assurance process.

This work focuses on this need and proposes an approach that guides developers devise more secure applications from the threat modelling stage, which is a process consisting in identifying the potential threats of an application, up to the testing one. The present paper is an extended version of [1], which provides additional details on the security test case generation, the formalisation of behavioural properties of security patterns with Linear Temporal Logic (LTL) properties, and on their automatic generation. We also provide an evaluation of the approach and discuss the threats to validity.

In order to guide developers, our approach is based upon several several digitalised security bases or documents gathered in a knowledge base. In particular, the latter includes security solutions under the form of security patterns, which can be chosen and applied as soon as the application design. Security patterns are defined as *reusable elements to design secure applications, which will enable software architects and designers to produce a system that meets their security requirements and that is maintainable and extensible from the smallest to the largest systems* [2]. Our approach helps developers chose security patterns with regard to given security threats. Then, it builds security test cases to check whether an application is vulnerable, and test whether security patterns are correctly implemented in the application. More precisely, the contributions of this work are summarised in the following points:

- the approach assists developers in the threat modelling stage by helping in the generation of Attack Defense Trees (ADTrees) [3]. The latter express the attacker possibilities to compromise an application, and give the defenses that may be put in place to prevent attacks. Defenses are here expressed with security patterns. We have chosen this tree model because it offers the advantage of being easy to understand even for novices in security;
- the second part of the approach supports developers in writing concrete security test cases. A test suite is automatically extracted from an ADTree. The test suite is made up of test case stubs, which are completed with comments or blocs of code. Once completed, these are used to experiment an application under test (shortened AUT), seen as a black-box. The test case execution provides verdicts expressing whether the AUT is vulnerable to the threats modelled in the ADTree;
- the last part of the approach allows developers to check whether security patterns are correctly implemented in

the application. Kobashi et al. dealt with this task by asking users to manually translate security pattern behaviours into formal properties [4]. Unfortunately, few developers have the required skills in formal modelling. We hence prefer proposing a practical way to generate them. After the security pattern choice, our approach provides generic UML sequence diagrams, which can be adapted to better match the application context. From these diagrams, the approach automatically generate LTL properties. After the test case execution, we check if these properties hold in the application traces. The developer is hence not aware of the LTL property generation.

We have implemented this approach in a tool prototype available in [5]. This tool was used to conduct several experiments on 10 Web applications to evaluate the security testing and security pattern testing effectiveness of the tool as well as its performance.

Paper Organisation: Section II outlines the context of this work. We recall some basic concepts and notations about security patterns and ADTrees. We also discuss about the related work and our motivations. Section III briefly presents the architecture of the knowledge base used by our approach. The approach steps are described in Section IV. These steps are gathered into 3 stages called threat modelling, security testing, and security pattern testing. Subsequently, Section V describes our prototype implementation, and Section VI evaluates the approach. Finally, Section VII summarizes our contributions and presents future work.

II. Background

This section recalls the basic concepts related to security patterns and Attack Defense trees. The related work is presented thereafter.

A. Security Patterns

Security patterns provide guidelines for secure system design and evaluation [6]. They also are considered as countermeasures to threats and attacks [7]. Security patterns have to be selected in the design stage, integrated in application models, and eventually implemented. Their descriptions are usually given with texts or schema. But, they are often characterised by UML diagrams capturing structural or behavioural properties.

Several security pattern catalogues are available in the literature, e.g., [8], [9], themselves extracted from other papers. In these catalogues, security patterns are systematically organised according to features and relationships among them. Among these features, we often find the solutions called intents, or the interests called forces. A security pattern may have different relationships with other patterns. These relations may noticeably help combine patterns together and not to devise unsound composite patterns. Yskout et al. proposed the following annotations between two patterns [10]: “depend”, “benefit”, “impair” (the functioning of the pattern can be obstructed by the implementation of a second one), “alternative”, “conflict”.

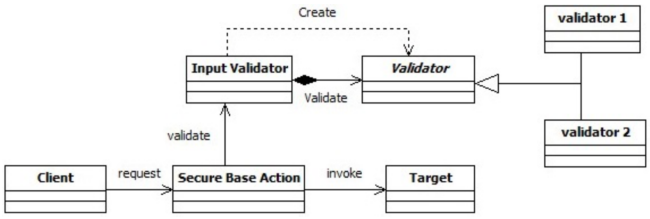


Figure 1. Class layout of the security pattern “Intercepting Validator”.

Figure 1 depicts the UML structural diagram of the security pattern “Intercepting Validator”, which is taken as example is the remainder of the paper. Its purpose is to provide the application with a centralized validation mechanism, which applies some filters (Validator classes) declaratively based on URL, allowing different requests to be mapped to different filter chains. This validation mechanism is decoupled from the other parts of the application and each data supplied by the client is validated before being used. The validation of input data prevents attackers from passing malformed input in order to inject malicious commands.

B. Attack Defense Trees

ADTrees *are graphical representations of possible measures an attacker might take in order to compromise a system and the defenses that a defender may employ to protect the system* [3]. ADTrees have two different kinds of nodes: attack nodes (red circles) and defense nodes (green squares). A node can be *refined* with child nodes and can have one child of the opposite type (linked with a dashed line). Node refinements can be disjunctive or conjunctive. The former is recognisable by edges going from a node to its children. The latter is graphically distinguishable by connecting these edges with an arc. We extend these two refinements with the sequential conjunctive refinement of attack nodes, defined by the same authors in [11]. This operator expresses the execution order of child attack nodes. Graphically, a sequential conjunctive refinement is depicted by connecting the edges, going from a node to its children, with an arrow.

For instance, the ADTree of Figure 2 identifies the objectives of an attacker or the possible vulnerabilities related to the supply of untrusted inputs to an application. The root node is here detailed with disjunctive refinements connecting three leaves, which are labelled by attack referenced in a base called the Common Attack Pattern Enumeration and Classification (CAPEC) [12]. The node CAPEC-66 refers to “SQL Injection”, CAPEC-250 refers to XML injections and CAPEC-244 to “Cross-Site Scripting via Encoded URI Schemes”.

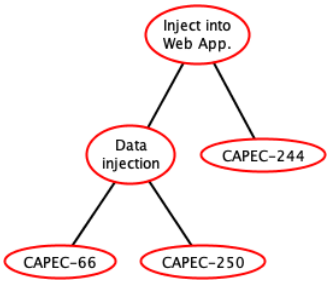


Figure 2. ADTree example modelling injection attacks

An ADTree T can be formulated with an algebraic expression called ADTerm and denoted $\iota(T)$. In short, the ADTerm syntax is composed of operators having types given as exponents in $\{o, p\}$ with o modelling an opponent and p a proponent. $\vee^s, \wedge^s, \overrightarrow{\wedge}^s$, with $s \in \{o, p\}$ respectively stand for the disjunctive refinement, the conjunctive refinement and the sequential conjunctive refinement of a node. A last operator c expresses counteractions (dashed lines in the graphical tree). $c^s(a, d)$ intuitively means that there exists an action d (not of type s) that counteracts the action a (of type s). The ADTree of Figure 2 can be represented with the ADTerm $\vee^p(\vee^p(\text{CAPEC-66, CAPEC-250}), \text{CAPEC-244})$.

C. Related Work

The literature proposes several papers dealing with the test case generation from Attack trees (or related models) and some other ones about security pattern testing. As these topics are related to our work, we introduce them below and give some observations.

1) Security Testing From Threat Models: the generation of concrete test cases from models has been widely studied in the last decade, in particular to test the security level of different kinds of systems, protocols or software. Most of the proposed approaches take specifications expressing the expected behaviours of the implementation. But, other authors preferred to bring security aspects out and used models describing attacker goals or vulnerability causes of the system. Such models are conceived during the threat modelling phase of the system [13], which is considered as a critical phase of the software life cycle since *"you cannot build a secure system until you understand your threats!"* [14]. Schieferdecker et al. presented a survey paper referencing some approaches in this area [15]. For instance, Xu et al. proposed to test the security of Web applications with models as Petri nets to describe attacks [16]. Attack scenarios are extracted from the Reachability graphs of the Petri nets. Then, test cases written for the Selenium tool are generated by means of a MIM (Model- Implementation Mapping) description, which maps each Petri net place and transition to a block of code. Bozic et al. proposed a security testing approach associating UML state diagrams to represent attacks, and combinatorial testing to generate input values used to make executable test cases derived from UML models [17].

Other authors adopted models as trees (Attack trees, vulnerability Cause Graphs, Security Activity Graphs, etc.) to represent the threats, attacks or vulnerability causes that should be prevented in an application. From these models, test cases are then written to check whether attacks can be successfully executed or whether vulnerabilities are detected in the implementation. Morai et al. introduced a security testing approach specialised for network protocols [18]. Attack scenarios are extracted from an Attack tree and are converted to Attack patterns and UML specifications. From these, attack scripts are manually written and are completed with the injection of (network) faults. In the security testing method proposed in [19], data flow diagrams are converted into Attack trees from which sequences are extracted. These sequences are composed of events combined with parameters related to regular expressions. These events are then replaced with blocks of code to produce test cases. The work published in [20] provides a manual process composed of eight steps. Given an Attack tree, these steps transform it into a State chart model, which is iteratively completed and transformed before using a model-based testing technique to generate test cases. In [21], test cases are generated from Threat trees. The latter are previously completed with parameters associated to regular expressions to generate input values. Security scenarios are extracted from the Threat trees and are manually converted to executable test scripts. Shahmehri et al. proposed a passive testing approach, which monitors an AUT to detect vulnerabilities [22]. The undesired vulnerabilities are modelled with security goal models, which are specialised directed acyclic graphs showing security goals, vulnerabilities and eventually mitigations. Detection conditions are then semi-automatically extracted and given to a monitoring tool.

We observed that the above methods either automatically generate abstract test cases from (formal) specifications or help write concrete test cases from detailed threat models. On the one hand, as abstract test cases cannot be directly used to experiment an AUT, some works proposed test case transformation techniques. However, this kind of technique is at the moment very limited. On the other hand, Only a few of developers have the required skills to write threat models or test cases, as a strong expertise on security is often required. Besides, the methods neither guide developers in the threat modelling phase nor provide any security solution. We focused on this problem and laid the first stone of the present approach in [23], [24], [25]. We firstly presented a semi-automatic data integration method [23] to build security pattern classifications. This method extracts security data from various Web and publicly accessible sources and stores relationships among attacks, security principles and security patterns into a knowledge base. Section III summarises the results of this work used in this paper, i.e.,

the first meta-model version of the data-store. In [24], we proposed an approach to help developers write ADTrees and concrete security test cases to check whether an application is vulnerable to these attacks. This work was extended in [25] to support the generation of test suites composed of lists of ordered GWT test cases, a list being devoted to check whether an AUT is vulnerable to an attack, which is segmented into an ordered sequence of attack steps. This test suite organisation is used to reduce the test costs with the deduction of some test verdicts under certain conditions. However, it does not assist developers to ensure that security patterns have been correctly implemented in the application. This work supplements our early study by covering this part.

2) Security Pattern Testing: the verification of patterns on models was studied in [26], [27], [28], [4], [29]. In these papers, pattern goals or intents or structural properties are specified with UML sequence diagrams [26] with expressions written with the Object Constraint Language (OCL) [27], [28], [4] or with LTL properties [29]. The pattern features are then checked on UML models.

Few works dealt with the testing of security patterns, which is the main topic of this paper. Yoshizawa et al. introduced a method for testing whether behavioural and structural properties of patterns may be observed in application traces [28]. Given a security pattern, two test templates (Object Constraint Language (OCL) expressions) are manually written, one to specify the pattern structure and another one to encode its behaviour. Then, developers have to make templates concrete by manually writing tests for experimenting the application. The latter returns traces on which the OCL expressions are verified.

We observed that these previous works require the modelling of security patterns or vulnerabilities with formal properties. Instead of assuming that developers are expert in the writing of formal properties, we propose a practical way to generate them. Intuitively, after the choice of security patterns, our approach provides generic UML sequence diagrams, which can be modified by a developer. From these diagrams, we automatically generate LTL properties, which capture the cause-effects relations among pairs of method calls. After the test case execution, we check if these properties hold in the application traces, obtained while the test case execution. The developer is hence not aware of the LTL property generation. As stated in the introduction, this work provides more details on test case generation and on the formalisation of behavioural properties of security patterns with LTL properties. We also complete the transformation rules allowing to derive more LTL properties from UML sequence diagrams. We also provide an evaluation of the approach targeting the security pattern testing stage and discuss the threats to validity.

III. Knowledge Base Overview

Our approach relies on a knowledge base, denoted KB in the remainder of the paper. It gathers information allowing to help or automate some steps of the testing process. We summarise its architecture in this section but we refer to [23] for a complete description of its associations and of the data integration process.

A. Knowledge Base Meta-Model

Figure 3 exposes the meta-model used to structure the knowledge base KB. The entities refer to security properties and the relations encode associations among them. The entities in white are used to generated ADTrees, while those in grey are specialised for testing. The meta-model firstly associates attacks, techniques, security principles and security patterns. This is the result of observations we made from the literature and some security documents, e.g., the CAPEC base or security pattern catalogues [8], [9]: we consider that an attack can be documented with more concrete attacks, which can be segmented into ordered steps; an attack step provides information about the target or puts an application into a state, which are reused by a potential next step. Attack steps are performed with techniques and can be prevented with countermeasures. Security patterns are characterised with strong points, which are pattern features extractable from their descriptions. The meta-model also captures the inter-pattern relationships defined in [10], e.g., ”depend” or ”conflict”. Countermeasures and strong points refer to the same notion of attack prevention. But finding direct relations between countermeasures and strong points is tedious as these properties have different purposes. To solve this issue, we used a text mining and a clustering technique to group the countermeasures that refer to the same security principles, which are desirable security properties. To link clusters and strong points, we chose to focus on these security principles as mediators. We organised security principles into a hierarchy, from the most abstract to the most concrete principles. We provide a complete description of this hierarchy in [24]. In short, we collected and organised 66 security principles covering the security patterns of the catalogue given in [9]. The hierarchy has four levels, the first one being composed of elements labelled by the most abstract principles, e.g., “Access Control”, and the lower level exhibiting the most concrete principles, e.g., “File Authorization”.

Furthermore, every attack step is associated to one test case structured with the Given When Then (GWT) pattern. We indeed consider in this paper that a test case is a piece of code that lists stimuli supplied to an AUT and responses checked by assertions assigning (local) verdicts. To make test cases readable and re-usable, we use the behaviour driven approach using the pattern “Given When Then” (shortened GWT) to break up test cases into several sections:

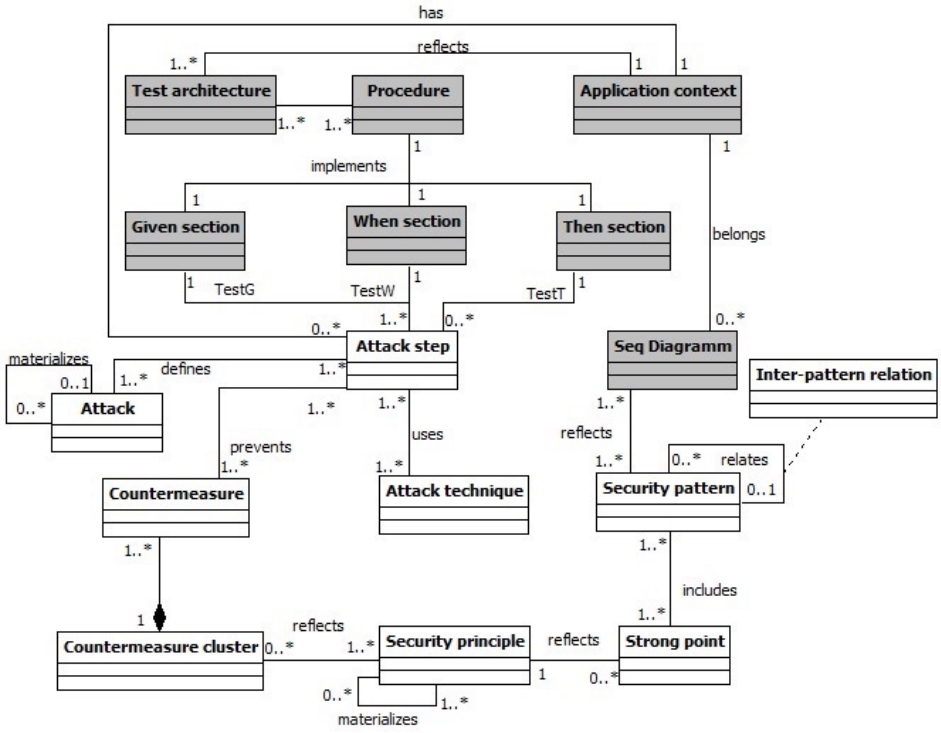


Figure 3. Data-store meta-model

- Given sections aim at putting the AUT into a known state;
- When sections trigger some actions (stimuli);
- Then sections are used to check whether the conditions of success of the test case are met with assertions. In the paper, the Then sections are used to check whether an AUT is vulnerable to an attack step st. In this case, the Then section returns the verdict "$Pass_{st}$". Otherwise, it provides the verdict "$Fail_{st}$". When a unexpected event occurs, we also assume that "$Inconclusive_{st}$" may be returned.

The meta-model of Figure 3 associates an attack step with a GWT test case by adding three entities (Given When and Then section) and relations. In addition, a test case section is linked to one procedure, which implements it. A section or a procedure can be reused with several attack steps or security patterns. The meta-model also reflects the fact that an attack step is associated with one "Test architecture" and with one "Application context". The former refers to textual paragraphs explaining the points of observation and control, testers or tools required to execute the attack step on an AUT. An application context refers to a family, e.g., Android applications, or Web sites. As a consequence, a GWT test case section (and procedure) is classified according to one application context and one attack step or pattern consequence.

We finally updated the meta-model in such a way that a security pattern is also associated to generic UML sequence diagrams, themselves arranged in Application contexts. Security pattern catalogues often provide UML sequence diagrams expressing the security pattern behaviours or structures. These diagrams often help correctly implement a security pattern with regard to an application context.

B. Data Integration

We integrated data into KB by collecting them from heterogeneous sources: the CAPEC base, several papers dealing with security principles [30], [31], [32], [33], [34], the pattern catalogue given in [35] and the inter-pattern relations given in [10]. We details the data acquisition and integration steps in [23]. Six manual or automatic steps are required: Steps 1 to 5 give birth to databases that store security properties and establishing the different relations presented in Figure 3. Step 6 consolidates them so that every entity of the meta-model is related to the other ones as expected. The steps 1,2 and 6 are automatically done with tools.

The current knowledge base KB includes information about 215 attacks (209 attack steps, 448 techniques), 26 security patterns, 66 security principles. We also generated 627 GWT test case sections (Given, When and Then sections) and 209 procedures. The latter are composed of comments explaining: which techniques can be used to execute an attack step and which observations reveal that the application is vulnerable. We manually completed 32 procedures, which

cover 43 attack steps. Security patterns are associated to at least one UML diagram. This knowledge base is available in [5].

It is worth noting that KB can be semi-automatically updated if new security data are available. If a new threat or type of attack is discovered and added to the CAPEC base, the steps 1, 2 and 5 have to be followed again. Likewise, if a new security pattern is proposed in the literature, the steps 3,4 and 5 have to be reapplied.

IV. Security Testing and Security Pattern Verification

A. Approach Overview

We present in this section our testing approach whose steps are illustrated in Figure 4. As illustrated in the figure, the purpose of this approach is threefold:

1) **Threat modelling:** it firstly aims at guiding developers through the elaboration of a threat model (left side of the figure). The developer gives an initial ADTree expressing attacker capabilities (Step 1). By means of KB, this tree is automatically detailed and completed with security patterns combinations expressing security solutions that may be put in place in the application design (Step 2). The tree may be modified to match the developer wishes (Step 3). The resulting ADTree, which is denoted T_f, captures possible attack scenarios and countermeasures given under the form of security pattern combinations. The set of security patterns chosen by the developer is denoted $SP(T_f)$.
2) **Security testing:** from T_f, the approach generates test case stubs, which are structured with the GWT pattern (Step 4). These stubs guide developers in the writing of concrete test cases (Step 8). The final test suite is executed on the AUT to check whether the AUT is vulnerable to the attack scenarios expressed in the ADTree T_f (Step 9).
3) **Security pattern verification:** the last part of the approach is devoted to checking whether security pattern behaviours hold in the AUT traces. A set of generic UML sequence diagrams are extracted, from KB, for every security pattern in $SP(T_f)$ (Step 5). These show how security patterns classes or components should behave and help developers implement them in the application. These diagrams are usually adapted to match the application context (Step 6). The approach skims the UML sequence diagrams and automatically generates LTL properties encoding behavioural properties of the security patterns (Step 7). While the test case execution, the approach collects the AUT method-call traces on which it checks whether the LTL properties are satisfied (Step 10).

The remaining of this section describes more formally the steps depicted in Figure 4.

B. Threat Modelling, Security Pattern Choice (Step 1 to 3)

Step 1: Initial ADTree Design

The developer draws a first ADTree T whose root node represents some attacker's goals. This node may be refined with several layers of children to refine these goals. Different methods can be followed, e.g., DREAD [36], to build this threat model. We here assume that the leaves of this ADTree are exclusively labelled by CAPEC attack identifiers, since our knowledge base KB is framed upon the CAPEC base. Figure 2 illustrates an example ADTree achieved for this step. The leaves of this tree are labelled by CAPEC attacks related to different kinds of injection-based attacks. Its describes in general terms attacker goals, but this model is not sufficiently detailed to generate test cases or to choose security solutions.

Step 2: ADTree Generation

KB is now queried to complete T with more details about the attack execution phase and with defense nodes labelled by security patterns. For every leave of T labelled by an attack A, an ADTree $T(A)$, is generated from KB. We refer to [24] for the description of the ADTree generation.

We have implemented the ADTree generation with a tool, which takes attacks of KB and yields XML files. These can be edited with the tool *ADTool* [3]. For instance, Figures 6 and 7 show the ADTrees generated for the attacks CAPEC-66 and CAPEC-244. The ADTrees generated by this step are composed of several levels of attacks, having different levels of abstraction. The attack steps have child nodes referring to attack techniques, which indicate how to carry out the step. For instance the technique 1.1.1 is "Use a spidering tool to follow and record all links and analyze the web pages to find entry points. Make special note of any links that include parameters in the URL". An attack step node is also linked to a defense node expressing security pattern combinations. Some nodes express inter-pattern relations. For instance, the node labelled by "Alternative" has children expressing several possible patterns to counter the attack step.

Figures 6 and 7 also reveal that our generated ADTrees follow the structure of our meta-model of Figure 3. This structure has the generic form given in Figure 5: ADTrees have a root attack node, which may be disjunctively refined with other attacks and so forth. The most concrete attack nodes are linked to defense nodes labelled by security patterns. We formulate in the next proposition that these nodes or sub-trees also are encoded with specific ADTerms, which shall be used for the test case generation:

Proposition 1 *An ADTree $T(A)$ achieved by the previous steps has an ADTerm $\iota(T(A))$ having one of these forms:*

1) *$\vee^p(t_1, \ldots, t_n)$ with $t_i(1 \leq i \leq n)$ an ADTerm also having one of these forms:*
2) *$\overrightarrow{\wedge}^p(t_1, \ldots, t_n)$ with $t_i(1 \leq i \leq n)$ an ADTerm having the form given in 2) or 3);*

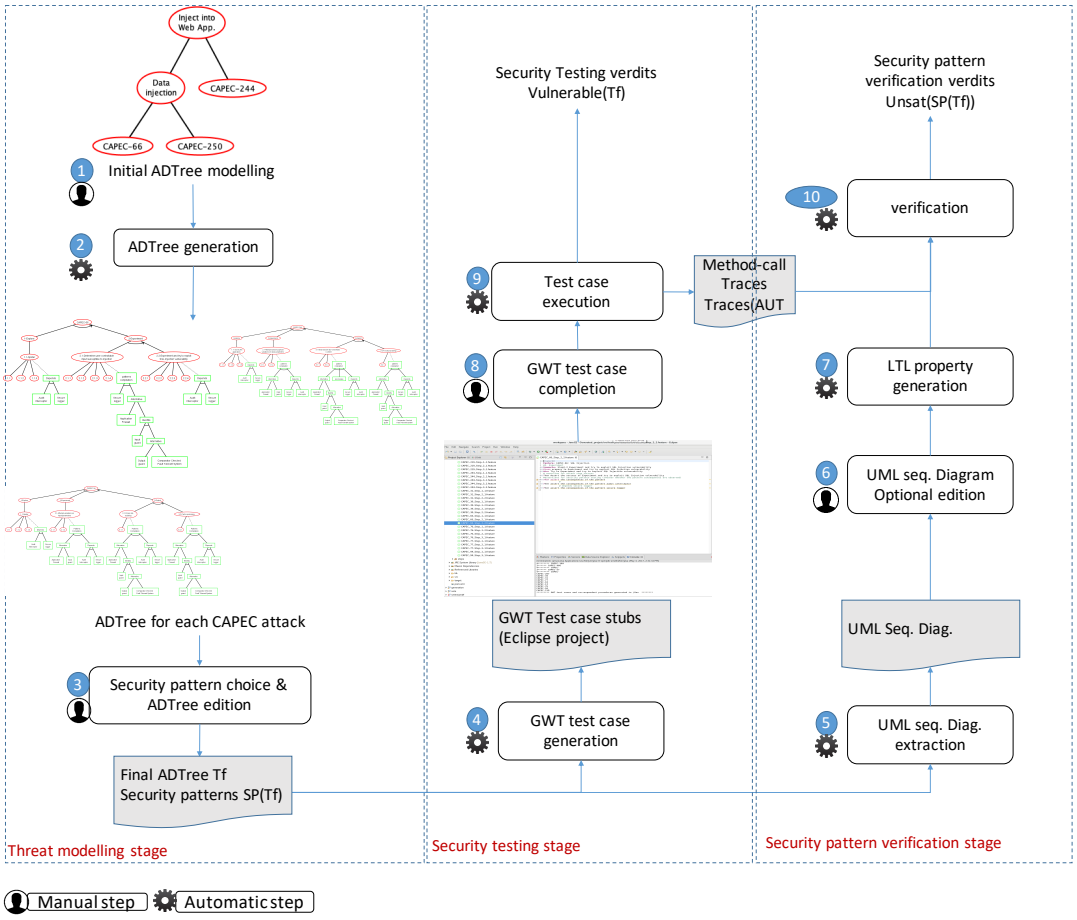


Figure 4. Overview of the 10 steps of the approach

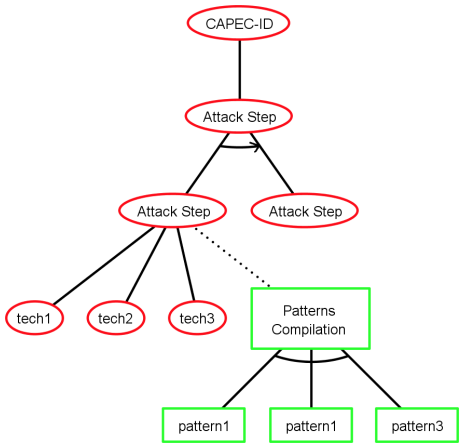


Figure 5. Genenal form of the generated ADTrees

3) $c^p(st, sp)$, *with* st *an ADTerm expressing an attack step and* sp *an ADTerm modelling a security pattern combination.*

The first ADTerm expresses child nodes labelled by more concrete attacks. The second one represents sequences of attack steps. The last ADTerm is composed of an attack step st refined with techniques, which can be counteracted by a security pattern combination $sp = \wedge^o(sp_1, \ldots, sp_m)$. In the remainder of the paper, we denote the last expression $c^p(st, sp)$ a *Basic Attack Defence Step*, shortened as BADStep:

Definition 2 (Basic Attack Defence Step (BADStep)) *A BADStep* b *is an ADTerm of the form* $c^p(st, sp)$*, where* st *is a step only refined with techniques and* sp *an ADTerm of the form:*

1) sp_1*, with* sp_1 *a security pattern,*
2) $\wedge^o(sp_1, \ldots, sp_m)$ *modelling the conjunction of the security patterns* $sp_1, \ldots, sp_m (m > 1)$.

$\text{defense}(b) =_{def} \{sp_1\}$ *iff* $sp = sp_1$*, or* $\text{defense}(b) =_{def} \{sp_1, \ldots, sp_m\}$ *iff* $sp = \wedge^o(sp_1, \ldots, sp_m)$.
$\text{BADStep}(T)$ *denotes the set of BADSteps of the ADTree* T.

Step 3: Security Pattern Choice and ADTree Edition

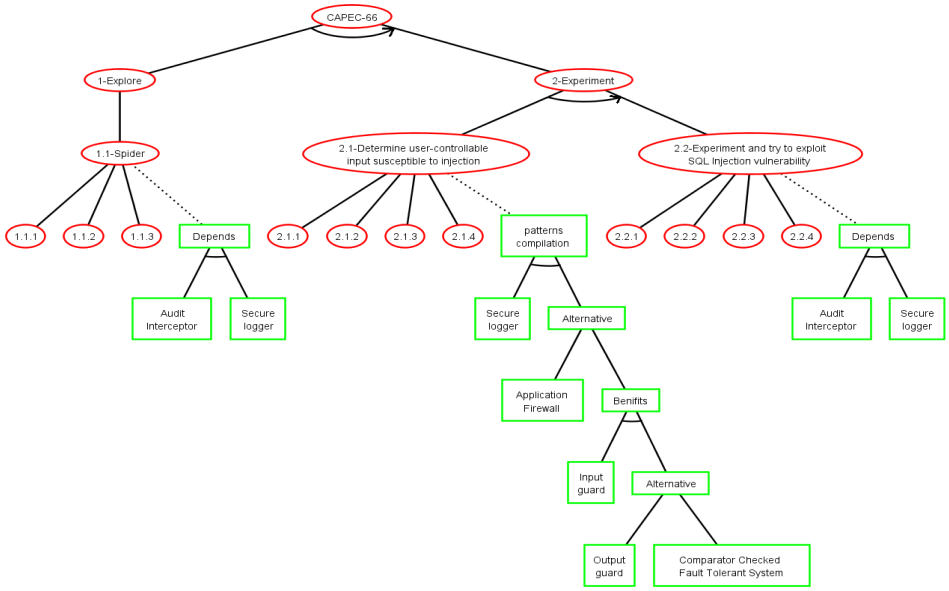


Figure 6. ADTree of the Attack CAPEC-66

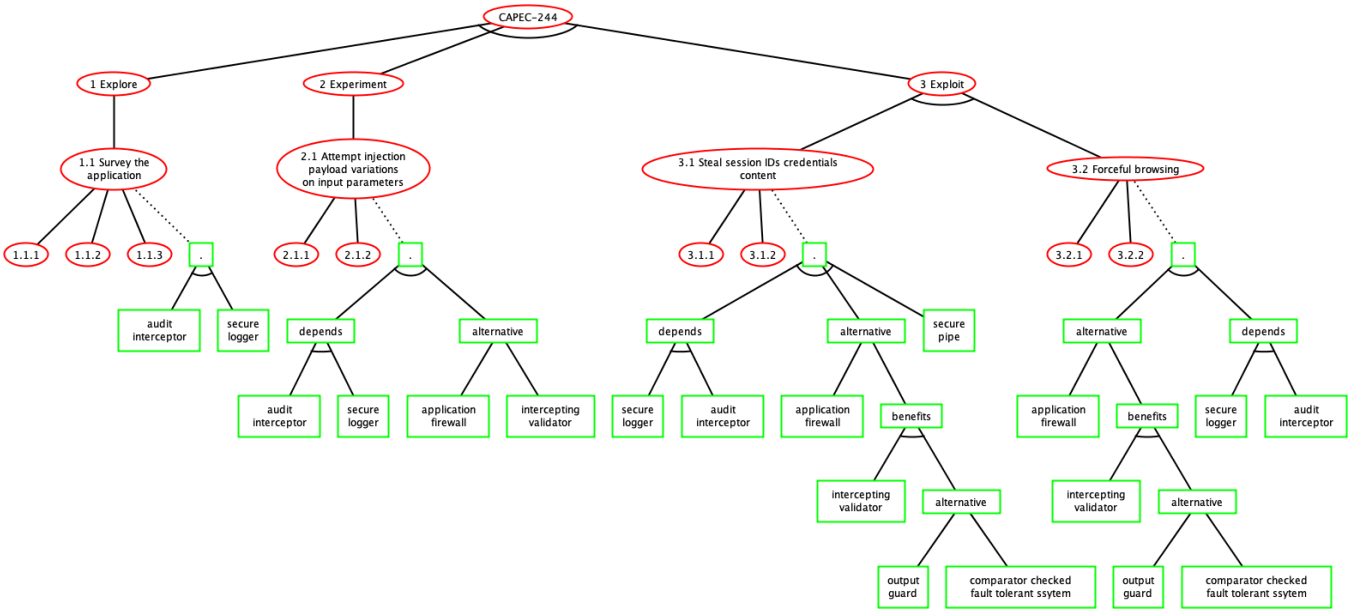


Figure 7. ADTree of the Attack CAPEC-244

The developer may now edit every ADTree $T(A)$ generated by the previous step and choose security patterns when several possibilities are available. We assume that the defense nodes linked to attack nodes have conjunctive refinements of nodes labelled by security patterns only. Figure 8 depicts an example of modified ADTree of the attack CAPEC-244.

Every attack node A of the initial ADTree T is now automatically replaced with the ADTree $T(A)$. This step is achieved by substituting every term A in the ADTerm $\iota(T)$ by $\iota(T(A))$. We denote $\iota(T_f)$ the resulting ADTerm and T_f the final ADTree. It depicts a logical breakdown of the options available to an attacker and the defences, materialised with security patterns, which have to be inserted into the application model and then implemented. The security pattern set found in T_f is denoted $SP(T_f)$.

This step finally builds a report by extracting from KB the test architecture descriptions needed for executing the attacks on the AUT and observing its reactions.

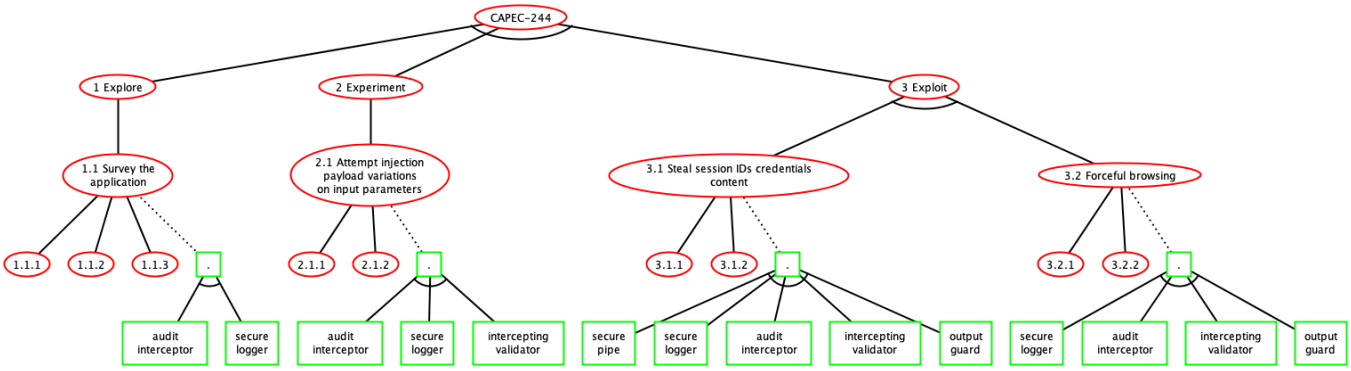


Figure 8. Final ADTree of the Attack CAPEC-244

C. Security Testing

We now extract attack-defense scenarios to later build test suites that will check whether attacks are effective on the AUT. An attack-defense scenario is a minimal combination of events leading to the root attack, minimal in the sense that, if any event of the attack-defense scenario is omitted, then the root goal will not be achieved.

The set of attack-defense scenarios of T_f are extracted by means of the disjunctive decomposition of $\iota(Tf)$:

Definition 3 (Attack scenarios) *Let T_f be an ADTree and $\iota(T_f)$ be its ADTerm. The set of Attack scenarios of T_f, denoted $SC(T_f)$ is the set of clauses of the disjunctive normal form of $\iota(T_f)$ over $BADStep(T_f)$.*

$BADStep(s)$ denotes the set of BADSteps of a scenario s.

An attack scenario s is still an ADTerm. Its satisfiability means that the main goal of the ADTree T_f is feasible by achieving the scenario formulated by s. $BADStep(s)$ denotes the set of BADSteps of s.

Step 4: Test Suite Generation

Let $s \in SC(Tf)$ be an attack-defense scenario and $b = c^p(st, sp) \in BADStep(s)$ a BADSteps of s. Step 4 generates the GWT test case $TC(b)$ composed of 3 sections extracted from KB with the relations $testG$, $testW$ and $testT$: we have one Given section, one When section and one Then section, each related to one procedure. This Then section aims to assert whether the AUT is vulnerable to the attack step st executed by the When section.

The final test suite TS, derived from an ADTree T_f, is obtained after having iteratively applied this test case construction on the scenarios of $SC(T_f)$. This is captured by the following definition:

Definition 4 (Test suites) *Let T_f be an ADTree, $s \in SC(Tf)$ and $b \in BADStep(s)$.*
$TS = \{TC(b) \mid b = c^p(st, sp) \in BADStep(s)$ *and* $s \in SC(T_f)\}$.

```
@capec244
Feature: CAPEC-244: Cross-Site Scripting via
    Encoded URI Schemes
#1. Explore
Scenario: Step1.1 Survey the application
Given prepare to Survey the application
When Try to Survey the application
# assertion for attack step success
Then Assert the success of Survey the
    application
```

Figure 9. The test case stub of the first step of the attack CAPEC 244

We have implemented these steps to yield GWT test case stubs compatible with the Cucumber framework [37], which supports a large number of languages. Figure 9 gives a test case stub example obtained with our tool from the first step of the attack CAPEC-244 depicted in Figure 7. The test case lists the Given When Then sections in a readable manner. Every section is associated to a generic procedure stored into another file. The procedure related to the When and Then sections are given in Figure 10. The comments come from KB and the CAPEC base. In this example, the procedure includes a generic block of code, which may be reused with several applications; the "getSpider()" method relates to the call of the ZAProxy[1] tool, which crawls a Web application to get its URLs.

Step 8: Test Case Stub Completion

In the beginning of this step, the test case procedures are generic, which means that they are composed of comments or generic block of codes that help developers complete them. In the previous test case example, it only remains for the developer to write the initial URL of the Web application before testing whether it can be explored. Unfortunately, with other test cases, the developer might have to implement it completely.

After this step, we assume that the test cases are correctly

[1]https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project

```
@When("Try to Survey the application for user
    -controllable inputs")
public void trysurvey(){
// Try one of the following techniques :
//1. Use a spidering tool to follow and record
    all links and analyze the web pages to find
    entry points. Make special note of any
    links that include parameters in the URL.
//2. Use a proxy tool to record all user input
    entry points visited during a manual
    traversal of the web application.
//3. Use a browser to manually explore the
    website and analyze how it is constructed.
    Many browsers' plugins are available to
    facilitate the analysis or automate the
    discovery.
String url ="";
ZAProxyScanner j = new
    ZAProxyScanner("localhost", 8080, "zap");
j.spider(url);
}
@Then("Assert the success of Survey the
    application for user-controllable inputs")
public void asssurvey(){
// Assert one of the following indications :
// -A list of URLs, with their corresponding
    parameters (POST, GET, COOKIE, etc.) is
    created by the attacker.
ZAProxyScanner j = new
    ZAProxyScanner("localhost", 8080, "zap");
int x =
    j.getSpiderResults(j.getLastSpiderScanId())
.size();
Assert.assertTrue(x>0);
}}
```

Figure 10. The procedure related to the When and Then sections of Figure 9

developed with assertions in Then sections as stated in Section III: a Then section of a test case $TC(b)$ returns the verdict "$Pass_{st}$" if an attack step st has been successfully applied on the AUT and "$Fail_{st}$" otherwise; when $TC(b)$ returns an unexpected exception or fault, we get the verdict "$Inconclusive_{st}$".

Step 9: Test Case Execution

The experimentation of the AUT with the test suite TS is carried out in this step. A test case $TC(b)$ of TS, which aims at testing whether the AUT is vulnerable to an attack step st leads to a local verdict denoted $\text{Verdict}(TC(b)||AUT)$:

Definition 5 (Local Test Verdicts) *Let AUT be an application under test, $b = c^p(st, sp) \in BADStep(T_f)$, and $TC(b) \in TS$ be a test case.*
$\text{Verdict}(TC(b)||AUT) =$

- *$Pass_{st}$, which means AUT is vulnerable to the attack step st;*
- *$Fail_{st}$, which means AUT does not appear to be vulnerable to the attack step st;*
- *$Inconclusive_{st}$, which means that various problems occurred while the test case execution.*

We finally define the final verdicts of the security testing stage with regard to the ADTree T_f. These verdicts are given with the predicates $\text{Vulnerable}(T_f)$ and $\text{Inconclusive}(T_f)$ returning boolean values. The intermediate predicate $\text{Vulnerable}(b)$ is also defined on a BADStep b to evaluate a substitution $\sigma : BADStep(s) \rightarrow \{true, false\}$ on an attack-defense scenario s. A scenario s holds if the evaluation of the substitution σ to s, i.e., replacing every BADStep term b with the evaluation of $\text{Vulnerable}(b)$, returns true. The predicate $\text{Vulnerable}(s)$ expresses whether an attack-defense scenario of T_f holds. In that case, the threat modelled by T_f can be achieved on AUT. This is defined with the predicate $\text{Vulnerable}(T_f)$:

Definition 6 (Security Testing Verdicts) *Let AUT be an application under test, T_f be an ADTree, $s \in SC(T_f)$ and $b = c^p(st, sp) \in BADStep(s)$.*

1) $\text{Vulnerable}(b) =_{def} true$ *if* $\text{Verdict}(TC(b)||AUT) = Pass_{st}$*; otherwise,* $\text{Vulnerable}(b) =_{def} false$*;*
2) $\text{Vulnerable}(s) =_{def} true$ *if* $eval(s\sigma)$ *returns true, with* $\sigma : BADStep(s) \rightarrow \{true, false\}$ *the substitution* $\{b_1 \rightarrow \text{Vulnerable}(b_1), \ldots, b_n \rightarrow \text{Vulnerable}(b_n)\}$*; otherwise,* $\text{Vulnerable}(s) =_{def} false$*;*
3) $\text{Inconclusive}(s) =_{def} true$ *if* $\exists b \in \text{BADStep}(s)$*:* $\text{Verdict}(TC(b)||\ AUT) = Inconclusive_{st}$*; otherwise,* $\text{Inconclusive}(s) =_{def} false$*.*
4) $\text{Vulnerable}(T_f) =_{def} true$ *if* $\exists s \in SC(T_f) : \text{Vulnerable}(s) = true$*; otherwise,* $\text{Vulnerable}(T_f) =_{def} false$*;*
5) $\text{Inconclusive}(T_f) =_{def} true$ *if* $\exists s \in SC(T_f)$, $\text{Inconclusive}(s) = true$*; otherwise,* $\text{Inconclusive}(T_f) =_{def} false$*.*

D. Security Pattern Verification

Our approach also aims at checking whether security patterns are correctly implemented in the application. The security testing stage is indeed insufficient because the non-detection of vulnerability in the AUT does not imply that a security pattern is correctly implemented. As stated earlier, we propose to generate LTL properties that express the behavioural properties of a security pattern. Then, these are used to check whether they hold on the AUT traces. The originality of our approach resides in the fact that we do not ask developers for writing formal properties, we propose to generate them by means of KB.

Steps 5 and 6: UML Sequence Diagram Extraction and Modification

After the threat modelling stage, this step starts by extracting from KB a list of generic UML sequence diagrams for each security pattern in $SP(T_f)$. These diagrams show how a security pattern should behave once it is correctly implemented, i.e., how objects interact in time. We now

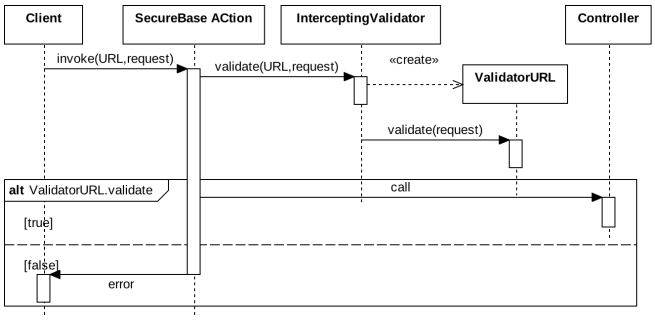


Figure 11. UML sequence diag. of the pattern Intercepting Validator

suppose that the developer implements every security pattern in the application. At the same time, he/she may adapt the behaviours illustrated in the UML sequence diagrams. In this case, we assume that the diagrams are updated accordingly.

Figure 11 illustrates an example of UML sequence diagram for the security pattern "Intercepting Validator". The diagram shows the interactions between an external Client, the pattern and the application, but also the interactions among the objects of the pattern. Here, the Intercepting Validator Object is called to validate requests. These are given to another object ValidatorURL, which filters the request with regard to the URL type. If the request is valid, it is processed by the application (Controller object), otherwise an error is returned to the client side.

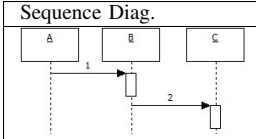
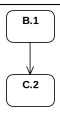
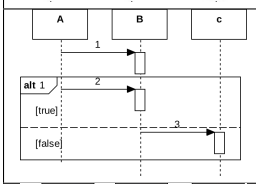
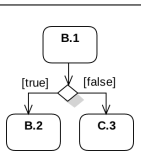
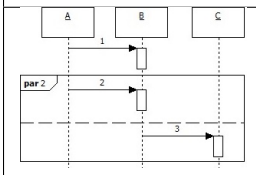
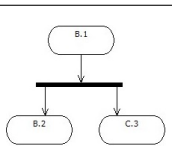
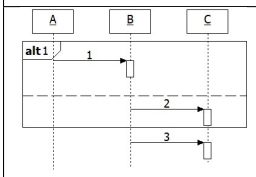
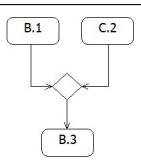
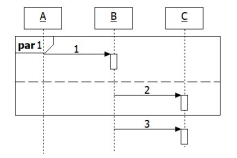
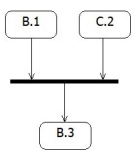
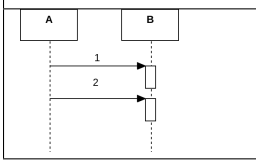
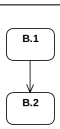
Step 7: Security Pattern LTL Property Generation

This step automatically generates LTL properties from UML sequence diagrams by detecting the cause-effect relations among method calls and expressing them in LTL. Initially, we took inspiration in the method of Muram et al. [38], which transforms activity diagrams into LTL properties. Unfortunately, security patterns are not described with activity diagrams, but with sequence diagrams. This is why we devised 20 conversion schemas allowing to transform UML sequence diagram constructs, composed of two or three successive actions, into UML activity diagrams. Table I gives 6 of these schemas. Intuitively, these translate two consecutive method calls found in a sequence diagram by activity diagrams composed of action states. The other schemas (not all given in Table I) are the results of slight adaptations of the five first ones, where the number of objects or the guards have been modified. For instance, the last schema of Table I is an adaptation of the first one, which depicts interactions between two objects instead of three.

Then, we propose 20 rules to translate these activity diagrams into LTL properties. The last column of Table I lists 6 of these rules. Some of these rules are based on those proposed by Muram et al, but we devised other rules related to our own activity diagrams, which are more detailed. For instance, we take into account the condition state in the second rule to produce more precise LTL properties.

At the end of this step, we consider having a set of LTL properties $P(sp)$ for every security pattern $sp \in SP(T_f)$. Although the LTL properties of $P(sp)$ do not necessarily cover all the possible behavioural properties of a security pattern sp, this process offers the advantages of not asking developers for writing LTL formula or to instantiate generic LTL properties to match the application model or code.

Table I
UML SEQUENCE DIAGRAMS TO LTL PROPERTIES TRANSFORMATION RULES.

Sequence Diag.	Activity Diag.	LTL properties
		$\square(B.1 \longrightarrow \Diamond C.2)$
		$\square(B.1 \longrightarrow \Diamond B.2)\ xor\ (\neg B.1 \longrightarrow \Diamond C.3))$
		$\square(B.1 \longrightarrow (\Diamond B.2) and (\Diamond C.3))$
		$\square(B.1 xor C.3 \longrightarrow \Diamond B.3)$
		$\square(B.1 and C.3 \longrightarrow \Diamond B.3)$
		$\square(B.1 \longrightarrow \Diamond B.2)$

From the example of UML sequence diagram given in Figure 11, 4 LTL properties are generated. Table II lists them. These capture the cause-effect relations of every pair of methods found in the UML sequence diagram.

Step 10: Security Pattern Verification

As stated earlier, we consider that the AUT is instrumented with a debugger or similar tool to collect the methods called in the application while the execution of the test cases of TS. After the test case execution, we hence have a set of method call traces denoted $Traces(AUT)$.

Table II
LTL PROPERTIES FOR THE PATTERN INTERCEPTING VALIDATOR.

p_1	$\Box(SecureBaseAction.invokes \longrightarrow \Diamond InterceptingValidator.validate$
p_2	$\Box(InterceptingValidator.validate \longrightarrow \Diamond ValidatorURL.create)$
p_3	$\Box(ValidatorURL.create \longrightarrow \Diamond ValidatorURL.validate)$
p_4	$\Box((ValidatorURL.validate \longrightarrow \Diamond Controller.call)\ xor\ (\neg ValidatorURL.validate \longrightarrow \Diamond SecureBaseAction.error))$

Table III
TEST VERDICT SUMMARY AND RECOMMENDATIONS.

Vulnerable(T_f)	$\mathrm{Unsat}^b(SP(T_f))$	Incon(T_f)	Corrective actions
False	False	False	No issue detected
True	False	False	At least one scenario is successfully applied on AUT. Fix the pattern implementation. Or the chosen patterns are inconvenient.
False	True	False	Some pattern behavioural properties do not hold. Check the pattern implementations with the UML seq. diag. Or another pattern conceals the behaviour of the former.
True	True	False	The chosen security patterns are useless or incorrectly implemented. Review the ADTree, fix AUT.
T/F	T/F	True	The test case execution crashed or returned unexpected exceptions. Check the Test architecture and the test case codes.

A model-checking tool is now used to detect the non-satisfiability of LTL properties on $Traces(AUT)$. Given a security pattern *sp*, the predicate $Unsat^b(sp)$ formulates the non-satisfiability of a LTL property of *sp* in $Traces(AUT)$. The final predicate $\mathrm{Unsat}^b(SP(T_f))$ expresses whether all the LTL properties of the security patterns given in T_f hold.

Definition 7 (Security Pattern Verification Verdicts) *Let AUT be an application under test, T_f be an ADTree, and $sp \in SP(T_f)$ be a security pattern.*

1) $\mathrm{Unsat}^b(sp) =_{def} true$ *if* $\exists p \in P(sp), \exists t \in Traces(AUT), t \not\models p$; *otherwise,* $\mathrm{Unsat}^b(sp) =_{def} false$;
2) $\mathrm{Unsat}^b(SP(T_f)) =_{def} true$ *if* $\exists sp \in SP(T_f), \mathrm{Unsat}^b(sp) = true$; *otherwise,* $\mathrm{Unsat}^b(SP(T_f)) =_{def} false$;

Table III informally summarises the meaning of some test verdicts and some corrections that may be followed in case of failure.

V. IMPLEMENTATION

Our approach is implemented in Java and is released as open source in [5]. At the moment, the AUT must be a Web application developed with any kind of language provided that the AUT may be instrumented to collect method call traces. The prototype tool consists of three main parts. The first one comes down to a set of command lines allowing to build the knowledge base KB. The data integration is mostly performed by calling the tool Talend, which is specialised into the extract, transform, load (ETL) procedure. An example of knowledge base is available in [5].

A second software program semi-automatically generates ADTrees and GWT test cases. ADTrees are stored into XML files, and may be edited with *ADTool* [3]. GWT test cases are written in Java with the Cucumber framework, which supports the GWT test case pattern. These test cases can be imported as an Eclipse project to be completed and executed. This software program also provides UML sequence diagrams stored in JSON files, which have to be modified to match the AUT functioning. LTL properties are extracted from these UML sequence diagrams.

The last part of the tool is a tester that experiments Web applications with test cases and returns test verdicts. While the test case execution, we collect log files including method call traces. The LTL property verification on these traces is manually done by these steps: 1) the log files usually have to be manually filtered to remove necessary events 2) the tool Texada [39] is invoked to check the satisfiability of every LTL property on the log files. This tool takes as inputs a log file, a LTL property composed of variables and a list of events specifying variables in the formula to be interpreted as a constant event. Texada returns the number of times that a property holds in a log file. We have chosen the Texada tool as it offers good performance and can be used on large trace sets. But other tools could also be used, e.g., the LTL checker plugin of the ProM framework [40] or Eagle [41].

VI. PRELIMINARY EVALUATION

First and foremost, it is worth noting that we carried out in [24] a first evaluation of the difficulty of using security patterns for designing secure applications. This evaluation was conducted on 24 participants and allowed us to conclude that the Threat modelling stage of our approach makes the security pattern choice and the test case development easier and makes users more effective on security testing. In this paper, we propose another evaluation of the security testing and security pattern testing parts of our approach. This evaluation addresses the following research questions:

- Q1: Can the generated test cases detect security issues?
- Q2: Can the generated LTL properties detect incorrect implementation of patterns?
- Q3: How long does it take to discover errors (Performance)?

A. Empirical Setup

We asked ten teams of two students to implement Web applications written in PHP as a part of their courses. They could choose to develop either a blog, or a todo list application or a RSS reader. Among the requirements, the Web applications had to manage several kinds of users (visitors, administrators, etc.), to be implemented in object-oriented programming, to use the PHP Data Objects (PDO) extension to prevent SQL injections, and to validate all the user inputs. As a solution to filter inputs, we proposed them

to apply the security pattern Intercepting Validator. But its use was not mandatory.

Then, we applied our tool on these 10 Web applications in order to:

- test whether these are vulnerable to both SQL and XSS injections (attacks CAPEC-66 and CAPEC-244). With our tool, we generated the ADTrees of Figures 6 and 7, along with GWT test cases. We completed them to call the penetration testing tool ZAProxy (as illustrated in Figure 10). All the applications were vulnerable to the steps "Explore" of the ADTrees (application survey), therefore we also experimented them with the test cases related to the steps "Experiment" (attempt SQL or XSS injections);
- test whether the behaviours of the pattern Intercepting Validator are correctly implemented in the 10 Web applications. We took the UML sequence diagram of Figure 11 and adapted it ten times to match the context of every application. Most of the time, we had to change the class or method names, and to add as many validator classes as there are in the application codes. When a class or method of the pattern was not implemented, we leaved the generic name in the UML diagram. Then, we generated LTL properties to verify whether they hold in the application traces.

B. Q1: Can the generated test cases detect security issues?

Procedure: to study Q1, we experimented the 10 applications with the 4 GWT test cases of the two Steps Explore and Experiment of the attacks CAPEC-66 and CAPEC-244. As these test cases call a penetration testing tool, which may report false positives, we manually checked the reported errors to only keep the real ones. We also inspected the application codes to examine the security flaw causes and to finally check whether the applications are vulnerable.

Results: Table IV provides the number of tests for both attacks (columns 2 and 3), the number of security errors detected by these tests (columns 4 and 5) and execution times in seconds (column 6). As a penetration testing tool is called, a large amount of malicious HTTP requests are sent to the applications in order to test them. The test number often depends on the application structure (e.g., number of classes, of called libraries, of URLs, etc.) but also on the number of forms available in an application.

Table IV shows that errors are detected in half of the applications. After inspection, we observed that several inputs are not filtered in App. 1, 5 and 6. On the contrary, for App. 3 and 7 all the inputs are checked. However, the validation process is itself incorrectly performed or too straightforward. For example, in App. 3 the validation comes down to checking that the input exists, which is far from sufficient to block malicious code injections. For the other applications, we observed that they all include a correct validation process, which is called after every client request. After the code inspection and the testing process, we conclude that they seem to be protected against both XML and SQL injections. These experiments tend to confirm that our approach can be used to test the security of Web applications.

Table IV
RESULTS OF THE SECURITY TESTING STAGE: NUMBER OF REQUESTS PERFORMED, NUMBER OF DETECTED SECURITY ERRORS, AND EXECUTION TIMES IN SECOND

App.	# XSS tests	# SQL tests	# XSS detection	# SQL detection	time(s)
1	1610	199	1	0	14
2	12358	796	0	0	924
3	8209	398	10	4	29
4	7347	199	0	0	81
5	2527	398	3	0	1137
6	5884	597	1	1	30
7	9954	1194	1	0	49
8	2464	796	0	0	1478
9	1709	796	0	0	47
10	16441	796	0	0	93

C. Q2: Can the generated LTL properties detect incorrect implementation of patterns?

Procedure: To investigate Q2, the PHP applications were instrumented with the debugger Xdebug, and we collected logs composed of method call traces while the test case execution. Then, we used the tool Texada to check whether every LTL property holds in these method call traces. When the pattern is strictly implemented as it is described in the UML sequence diagram of Figure 11 (1 class Validator), 4 LTL properties are generated, as in Table I. However, the number of LTL properties may differ from one application to the other, with regard to the number of classes used to implement the security pattern. When there are more than 4 LTL properties for an application, the additional ones capture the call of supplementary Validator classes and only differ from the properties of Table I by the modification of the variable ValidatorURL. To keep our results comparable from one application to another, we denote with the notation p_i the set of properties related to the property p_i in I.

Furthermore, both authors independently checked the validation part in every applications to assess how the security pattern is implemented in order to ensure that a property violation implies that a security pattern behaviour is not correctly implemented.

Results: Table V lists in columns 2-5 the violations of the properties derived from those given in Table I for the 10 applications. These results firstly show that our approach detects that the security pattern Intercepting Validator is never correctly implemented. The pattern seems to be almost implemented in App. 2 because only p_4 does not hold here. An inspection of the application code confirms that the pattern structure is correctly implemented as well as most of its method call sequences. But we observed that

the application does not always return an error to the user when some inputs are not validated. This contradicts one of the pattern purposes.

App. 3, 4, 7-10 include some sorts of input filtering processes at least defined in one class. But, these do not respect the security pattern behaviours. Most of the time, we observed that the validation process is implemented in a single class instead of having an Intercepting Validator calling other Validator classes. This misbehaviour is detected by the violations of the properties p_2 and p_3. Besides, we observed that the input validation is not systematically performed in App. 1, 5 and 6. This is detected by our tool with the violation of p_1. As a consequence, it is not surprising to observe that these applications are vulnerable to malicious injections. We also observed that when App. 5 validates the inputs, it does not define the validation logic in a class. The fact that the security pattern is not invoked is detected by the violation of p_1. But, this property violation does not reveal that there is another validation process implemented.

In summary, our application code inspections confirmed the results of Table V. In addition to assessing whether the security pattern behaviours are correctly implemented, we observed that our approach may also help learn more information about the validation process, without inspecting the code. For instance, the properties based on p_1 check whether a validation method defined in a class is called every time a client request is received. The properties based on p_4 give information about the error management. Their violations express that users are not always warned when invalid inputs are provided to the applications.

Table V
RESULTS OF THE SECURITY PATTERN TESTING STAGE: VIOLATION OF THE LTL PROPERTIES AND EXECUTION TIMES IN SECOND

App.	p_1	p_2	p_3	p_4	Time(min)
1	X	X			4,02
2				X	51,15
3			X	X	19,12
4			X	X	29,34
5	X	X	X	X	6,5
6	X	X	X	X	14,40
7			X	X	24,77
8		X	X	X	7,24
9		X	X	X	5,56
10			X	X	67,03

D. Q3: How long does it take to discover errors (Performance)?

Procedure: We measured the time consumed by the tool to carry out security testing and security pattern verification for the 10 applications. Execution times are given in Tables IV and V. Furthermore, we also measured the number of LTL properties that are generated for 11 security patterns, which are often used with Web applications, as the LTL property number influences execution times.

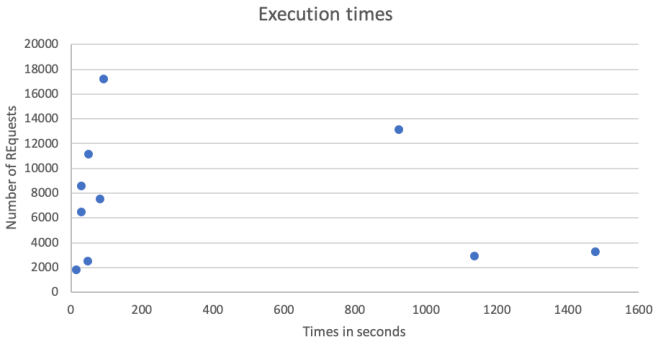


Figure 12. Execution times of the security testing stage for the ten applications

Results: The plot chart of Figure 12 shows that security testing requires less than 2 minutes for 7 applications independently on the number of tests, whereas it requires more than 15 minutes for the 3 others. The security testing stage depends on several external factors, which makes it difficult to draw consistent conclusions. It firstly depends on the test case implementation; in our evaluation, we choose to call a penetration testing tool, therefore, execution times mostly depend on it. Another factor is the application structure (nb of classes, calls of external URLS, etc.). Therefore, we can only conclude here is that execution times are lower than 25 minutes, which remains reasonable with regard to the number of requests sent to applications.

The time required to detect property violations in method call traces is given in Column 6 of Table V. Execution times vary here between 4 and 67 minutes according to the number of traces collected from the application and the number of generated LTL properties. For example, for App. 1O, 17237 security tests have been executed, and 17237 traces of about 30 events have been stored in several log files. Furthermore, 7 LTL properties have been generated for this applications. These results, and particularly the size of the trace set, explain the time required to check whether the LTL properties hold. In general terms, we consider that execution times remain reasonable with regard to the trace set sizes of the applications.

Table VI finally shows the number of LTL properties generated from generic UML properties (without adapting them to application contexts) for 11 security patterns whose descriptions include UML sequence diagrams. For these patterns, the property number is lower or equal than 13. For every pattern, the property number is in a range that seems reasonably well supported by model checkers. However, if several security patterns have to be tested, the property number might quickly exceed the model-checker limits. This is we have chosen in our approach to check the satisfiability of each LTL property, one after the other, on method call traces.

Table VI
LTL PROPERTY GENERATION FOR SOME SECURITY PATTERNS

Security pattern	# UML diag.	# LTL properties
Authentication Enforcer	3	9
Authorization Enforcer	3	13
Intercepting Validator	2	4
Secure Base Action	2	5
Secure Logger	2	5
Secure Pipe	2	10
Secure Service Proxy	2	6
Intercepting Web Agent	2	9
Audit Interceptor	2	7
Container Managed Security	2	7
Obfuscated Transfer Object	2	10
Obfuscated Transfer Object	2	10

E. Threat to Validity

This preliminary experimental evaluation is applied on 10 Web applications, and not on other kinds of software or systems. This is a threat to external validity, and this is why we avoid drawing any general conclusion. But, we believe that this threat is somewhat mitigated by our choice of application, as the Web application context is a rich field in great demand in the software industry. Web applications also expose a lot of well-known vulnerabilities, which helps in the experiment set-up. In addition, the numbers of security patterns considered in the evaluation were insufficient. Hence, it is possible that our method is not applicable to all security patterns. In particular, we assume that generic UML sequence diagrams are provided in the security pattern descriptions. This is the case for the patterns available in the catalogue of Yskout et al. [35], but not for all the patterns listed in [8]. To generalise the approach, we also need to consider more general patterns and employ large-scale examples.

The evaluation is based on the work of students, but this public is sometimes considered as a bias in evaluations. Students are usually not yet meticulous on the security solution implementation, and as we wished to experiment vulnerable applications to check that our approach can detect security flaws, we consider that applications developed by students meet our needs.

A threat to internal validity is related to the test case development. Our approach aims at guiding developers in the test case writing and security pattern choice. In the evaluation, we chose to complete test cases with the call of a penetration testing tool. The testing results would be different (better or worse) with other test cases. Significant advances have been made in these tools, which are more and more employed in the industry. Therefore, we believe that their use and the test cases considered in the experiments are close to real use cases. In the same way, we manually updated UML sequence diagrams to generate LTL properties that correspond to the application contexts. But, it is possible that we inadvertently made some mistakes, which led to false positives. To avoid this bias, we manually checked the correctness of the results by replaying the counterexamples returned by the model-checker and by inspecting the application codes.

VII. CONCLUSION

Securing software requires that developers acquire a lot of expertise in various stages of software engineering, e.g., in security design, or in testing. To help them in these tasks, we have proposed an approach based on the notion of knowledge base, which helps developers in the implementation of secure applications through steps covering threat modelling, security pattern choice, security testing and the verification of security pattern behavioural properties. This paper proposes two main contributions. It assists developers in the writing of concrete security test cases and ADTrees. It also checks whether security patterns properties are met in application traces by automatically generating LTL properties from the UML sequence diagrams that express the behaviours of patterns. Therefore, the approach does not require developers to have skills in (formal) modelling or in formal methods. We have implemented this approach in a tool prototype [5]. We conducted an evaluation of our approach on ten Web applications, which suggests that it can be used in practice.

Future work should complement the evaluation to confirm that the approach can be applied on more kinds of applications. We also mentioned that security pattern descriptions do not all include UML sequence diagrams, which are yet mandatory by our approach. We will try to solve this lack of documentation by investigating whether security pattern behavioural properties could be expressed differently, e.g., with annotations added inside application codes. In addition, we intend to consider how our ADTree generation could support the teaching of security testing and security by design.

REFERENCES

[1] L. Regainia and S. Salva, “A practical way of testing security patterns,” in *Thirteenth International Conference on Software Engineering Advances (ICSEA’18)*, Nice, France, Oct. 2018, pp. 1–7.

[2] E. Rodriguez, “Security Design Patterns,” in *19th Annual Computer Security Application Conference (ACSAC’03)*, 2003.

[3] B. Kordy, S. Mauw, S. Radomirović, and P. Schweitzer, “Attack–defense trees,” *Journal of Logic and Computation*, pp. 1–38, 2012.

[4] T. Kobashi, M. Yoshizawa, H. Washizaki, Y. Fukazawa, N. Yoshioka, T. Okubo, and H. Kaiya, “Tesem: A tool for verifying security design pattern applications by model testing,” in *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*, April 2015, pp. 1–8.

[5] L. Regainia and S. Salva. (2019) Security pattern classification, companion site. (Date last accessed march 2019). [Online]. Available: http://regainia.com/research/companion.html

[6] J. Yoder, J. Yoder, J. Barcalow, and J. Barcalow, "Architectural patterns for enabling application security," *Proceedings of PLoP 1997*, vol. 51, p. 31, 1998.

[7] M. Schumacher, *Security Engineering with Patterns: Origins, Theoretical Models, and New Applications*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2003.

[8] R. Slavin and J. Niu. (2017) Security patterns repository. [Online]. Available: http://sefm.cs.utsa.edu/repository/

[9] K. Yskout, R. Scandariato, and W. Joosen, "Do security patterns really help designers?" in *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ser. ICSE '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 292–302.

[10] K. Yskout, T. Heyman, R. Scandariato, and W. Joosen, "A system of security patterns," 2006.

[11] R. Jhawar, B. Kordy, S. Mauw, S. Radomirović, and R. Trujillo-Rasua, "Attack trees with sequential conjunction," in *IFIP International Information Security Conference*. Springer, 2015, pp. 339–353.

[12] Mitre corporation. (2019) Common attack pattern enumeration and classification, url:https://capec.mitre.org/. (Date last accessed march 2019). [Online]. Available: https://capec.mitre.org/

[13] P. Torr, "Demystifying the threat modeling process," *IEEE Security Privacy*, vol. 3, no. 5, pp. 66–70, Sept 2005.

[14] M. Howard and D. LeBlanc, *Writing Secure Code*, M. Press, Ed., 2003.

[15] I. Schieferdecker, J. Grossmann, and M. A. Schneider, "Model-based security testing," in *Proceedings 7th Workshop on Model-Based Testing, MBT 2012, Tallinn, Estonia, 25 March 2012.*, 2012, pp. 1–12.

[16] D. Xu, M. Tu, M. Sanford, L. Thomas, D. Woodraska, and W. Xu, "Automated security test generation with formal threat models," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 4, pp. 526–540, July 2012.

[17] J. Bozic, D. E. Simos, and F. Wotawa, "Attack pattern-based combinatorial testing," in *Proceedings of the 9th International Workshop on Automation of Software Test*, ser. AST 2014. New York, NY, USA: ACM, 2014, pp. 1–7.

[18] A. Morais, E. Martins, A. Cavalli, and W. Jimenez, "Security protocol testing using attack trees," in *2009 International Conference on Computational Science and Engineering*, vol. 2, Aug 2009, pp. 690–697.

[19] A. Marback, H. Do, K. He, S. Kondamarri, and D. Xu, "Security test generation using threat trees," in *2009 ICSE Workshop on Automation of Software Test*, May 2009, pp. 62–69.

[20] O. El Ariss and D. Xu, "Modeling security attacks with statecharts," in *Proceedings of the Joint ACM SIGSOFT Conference – QoSA and ACM SIGSOFT Symposium – ISARCS on Quality of Software Architectures – QoSA and Architecting Critical Systems – ISARCS*, ser. QoSA-ISARCS '11. New York, NY, USA: ACM, 2011, pp. 123–132.

[21] A. Marback, H. Do, K. He, S. Kondamarri, and D. Xu, "A threat model-based approach to security testing," *Softw. Pract. Exper.*, vol. 43, no. 2, pp. 241–258, Feb. 2013.

[22] N. Shahmehri *et al.*, "An advanced approach for modeling and detecting software vulnerabilities," *Inf. Softw. Technol.*, vol. 54, no. 9, pp. 997–1013, 2012.

[23] S. Salva and L. Regainia, "Using data integration to help design more secure applications," in *Proceedings of the 12th International Conference on Risks and Security of Internet and Systems*. Dinard, France: Springer-Verlag, Aug. 2017.

[24] ——, "Using data integration for security testing," in *Proceedings 29th International Conference, ICTSS 2017*, St. Petersburg, Russia, Oct. 2017, pp. 178–194.

[25] ——, "An approach for guiding developers in the choice of security solutions and in the generation of concrete test cases," *Software Quality Journal*, vol. 27, no. 2, pp. 675–701, January 2019. [Online]. Available: https://hal-clermont-univ.archives-ouvertes.fr/hal-02019145

[26] J. Dong, T. Peng, and Y. Zhao, "Automated verification of security pattern compositions," *Inf. Softw. Technol.*, vol. 52, no. 3, pp. 274–295, Mar. 2010.

[27] B. Hamid, C. Percebois, and D. Gouteux, "A methodology for integration of patterns with validation purpose," in *Proceedings of the 17th European Conference on Pattern Languages of Programs*, ser. EuroPLoP '12. New York, NY, USA: ACM, 2012, pp. 8:1–8:14.

[28] M. Yoshizawa *et al.*, "Verifying implementation of security design patterns using a test template," in *2014 Ninth International Conference on Availability, Reliability and Security*, Sept. 2014, pp. 178–183.

[29] L. Regaigna, C. Bouhours, and S. Salva, "A systematic approach to assist designers in security pattern integration," in *Second International Conference on Advances and Trends in Software Engineering (SOFTENG 2016)*, Lisbon, Portugal, Feb. 2016.

[30] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975.

[31] J. Viega and G. McGraw, *Building Secure Software: How to Avoid Security Problems the Right Way, Portable Documents*. Pearson Education, 2001.

[32] J. Scambray and E. Olson, *Improving Web Application Security*, 2003.

[33] V. Dialani, S. Miles, L. Moreau, D. De Roure, and M. Luck, "Transparent fault tolerance for web services based architectures," in *Euro-Par 2002 Parallel Processing*. Springer, 2002, pp. 889–898.

[34] J. Meier, "Web application security engineering," *Security & Privacy, IEEE*, vol. 4, no. 4, pp. 16–24, 2006.

[35] K. Yskout, R. Scandariato, and W. Joosen. Security pattern catalog. (Date last accessed march 2019). [Online]. Available: https://people.cs.kuleuven.be/~koen.yskout/icse15/catalog.pdf

[36] OWASP. (2019) Owasp testing guide v3.0 project. (Date last accessed march 2019). [Online]. Available: http://www.owasp.org/index.php/Category:OWASP_Testing_Project\#OWASP_Testing_Guide

[37] (2019) The cucumber framework. (Date last accessed march 2019). [Online]. Available: https://cucumber.io/

[38] F. U. Muram, H. Tran, and U. Zdun, "Automated mapping of UML activity diagrams to formal specifications for supporting containment checking," in *Proceedings 11th International Workshop on Formal Engineering approaches to Software Components and Architectures, FESCA 2014, Grenoble, France, 12th April 2014.*, 2014, pp. 93–107.

[39] C. Lemieux, D. Park, and I. Beschastnikh, "General ltl specification mining (t)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Nov 2015, pp. 81–92.

[40] F. M. Maggi, M. Westergaard, M. Montali, and W. M. P. van der Aalst, "Runtime verification of ltl-based declarative process models," in *Runtime Verification*, S. Khurshid and K. Sen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 131–146.

[41] H. Barringer, A. Goldberg, K. Havelund, and K. Sen, "Program monitoring with ltl in eagle," in *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.*, April 2004, pp. 264–272.

Network Analysis of City Streets: Forecasting Burglary Risk in Small Areas

Maria Mahfoud[1], Sandjai Bhulai[2], Rob van der Mei[1], Dimitry Erkin[1], and Elenna Dugundji[1]

[1] CWI National Research Institute for Mathematics and Computer Science
[2] Vrije Universiteit Amsterdam, Faculty of Science, Department of Mathematics
Email: M.Mahfoud@cwi.nl, S.Bhulai@vu.nl, R.D.van.der.Mei@cwi.nl, Dimitry.Erkin@gmail.com, E.R.Dugundji@vu.nl

***Abstract*—Predicting residential burglary can benefit from understanding human movement patterns within an urban area. Typically, these movements occur along street networks. To take the characteristics of such networks into account, one can use two measures in the analysis: betweenness and closeness. The former measures the popularity of a particular street segment, while the latter measures the average shortest path length from one node to every other node in the network. In this paper, we study the influence of the city street network on residential burglary by including these measures in our analysis. We show that the measures of the street network help in predicting residential burglary exposing that there is a relationship between conceptions in urban design and crime.**

Keywords–predictive analytics; forecasting; street network; betweenness centrality; closeness centrality; residential burglary

I. INTRODUCTION

Residential burglary is a crime with high impact for victims. Substantial academic research has accordingly been dedicated to understanding the process of residential burglary in order to prevent future burglaries [1]. In this attempt, several studies have focused on the role of the urban configuration in shaping crime patterns; this is regarded as one of the fundamental issues in environmental criminology, e.g., [2].

According to [3], environmental criminology is based on three premises. The first premise states that the nature of the immediate environment directly influences criminal behavior, thus a crime is not only reliant on criminogenic individuals, but also on criminogenic elements in the surroundings of a crime. The second premise states that crime is non-randomly distributed in time and space, meaning that crime is always concentrated around opportunities which occur on different moments in a day or week, or different places in a given geographical area. The third premise argues that understanding the criminogenic factors within a targeted environment, and capturing patterns and particular characteristics of that area, can reduce the number of crimes within that area.

Understanding human movement patterns within an urban area is essential for determining crime patterns [4]. These movements occur along a street network consisting of roads and intersections. Throughout the city street network, various places are connected, allowing transportation from one point to the next. Within the network, a street segment can be described as the road, or edge, linking two intersections, or nodes. In their study, [5] found that crime is tightly concentrated around crime hotspots that are located at specific points within the urban area. The urban configuration influences where these hotspots are located, suggesting that it is possible to deal with a large proportion of crime by focusing on relatively small areas. They found that crime hotspots are characterized by being stable over time, and that the hotspots are influenced by social and contextual characteristics of a specific geographical location. To be able to understand and prevent crime, it is important to examine these very small geographic areas, often as small as addresses of street segments, within the urban area. In an analysis of crime at street segment level, [6] reveal that crime trends at specific street segments were responsible for the overall observed trend in the city, emphasizing the need for understanding the development of crime at street segment level.

In urban studies, betweenness is a measure used to determine popularity or usage potential of a particular street segment for the travel movements made by the resident or ambient population through a street network [7], [8]. In criminology, betweenness represents the collective awareness spaces developed by people, including offenders, during the course of their routine activities. This metric provides a means to represent concepts, such as offender awareness, in empirical analysis [9]. Several studies have been conducted to uncover the effects of betweenness on crime. [9] investigated whether street segments that have a higher user potential measured by the network metric betweenness, have a higher risk of burglary. Also included in their research was the geometry of street segments via a measure of their linearity and different social-demographic covariates. They concluded that betweenness is a highly significant covariate when predicting burglaries at street segment level. In another study conducted by [10], a mathematical model of crime was presented that took the street network into account. The results of this study also show an evident effect of the street network.

In this research, we examine for small urban areas (4-digit postal codes: PC4) what the influence of the city street network is on residential burglary by applying betweenness as well as another centrality measure, closeness. These two centrality measures give different indications of the accessibility of an area and we study whether a more accessible area has a higher risk of residential burglary compared to a less accessible area. For comparison, we consider the same areas defined in our previous research [11]. In this earlier study, we predicted residential burglaries within different postal code areas for the district of Amsterdam-West. We extend the model of our earlier research by including the centrality measures closeness and betweenness as explanatory variables. Furthermore, we investigate which of the two centrality measures gives better outcomes, closeness or betweenness.

This paper is organized as follows. Section II describes

the dataset and the data analysis. Section III provides the methodological framework of this research. The results of the analysis are discussed in Section IV. In Section V, conclusions and recommendations for further research are presented.

II. DATA

The data used for this research is collected from three different data sources. The first dataset is provided by the Dutch Police and ranges from the first of January 2009 to 30 April 2014. The original dataset includes all recorded incidents of residential burglaries in the city of Amsterdam recorded at a monthly level and grouped into grids of 125 $\times$ 125 meters resulting in 94,224 records. Next to residential burglary, the dataset includes a wide range of covariates. These covariates provide information on the geographic information of the grid such as the number of Educational Institutions (EI) in the grid. In addition to these covariates, the data includes also spatial-temporal indicators of the following crime types: violation, mugging, and robbery. These spatial-temporal indicators measure the number of times a crime type happened within a given grid cell for a given time lag. The second dataset is obtained from the Statistics Netherlands (CBS) and includes various demographic and socio-economic covariates such as the average monthly income. This data is provided on a six alphanumeric postal code level where the first two digits indicate a region and a city, the second two digits indicate a neighborhood and the last two letters indicate a range of house numbers usually a street or a segment of a street. The third dataset is an internal dataset containing different centrality measures calculated on the street network of Amsterdam.

As this research focuses on explaining and predicting residential burglaries at the four-digit postal code level (PC4), the data should be aggregated at this level. Before aggregating the data we perform some pre-processing steps. First, we check the crime records for missing postal codes: if the postal code is missing then all linked data from CBS and the street network will be missing. We observed that 309 of the total 1,812 grid cells had a missing postal code (PC6). Some of these grid cells (34) were subsequently updated manually; other grid cells referred to industrial areas, bodies of water, railroads, grasslands, and highways. As a double check, we also confirmed whether there were residential burglaries in the remaining grid cells with missing postal codes; in our case, there were indeed none. These grid cells were further removed from the dataset and the data were aggregated based on PC4 conditioning on the district as some postal codes (PC4) can cover different police districts. Discrete covariates were aggregated by taking the sum of the covariate on all PC6. For continuous covariates, this was done by taking the average on all PC6. Exploring the data is done in a similar way as discussed in [11], where an extensive data analysis is applied to the crime data and the CBS data. To analyze this data we extend the final set of covariates by the different centrality measures and repeat the same step again. The dataset was assessed for outliers and collinearity. The presence of outliers was graphically assessed by the Cleveland dot plot and analytically by the Local Outlier Factor (LOF) with 10 neighbors and a threshold of 1.3. Results of this analysis show that the training data exhibits a percentage of outliers of 7.6. The majority of these occurred in December and January. Due to the high percentage of outliers in the training set, we decided to apply the analysis initially without outliers then apply the analysis with the outliers.

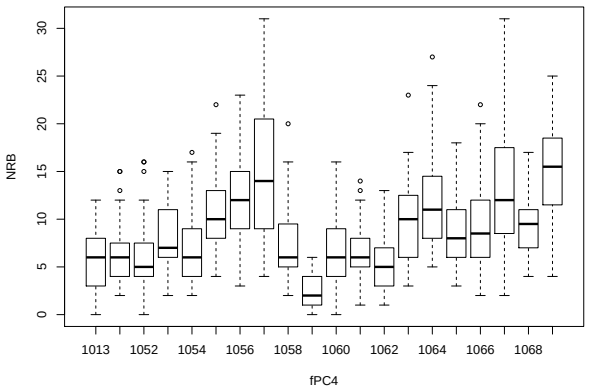


Figure 1. Boxplot of the number of burglaries conditional on the postal code indicating heterogeneity of variance in the number of burglaries within the different postal codes.

The collinearity was assessed by the calculating the variance inflation factor values (VIF) that measures the amount by which the variance of a parameter estimator is increased due to collinearity with other covariates rather than being orthogonal, e.g., [12]. A VIF threshold of 2 is used to assess collinearity [11]. This analysis results in the following set of covariates: the temporal covariate MONTH; the number of educational institutions (sEI), the number of restaurants (sRET), percentage of single-person households (aSH), the number of persons that generate income (sNPI), the total observed mugging incidents in the grid and its direct neighborhood in the last three months (sMuGL3M) and finally, the average monthly income (aAMI).

Furthermore, the relationship between residential burglaries and the categorical covariates was assessed using conditional box plots. Results show a temporal monthly effect and a spatial postal code effect on the burglaries. The effect of the postal codes on the burglaries is illustrated in Figure 1 where a clear difference in the mean and in the variance of the monthly number of burglaries is observed between the different postal codes.

III. METHODOLOGY

A. Centrality measures

Before discussing the centrality measures, we first need to introduce some important concepts of graph theory. A network represented mathematically by a graph is defined as a finite non-empty set V of vertices connected by edges E. A graph is usually written as $G = (V, E)$ where V is the set of vertices and E represents the set of edges where the number of vertices in G is called the order and the number of its edges is called the size. Two vertices u and v are said to be adjacent if there is an edge that links them together. In this case, u and v are also neighbors of each other. If two edges share one vertex then these edges are called adjacent edges. Using this concept of adjacency between all vertices represented in a matrix form

results in an adjacency matrix that summarizes all information describing a network.

Another concept for understanding centrality measures is the one of paths and shortest paths. Informally, a path is a way of traveling along edges from vertex u to vertex v without repeating any vertices [13]. Formally, a path P in a graph G is a subgraph of G whose vertices form an ordered sequence, such that every consecutive pair of vertices is connected by an edge. A path P is called an $u-v$ path in G if $P = (u = x_0, x_1, \ldots, x_j = v)$ s.t. $x_0x_1, x_1x_2, \ldots, x_{j-1}x_j$ are all edges of P. The number of edges in a path is called its length. The path $u-v$ with the minimum length is called the shortest path between u and v.

In the context of our analysis, a vertex represents an intersection between streets and an edge is a transport infrastructure supporting movements between the two intersections.

Paths can be considered as the key elements in defining centrality measures. In a transportation network, these centrality measures describe the flow of traffic on each particular edge of the network identifying the most important vertices in it. Some of these centrality measures that we will use in this paper are the closeness (CC) centrality and the betweenness centrality (BC).

Closeness is a very simple centrality measure to calculate. It is a geometric measure where the importance of a vertex depends on how many nodes exist at every distance. Closeness centrality can be defined as the average of the shortest path length from one node to every other node in the network and is given by:

$$\mathrm{CC}(\nu) = \frac{1}{\sum_{d(u,\nu)<\infty} \mathrm{d(u,\nu)}}, \tag{1}$$

where $d(u,\nu)$ is the distance between u and ν. Informally, closeness centrality measures how long it will take to spread information from node ν to all other nodes in the network and it is used to identify influential nodes in the network. The closeness of an edge $u-v$ can be calculated by taking the average closeness values of the nodes u and v.

The betweenness centrality BC is a path-based measure that can be used to identify highly influential nodes in the flow through the network. Given a specific node ν, the intuition behind betweenness is to measure the probability that a random shortest path will pass through ν. Formally, the betweenness of node ν, $\mathrm{BC}(\nu)$ is the percentage of shortest paths that include ν and can be calculated as follows:

$$\mathrm{BC}(\nu) = \sum_{u\neq w\neq \nu \in \mathrm{V}} \frac{\sigma_{u,w}(\nu)}{\sigma_{u,w}}, \tag{2}$$

where $\sigma_{u,w}$ is the total number of shortest paths between node u and w. Moreover, $\sigma_{u,w}(\nu)$ is the total number of shortest paths between node u and w that pass through ν. The betweenness of an edge e can be regarded as the degree to which an edge makes other connections possible and can be calculated in the same way by replacing the node ν by an edge e. An edge with high betweenness value forms an important bridge within the network. Removing this edge will severely hamper the flow of the network as it partitions the network into two large subnetworks.

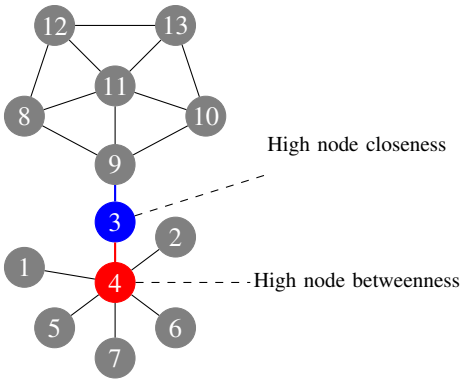


Figure 2. Illustration of high node (edge) betweenness and closeness.

High betweenness or closeness values indicate that a vertex or an edge can reach other vertices or edges, respectively, on relatively short paths. An example of a network is illustrated in Figure 2. In this example, node 3 has the highest closeness and node 4 the highest betweenness. The edge connecting the nodes 3 and 9 has the highest closeness within this network. This edge has also the highest betweenness together with the edge connecting the nodes 3 and 4.

In practice, it is almost impossible to calculate the exact betweenness or closeness scores. To make the calculations feasible, one can set a cut-off distance d and allow only paths that are at distances shorter or equal to d.

B. GAMM including centrality measures

In our paper [11], we used generalized additive mixed-effect models with different structures of the random component and showed that the one-way nested model with postal code as a random intercept has the optimal structure of the random component. Further, we showed that using the population as offset captures the most variation in the data. Moreover, the covariates month and the average monthly income seem to be the most important predictors for the number of burglaries within postal codes. In this paper, the optimal model discussed in [11] will be extended by two different centrality measures as covariates. We assess the effect of these centrality measures on explaining and forecasting the number of burglaries within the postal code. This model is given by:

$$\begin{aligned} \mathrm{y}_{i,t} &\sim \mathrm{Poisson}(\mu_{i,t}), \\ \mu_{i,t} &= \exp(\mathrm{base}_{i,t} + \mathrm{CM}_i + \mathrm{a}_i), \\ \mathrm{a}_i &\sim N(0, \sigma^2_{PC4}), \end{aligned} \tag{3}$$

where a_i is a random intercept for the postal code and $\mathrm{CM_i}$ represents the closeness $\mathrm{CC_i}$ or the betweenness $\mathrm{BC_i}$. The $\mathrm{base_{i,t}}$ is given by:

$$\begin{aligned} \mathrm{base}_{i,t} =& 1 + \mathrm{sEI}_i + \mathrm{sRET}_i + \mathrm{aSH}_i + \mathrm{sNPI}_i + \\ & \mathrm{sMugL3M}_{i,t} + f_1(\mathrm{aAMI}_i) + f_2(\mathrm{Month}_t). \end{aligned} \tag{4}$$

The models were fitted using the Laplace approximate maximum likelihood [14]. This allows comparing the models

based on the Akaike Information Criterion (AIC). All analyses were conducted using the `gamm4` package [15].

To assess the predictive performance of the models, the Root Mean Squared Error (RMSE) is calculated for an out-of-sample test. If $\mathrm{y}_{i,t}$ denotes the realization in postal code i and in month t, and $\hat{\mathrm{y}}_{i,t}$ denotes the forecast in the same postal code and in the same month, then the forecast error is given by $\mathrm{e}_{i,t} = \mathrm{y}_{i,t} - \hat{\mathrm{y}}_{i,t}$ and the RMSE is given by:

$$\mathrm{RMSE} = \sqrt{\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\mathrm{e}_{i,t}^2}. \tag{5}$$

C. Space-time model including centrality measures

In our paper [11], we have shown that adding the centrality measure as a covariate to a random intercept model has improved the performance of the model. We also showed that using the closeness as centrality measure with smaller thresholds results in better model performance. The betweenness as a centrality measure leads to better model performance using larger thresholds (larger than 4 minutes). In this paper, we will assess the effect of the centrality measures on burglary risk when modeling the spatial and temporal effects explicitly. Instead of using a random intercept model to account for extra variation within the postal codes, we will model the spatial effect taking into account the spatial autocorrelation.

The main spatial effect ξ_i of area i will be modeled as the sum of a structured effect u_i and an unstructured spatial effect ν_i. The structured spatial effect will be modeled by the mean of a first order intrinsic Gaussian Markov random field [16], [17]. In this specification, the mean of u_i is given by the mean of the adjacent u_i's and the variance of u_i is inversely proportional to the number of neighbors $\mathcal{N}_i$ of area i. The unstructured spatial effect is modeled using exchangeability among the different postal codes. Moreover, the temporal trend of burglary risk is modeled by the mean of a structured and an unstructured component. The temporally structured component is modeled dynamically using a random walk of order 2 and the unstructured component is specified as zero-mean white noise with precision τ_ν.

$$\begin{aligned}
&\mathrm{y}_{i,t} \sim \mathrm{Poisson}(\mu_{i,t}),\\
&\mu_{i,t} = \exp\left(\mathrm{base}_{i,t} + \mathrm{CM}_i + \mathrm{u}_i + \nu_i + \gamma_t + \phi_t\right),\\
&\nu_i \sim N\Big(0, \frac{1}{\tau_\nu}\Big),\\
&\mathrm{u}_i|\{\mathrm{u}_j; j \neq i\}, \tau_\mathrm{u} \sim N\Big(\frac{1}{\mathcal{N}_i}\sum_{j=1}^{n} a_{ij}\mathrm{u}_j, \frac{1}{\mathcal{N}_i\tau_\mathrm{u}}\Big),\\
&\gamma_t|\gamma_{t-1}, \gamma_{t-2} \sim N(2\gamma_{t-1} + \gamma_{t-2}, \sigma^2),\\
&\phi_t \sim N(0, 1/\tau_\phi),
\end{aligned} \tag{6}$$

where a_{ij} is 1 if the area's i and j are neighbors, and 0 otherwise. CM_i represents the closeness CC_i or the betweenness BC_i. The $\mathrm{base}_{i,t}$ is given by:

$$\begin{aligned}
\mathrm{base}_{i,t} =& 1 + \mathrm{sEI}_i + \mathrm{sRET}_i + \mathrm{aSH}_i + \mathrm{sNPI}_i +\\
&\mathrm{sMugL3M}_{i,t} + f_1(\mathrm{aAMI}_i) + f_2(\mathrm{Month}_t).
\end{aligned} \tag{7}$$

The models are fitted using the Integrated Laplace Approximation (INLA) implemented in the R package INLA [18]. The model selection is performed using two selection criterions based on the deviance. First, we use the deviance information criterion (DIC), proposed by Spiegelhalter et al. [19]. The DIC is defined as:

$$\mathrm{DIC} = D(\bar{\theta}) + 2p_D, \tag{8}$$

where $D(\bar{\theta})$ is the deviance using the posterior mean of the parameters, and p_D is the effective number of parameters. As the posterior marginal distributions of some hyperparameters might be highly skewed, especially the precisions, INLA evaluates the DIC at the posterior mode of the hyperparameters. For the latent field, INLA uses the posterior mean [20]. We used also the Watanabe-Akaike information criterion (WAIC) [21]. This criterion is based on the data partition and is closely linked to the Bayesian leave-one-out cross-validation. The WAIC is considered to be an improvement on the DIC criterion [22].

To assess the predictive performance of the models, the Root Mean Squared Error (RMSE) and the Weighted Absolute Percentage Error (WAPE) are calculated using an out-of-sample data set. As before, if $\mathrm{y}_{i,t}$ denotes the realization in postal code i and in month t, and $\hat{\mathrm{y}}_{i,t}$ denotes the forecast in the same postal code and in the same month, then the forecast error is given by $\mathrm{e}_{i,t} = \mathrm{y}_{i,t} - \hat{\mathrm{y}}_{i,t}$. The RMSE is given by Equation (5), and the WAPE is given by Equation (9) defined as follows.

$$\mathrm{WAPE} = \frac{\sum_{i=1}^{N}\sum_{t=1}^{T} \mid \mathrm{e}_{i,t} \mid}{\sum_{i=1}^{N}\sum_{t=1}^{T}\mathrm{y}_{i,t}}. \tag{9}$$

IV. Results

In this section, we first present the results of the centrality measures. Then, we will discuss the results of the two models including these centrality measures as covariates.

A. Centrality measures

As discussed in Section III-A, in practice it is computationally very expensive to calculate the exact betweenness and closeness scores. In general, these can be estimated by setting up a buffer zone using a cut-off distance d and calculating these centrality measures by considering only the paths at a shorter length than d. Using historical data, the average speed per street segment was calculated and five different time cut-offs were used. Segments that are reachable within one to five minutes are used to calculate the centrality measures. Note that these averages make sense because the centrality measures are calculated for the whole city and not for each area separately.

The betweenness and the closeness on the street segment level using a cut-off of four minutes are illustrated in Figure 3 and Figure 4, respectively. The corresponding average betweenness and closeness per area are illustrated in Figure 5 and Figure 6, respectively. Figures 3 and 4 show a wide red road running from top to bottom. This road corresponds with the A10, which is the ring road of Amsterdam. Figure 3 also shows that the roads with high betweenness correspond to the main access roads within this district. Figure 4 reveals that the roads within the areas situated on the right-hand side of the A10 have a higher closeness in general. This part of the city was built mainly before the Second World War [23] and has a higher density due to enclosed building blocks creating a

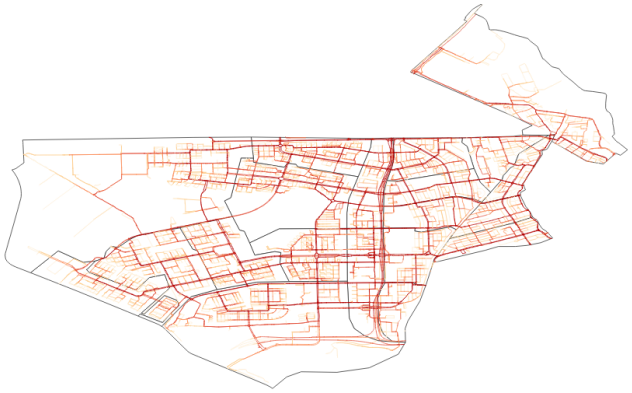


Figure 3. Betweenness of the street segments in Amsterdam West. The betweenness is calculated using the average speed on the street segment and a time threshold of four minutes.

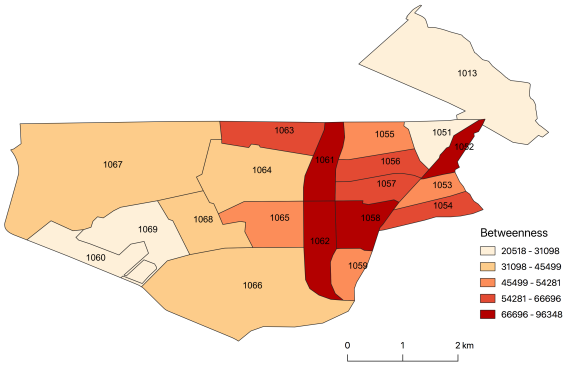


Figure 5. Average betweenness per postal code.

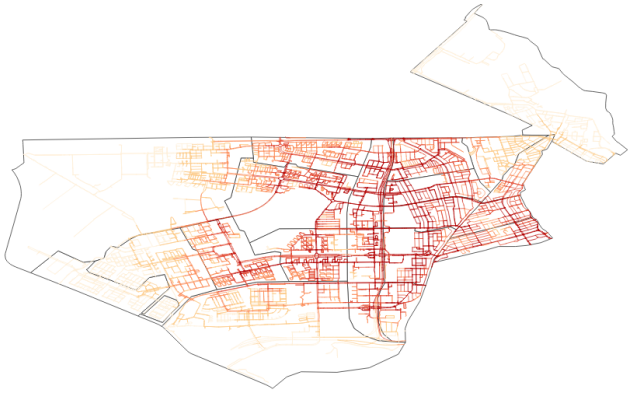


Figure 4. Closeness of the street segments in Amsterdam West. The closeness is calculated using the average speed on the street segment and a time threshold of four minutes.

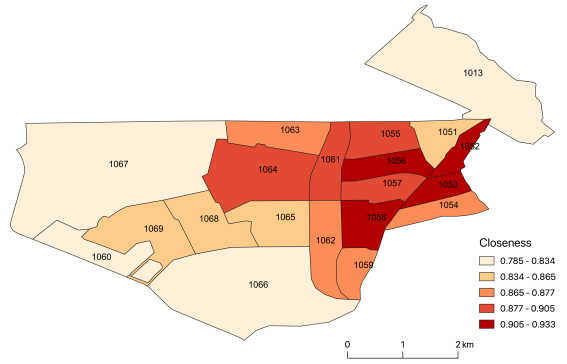


Figure 6. Average closeness per postal code using a threshold of four minutes.

more finely meshed network of roads when compared to the left-hand side of the ring road. This part was built after the Second World War and is characterized by a lower density due to more open building blocks with an emphasis on more green areas and better enclosure of the residential area via main access roads. The blank areas in the district correspond with green areas, such as parks, lakes and agricultural land.

B. GAMM including centrality measures

Adding a centrality measure to the GAMM model results in a better prediction based on the RMSE. The RMSE of the GAMM model without centrality measure was about 4.5519 and as can be seen from Table I, extending the model with the betweenness or the closeness results in a generally lower RMSE. It is noteworthy that the closeness leads to better predictions when using lower thresholds (lower or equal 3 min); see Figures 7 and 8. If the threshold is four minutes or higher including the betweenness in the model results in better predictions. This can be explained by the average time an offender might need to flee from the scene of the crime on a residential street to the nearest main access road. In this case, the closeness describes the number of different routes the offender can take during his flight. Within 4 or 5 minutes, the offender can be traveling on the main access road in order to create as much distance as possible from the crime scene.

The results in the area with the postal code 1067 differ from the other areas. Including the closeness and betweenness does not improve the model, the error on the other hand increased. Taking a closer look at this area revealed that this area mainly consists of green areas with few roads. With less alternative routes available, the closeness gives a higher error.

When looking at the other areas, it is possible to say that the

Table I. ROOT MEAN SQUARED ERROR (RMSE) VALUES FROM FITTING THE GAMM MODEL WITH CLOSENESS AND BETWEENNESS USING DIFFERENT THRESHOLDS.

Model	1 min	2 min	3 min	4 min	5 min
GAMM + CC	4.5297	4.5323	4.5366	5.5437	4.5478
GAMM + BC	4.5562	4.5497	4.5405	4.5279	4.5326

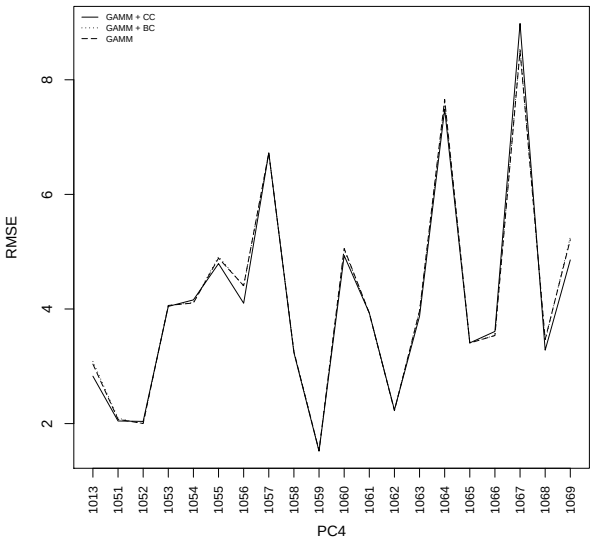


Figure 7. RMSE per PC4 base on an out-of-sample for the GAMM model, the GAMM + CC and the GAMM + BC using a threshold of 1 minute.

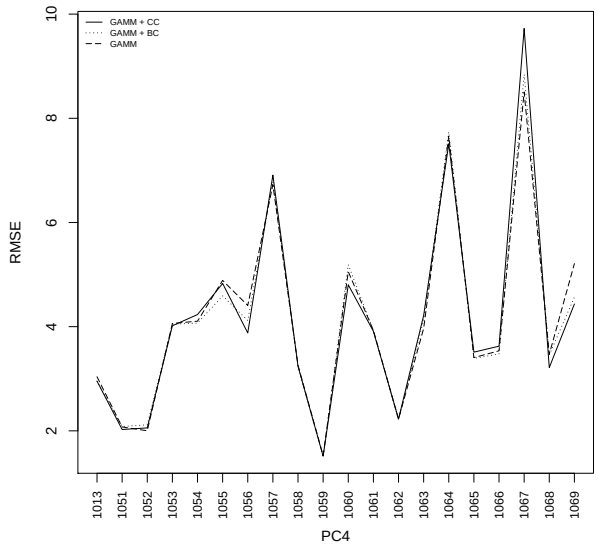


Figure 8. RMSE per PC4 base on an out-of-sample for the GAMM model, the GAMM + CC and the GAMM + BC using a threshold of 4 minutes.

building density influences the effectiveness of the centrality measures on the models. In areas with a lower density, the centrality measures have almost no influence on the outcomes, whereas in the urban areas with a high building density adding the centrality measures to the model improves the outcomes of the model.

Most studies use betweenness as a centrality measure, however, these studies focus on social networks. Given our results, we believe that the closeness is a better centrality measure for modeling crime based on small geographic areas. However, as shown there is a difference in effectiveness of this centrality measure related to the building density of the area.

C. Space-time model including centrality measures

In this section, we will present the results from fitting the space-time models with the betweenness and the closeness using the different thresholds. First, we fit the models including all covariates and compare their DIC and WAIC values. Table II shows that the model with the closeness centrality with a threshold of five minutes provides the lowest DIC and WAIC values. However, the differences remain small. Looking at the estimated posterior mean values and their 95% credible intervals (CI), we can see that the betweenness centrality and the average number of educational institutions are not important as the zero lies within the 95% CI. In contrast, the closeness centrality seems important regardless of the threshold used, see Figure 9.

Based on the DIC and the WAIC values, the best performing model is selected. This model has a closeness measure with a threshold of five minutes and includes all covariates, except the average number of educational institutions and the average number of persons generating income.

The estimated parameters of the best-obtained model on logarithmic scale are presented in Table III. From this table, we can see that the number of retail stores in the postal code, the number of mugging incidents, and the average closeness with a threshold of five minutes have a positive effect on burglaries. The number of households with a single parent has a negative effect on burglaries. To assess the exact effect of these covariates on residential burglaries, we converted the posterior distributions from the logarithmic scale to the original scale of the data. Then we calculated the posterior mean and the 95% credible intervals on the original scale. From Table IV,

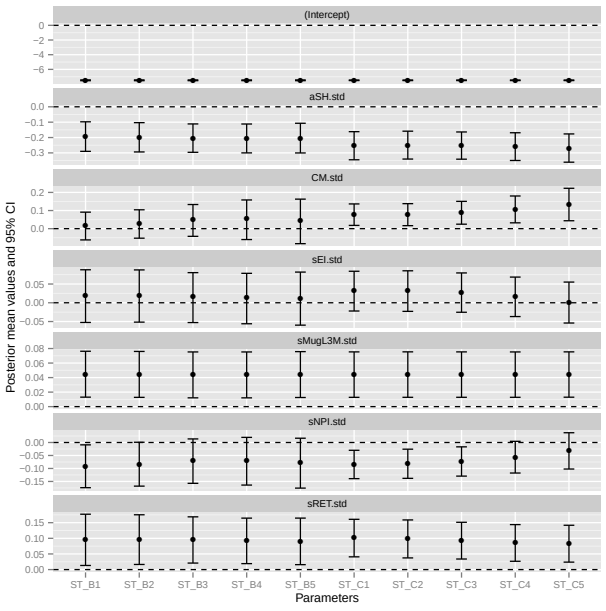


Figure 9. Posterior mean values and 95% credible intervals for all regression parameters obtained using the different models.

Table II. DIC and WAIC values of the models including all covariates.

Criterion	BC1	BC2	BC3	BC4	BC5	CC1	CC2	CC3	CC4	CC5
DIC	3991.795	3991.877	3992.044	3992.049	3991.902	3991.591	3991.725	3991.507	3991.445	3991.434
WAIC	4019.198	4019.266	4019.338	4019.357	4019.263	4018.372	4018.613	4018.255	4018.217	4018.217

Table III. The posterior mean and 95% credible intervals of the fixed effects on logarithmic scale.

Estimate	Mean	Std.dev	0.025 quantile	0.975 quantile
Intercept	-7.486	0.019	-7.525	-7.448
sRET.std	0.086	0.028	0.030	0.142
aSH.std	-0.279	0.044	-0.365	-0.190
sMugL3M.std	0.043	0.016	0.013	0.074
avgCloseness5.std	0.160	0.033	0.094	0.225

Table IV. The posterior mean and 95% credible intervals of the fixed effects on the natural scale.

Estimate	Mean	Std.dev	0.025 quantile	0.975 quantile
Intercept	0.00056085	1.07431e-05	0.00053957	0.000582493
sRET.std	1.09048	0.0303216	1.03077	1.15139
aSH.std	0.757577	0.0331817	0.694364	0.825984
sMugL3M.std	1.04458	0.0161383	1.01311	1.07653
avgCloseness5.std	1.17414	0.038402	1.09919	1.25157

we can conclude that an increase in one unit in the number of retail stores, the number of muggings and in the closeness is associated with an increase of 9.05%, 4.46%, and 17.41%, respectively, in the risk of burglary. Among all covariates, the closeness has the most impact on the risk of residential burglaries. In contrast, the average number of households with a single parent results in a decrease in the risk of burglaries.

After taking account of the covariates, the residual relative risk of each area ($\exp(\xi)$) and their posterior probability of exceeding one ($\Pr(\xi_i > 0 \,|\, \mathrm{y})$) are represented in Figures 10 and 11, respectively. As can be seen from Figure 10, the postal code area 1057 has the highest relative risk of burglaries compared to the whole Amsterdam West. This area also has a higher probability of excess risk on burglaries next to the postal codes 1063 and 1051. These results are in line with our expectations. The postal code area 1057 is a pre-war build neighborhood along the main excess road. These neighborhood houses have many problems such as a higher poverty rate [24] and a higher pollution rate [25]. According to OIS Amsterdam (2017), the adjacent neighborhood inhabits many crime-suspects [24].

The posterior temporal trends are illustrated in Figure 12. This figure represents the structured and the unstructured temporal components modeled dynamically by means of an RW(2) model and an exchangeable Gaussian prior, respectively. As can be seen from this figure, a clear seasonal pattern in burglaries can be observed with a higher risk of burglaries between September and February in general. As expected, peaks are observed for December and January. The same figure reveals that June and July are the months with the lowest risk of burglaries. It is also noteworthy to mention that the second year (2010) clearly has a lower risk of burglaries during the dark months compared to the other years. The temporally unstructured effect, $\exp(\phi_t)$ fluctuates around one.

Finally, we assessed the predictive performance of the models based on out-of-sample data and compared the results to the best obtained GAMM models with the one of the space-time model with a closeness considering a threshold of four

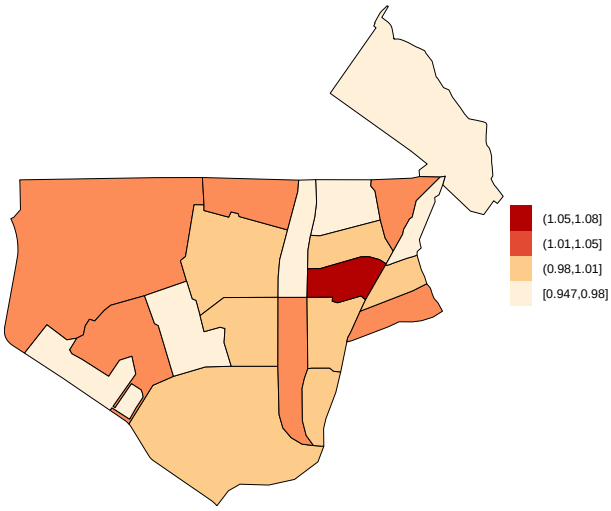


Figure 10. Posterior mean of the residual relative risk for each PC4.

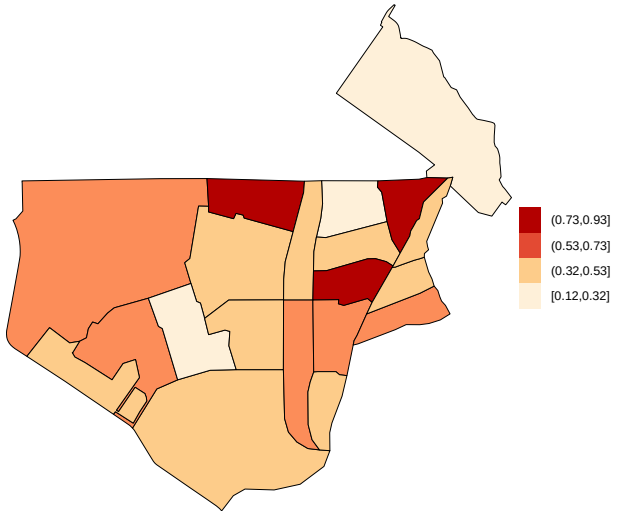


Figure 11. Posterior mean of the residual excess risk for each PC4.

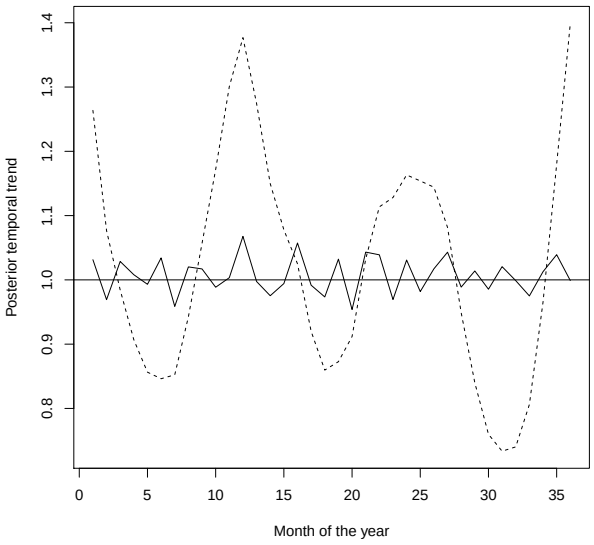


Figure 12. Model with closeness with a threshold of 4 minutes.

minutes. This concerns the GAMM models with the closeness considering a threshold of 1 minute and the GAMM model with the betweenness considering a threshold of 4 minutes. To compare the models we used the RMSE and the WAPE accuracy measures. First, we compared the total performance of the models, then we compared the performance of the models for each postal code separately. The space-time model including the closeness centrality (ST.CC4) clearly results in lower RMSE and WAPE errors compared the GAMM models, see Table V. When we compare the predictive performance of

Table V. Predictive performance of the models based on out-of-sample data.

Model	RMSE	WAPE (%)
ST.CC4	3.98	30.47
GAMM.CC1	5.05	36.82
GAMM.BC5	5.35	38.34

the models for each postal code, we can see that ST.CC4 performs clearly better compared to the other models, especially in PC4 1056, see Figures 13 and 14.

V. Conclusion and Future work

During this research, we have tried to determine the influence of accessibility of the street network within small urban areas on residential burglary by applying the centrality measures closeness and betweenness. Given the results in the literature, it is natural to study this problem in the context of GAMM models. We have found that adding the centrality measures as a variable to the GAMM model has improved the performance of this model as can be concluded from the lower RMSE. Furthermore, we have shown that there is a relation between the different conceptions in urban design over time and residential burglary. Our results show that the pre-world War II neighborhoods suffer from more residential burglary than the neighborhoods built after the Second World War.

Rather contrastingly, differences in the performance of the two centrality measures were found when using the GAMM model. Closeness as a centrality measure gives better predictions when taking into consideration a threshold smaller than 4 minutes. If the threshold is 4 minutes or larger, the betweenness gives better predictions. This contradiction disappears when modelling the spatial and temporal effects explicitly. In that case, the model with the closeness centrality with a threshold

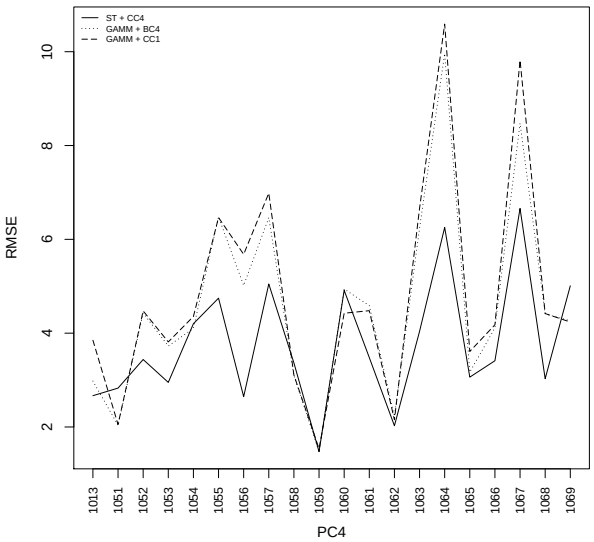


Figure 13. RMSE per PC4 based on out-of-sample data for the ST model including CC5, GAMM + CC1 and the GAMM + BC4.

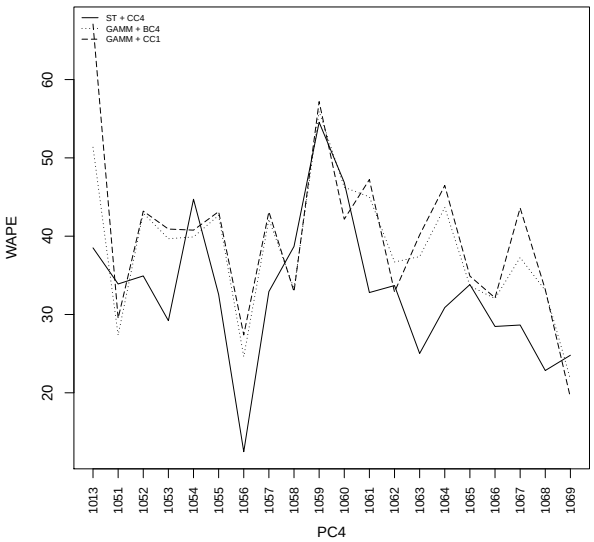


Figure 14. WAPE per PC4 based on out-of-sample data for the ST model including CC5, GAMM + CC1 and the GAMM + BC4.

of five minutes provides the best results, and even beats the best GAMM model.

Our study has shown that there is a relationship between the conceptions in urban design and crime. Neighborhoods built under a certain conception of urban design tend to have a higher risk of residential burglary, which can be explained by how the public space is designed. Further research is necessary to confirm this hypothesis.

REFERENCES

[1] M. Mahfoud, S. Bhulai, and R. van der Mei, "Forecasting burglary risk in small areas via network analysis of city streets," in Proceedings of the 7th International Conference on Data Analytics. IARIA, 2018, pp. 109–114.

[2] P. J. Brantingham, P. L. Brantingham et al., Environmental criminology. Sage Publications Beverly Hills, CA, 1981.

[3] R. Wortley and M. Townsley, Environmental criminology and crime analysis. Taylor & Francis, 2016, vol. 18.

[4] P. Brantingham and P. Brantingham, "Crime pattern theory," in Environmental criminology and crime analysis. Willan, 2013, pp. 100–116.

[5] D. Weisburd, E. R. Groff, and S.-M. Yang, The criminology of place: Street segments and our understanding of the crime problem. Oxford University Press, 2012.

[6] D. Weisburd, S. Bushway, C. Lum, and S.-M. Yang, "Trajectories of crime at places: A longitudinal study of street segments in the city of Seattle," Criminology, vol. 42, no. 2, 2004, pp. 283–322.

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

[8] P. Crucitti, V. Latora, and S. Porta, "Centrality measures in spatial networks of urban streets," Physical Review E, vol. 73, no. 3, 2006, p. 036125.

[9] T. Davies and S. D. Johnson, "Examining the relationship between road structure and Quantitative Criminology, vol. 31, no. 3, 2015, pp. 481–507.

[10] T. P. Davies and S. R. Bishop, "Modelling patterns of burglary on street networks," Crime Science, vol. 2, no. 1, 2013, p. 10.

[11] M. Mahfoud, S. Bhulai, R. van der Mei, and D. Kardaras, "Spatio-temporal modeling for residential burglary," in Proceedings of the 6th International Conference on Data Analytics. IARIA, 2018, pp. 59–64.

[12] D. Liao and R. Valliant, "Variance inflation factors in the analysis of complex survey data," Survey Methodology, vol. 38, no. 1, 2012, pp. 53–62.

[13] A. Benjamin, G. Chartrand, and P. Zhang, The fascinating world of graph theory. Princeton University Press, 2015.

[14] R. H. Baayen, D. J. Davidson, and D. M. Bates, "Mixed-effects modeling with crossed random effects for subjects and items," Journal of memory and language, vol. 59, no. 4, 2008, pp. 390–412.

[15] S. Wood and F. Scheipl, gamm4: Generalized additive mixed models using mgcv and lme4, 2014, r package version 0.2-3. [Online]. Available: http://CRAN.R-project.org/package=gamm4

[16] J. Besag, J. York, and A. Mollié, "A bayesian image restoration with two applications in spatial statistics ann inst statist math 43: 1–59," Find this article online, 1991.

[17] H. Rue and L. Held, Gaussian Markov random fields: theory and applications. CRC press, 2005.

[18] H. Rue, S. Martino, and N. Chopin, "Approximate bayesian inference for latent gaussian models by using integrated nested laplace approximations," Journal of the royal statistical society: Series b (statistical methodology), vol. 71, no. 2, 2009, pp. 319–392.

[19] D. J. Spiegelhalter, N. G. Best, B. P. Carlin, and A. Van Der Linde, "Bayesian measures of model complexity and fit," Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 64, no. 4, 2002, pp. 583–639.

[20] M. Blangiardo and M. Cameletti, Spatial and spatio-temporal Bayesian models with R-INLA. John Wiley & Sons, 2015.

[21] S. Watanabe, "Asymptotic equivalence of bayes cross validation and widely applicable information criterion in singular learning theory," Journal of Machine Learning Research, vol. 11, no. Dec, 2010, pp. 3571–3594.

[22] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, Bayesian data analysis. CRC press Boca Raton, FL, 2014, vol. 2.

[23] G. Amsterdam. De groei van amsterdam. [Online]. Available: https://maps.amsterdam.nl/bouwjaar/?LANG=nl (2018)

[24] "Ois amsterdam," https://www.ois.amsterdam.nl/publicaties/de-staat-van-de-stad-amsterdam/?50159, last access date: 1 March, 2019.

[25] "Wonen in amsterdam 2017: leefbaarheid," https://www.ois.amsterdam.nl/publicaties/de-staat-van-de-stad-amsterdam/?50159, last access date: 1 March, 2019.

Deploying Artificial Intelligence to Combat Disinformation Warfare

Identifying and Interdicting Disinformation Attacks Against Cloud-based Social Media Platforms

Barry Cartwright
School of Criminology
Simon Fraser University
Burnaby, Canada
Email: bcartwri@sfu.ca

George R. S. Weir
Department of Computer & Information Sciences
University of Strathclyde
Glasgow, Scotland, UK
Email: george.weir@strath.ac.uk

Richard Frank
Karmvir Padda
School of Criminology
Simon Fraser University
Burnaby, Canada
Email: {rfrank; karmvir_padda}@sfu.ca

***Abstract*—Disinformation attacks that make use of Cloud-based social media platforms, and in particular, the attacks orchestrated by the Russian "Internet Research Agency," before, during and after the 2016 U.S. Presidential election campaign and the 2016 Brexit referendum in the U.K., have led to increasing demands from governmental agencies for technological tools that are capable of identifying such attacks in their earliest stages, rather than identifying and responding to them in retrospect. This paper reports on the interim results of an ongoing research project that was sponsored by the Canadian government's Cyber Security Directorate. The research is being conducted by the International CyberCrime Research Centre (ICCRC) at Simon Fraser University (Canada), in cooperation with the Department of Information and Computer Sciences at the University of Strathclyde (Scotland). Our ultimate objective is the development of a "critical content toolkit," which will mobilize artificial intelligence to identify hostile disinformation activities in "near-real-time." Employing the ICCRC's Dark Crawler, Strathclyde's Posit Toolkit, Google Brain's TensorFlow, plus SentiStrength and a short-text classification program known as LibShortText, we have analyzed a wide sample of social media posts that exemplify the "fake news" that was disseminated by Russia's Internet Research Agency, comparing them to "real news" posts in order to develop an automated means of classification. To date, we have been able to classify posts as "real news" or "fake news" with an accuracy rate of 90.7%, 90.12%, 89.5%, and 74.26% using LibShortText, Posit, TensorFlow and SentiStrength respectively.**

Keywords-Social media; disinformation warfare; machine learning.

I. INTRODUCTION

This paper elaborates on an earlier paper on the subject of fighting disinformation warfare through the use of artificial intelligence, presented at the Tenth International Conference on Cloud Computing, GRIDs, and Virtualization, held in Venice, Italy, in May 2019 [1]. As observed in our earlier conference paper, the key challenges facing law enforcement agencies, intelligence agencies, cybersecurity personnel and business owners-operators worldwide are how to monitor and effectively respond to dynamic and emerging cybersecurity threats, with increasing attention being paid to hostile disinformation activities in Cloud-based social media platforms [1]. To illustrate, Cambridge Analytica, through an app that it developed, managed to scrape data from over 80 million Facebook pages worldwide. This information was in turn used to micro-target voters through Facebook advertisements, which were premised upon the demographic profiles and known political leanings of those voters, in turn based upon information which had been extracted with the help of the Cambridge Analytica app [2], [3]. In July 2018, Facebook was fined £500,000—the maximum allowable under British law—for its mishandling of data in the Cambridge Analytica scandal [4]. In July 2019, the US Federal Trade Commission fined Facebook five billion USD for its failure to protect user privacy [5]. The nexus between Cambridge Analytica, WikiLeaks, and Russian interference in the 2016 U.S. Presidential election remained under investigation by the U.S. Congress as recently as the Summer of 2019 [6].

According to a 2017 Intelligence Community Assessment, prepared jointly by the Central Intelligence Agency (CIA), the Federal Bureau of Investigation (FBI) and the National Security Agency (NSA), a number of other Cloud-based social media, including Twitter and Instagram, have also been implicated as (possibly unaware) participants in the hosting and dissemination of disinformation attacks associated with the Russian "Internet Research Agency" (IRA) [7]. According to Special Counsel Robert Mueller's recently released report into Russian interference in the U.S. Presidential election [8], Facebook and Twitter accounts targeted certain groups, such as Blacks (through the Blacktivist Facebook page), Southern Whites (through the

Patriototus Facebook page), and the right-wing anti-immigration movement (through the Secured Borders Facebook page), as well as through Twitter feeds such as @TEN_GOP (which falsely claimed to have a connection to the Republican Party of Tennessee), and @America_1st (an anti-immigration account). In the U.K., the "fake news"—which primarily stoked Islamaphobic and anti-immigration passions—made extensive use of Twitter, employing Twitter handles such as ReasonsToLeaveEU, or #voteleave [9], [10], [11], [12]. Evidence also indicates that the Russian IRA maximized use of social media bots in their 2016 assaults on the U.S. Presidential election and the U.K. Brexit referendum [9], [10], [13], [14], thus amplifying the content in order to reach and influence a much wider audience. More will be said about Russian involvement in disinformation warfare in Section II of this paper, wherein we present our literature review.

Our research, sponsored by the Canadian government's Cyber Security Cooperation Program, and conducted by the International CyberCrime Research Centre at Simon Fraser University, in cooperation with the Department of Information and Computer Sciences at the University of Strathclyde, involves the development of a tool for identifying hostile disinformation activities in the Cloud. This research project commenced with a dataset of 2,946,219 "fake news" Twitter messages (tweets), identified as emanating from the Russian IRA. Later, our research came to include datasets that combined both Twitter and Facebook "fake news" messages, eventually including a number of comparator datasets of "real news" messages, plus a potential "training" dataset for machine learning that we have not yet explored fully, the latter consisting of a wide range of "real news" and "fake news" [15]. It is anticipated that the knowledge generated by this research will establish the foundation for more advanced work, eventually culminating in the construction of a "critical content toolkit," which will aid governmental agencies in the rapid and accurate pinpointing of disinformation attacks in their very early stages.

The research team has several years of collaborative experience in collecting and analyzing data from online extremist forums, child pornography websites, social media feeds and the Dark Web. Our previous experience in data classification has demonstrated that we are able, through automation, to achieve predictive accuracy in the 90-95% range when it comes to detecting the nuanced text found in extremist content on the Web [1], [16], [17], [18]. From this background, we have a methodology that is applicable to the analysis and classification of data from Cloud-based social media platforms. In the past, our predictive accuracy was accomplished by applying a combination of technologies, including the Dark Crawler, SentiStrength, and Posit [1]. For the present study, we have employed the Dark Crawler, Posit, SentiStrength, TensorFlow, and LibShortText. Additional information on these research tools is provided in Section III, wherein we set out our methodology. Our research results are reported in Section IV, and elucidated further in Sections V and VI, wherein we discuss our results, set out the directions that our future research endeavors are expected to take, and present our interim conclusions.

II. LITERATURE REVIEW

As noted in our introductory comments in Section I, Cloud-based social media platforms have come under increasing scrutiny for permitting hostile foreign actors to manipulate public opinion through the creation of fake social media accounts that disseminate false information, often referred to as "fake news" [19], [20], [21]. This false information, or fake news, can be broken down into two broader categories: misinformation and disinformation. The less sinister of the two, misinformation, is simply inaccurate or false information. Misinformation may be based upon a genuine misapprehension of the facts, as opposed to having been created with any particular intention of deceiving or manipulating people [22], [23], [24]. Disinformation, on the other hand, especially when employed by hostile foreign actors, is information that is created and spread intentionally, for the express purpose of deception and manipulation of public opinion [22], [24], [25].

The activities of Russia's IRA during the 2016 U.S. Presidential election would be a prime example of a disinformation campaign mounted by a hostile foreign actor [10], [13], [26], [27]. In February 2108, U.S. Special Counsel Robert Mueller, duly appointed to investigate Russian interference in the U.S. election, obtained a grand jury indictment against the IRA (which was bankrolled by Yevgeniy Prigozhin, often referred to as "Putin's chef"), plus Prigozhin's American-based companies Concord Management and Consulting LLC and Concord Catering as well as Prigozhin himself, along with a dozen Russian "trolls" who were employed by Prigozhin's IRA. The indictment stated that the accused had "operated social media pages and groups designed to attract U.S. audiences" in order to advance divisive issues and create dissension, falsely claiming that those pages and groups were controlled by American activists [9], [28].

The dozen Internet "trolls" who were described in the indictment obtained by Mueller belonged to an identifiable sub-group of a much larger workforce, comprised of 1,000 or more Russian trolls, all employed by Prigozhin's IRA [29], [30], [31]. These IRA employees, working in a building in the Russian city of St. Petersburg, toiled around the clock in two, 12-hour shifts (a day shift and a night shift), with the objective of fomenting division, distrust, dissent, and hostility within and between targeted groups in the American populace [32], [33], [34]. In particular, it has been said that these IRA trolls were instructed to spread disinformation that would buttress Donald Trump's campaign for the U.S. Presidency, and at the same time, undermine the campaign of Hillary Clinton [7], [30], [34], [35].

The Computational Propaganda Project, a multi-national project housed primarily in the Oxford Internet Institute, has reported that 19 million identifiable "bot" accounts tweeted in support of Trump or Clinton in the week leading up to the 2016 Presidential election, with 55.1% of those in favour of Trump, and only 19.1% in favour of Clinton [36], [37], [38]. The evident disparity in Twitter support would seem difficult

to account for, other than in terms of highly-orchestrated and deliberate political interference, given that Hillary Clinton received 65,844,954 votes, compared to 62,979,879 votes for Donald Trump [39].

A 2017 study by Zannettou et al. revealed that 71% of these "fake" accounts were created prior to the 2016 election [34]. In fact, the 2017 Intelligence Community Assessment, prepared jointly by the CIA, FBI and NSA, indicated that Russian operatives had begun researching U.S. electoral processes and election-related technology and equipment as early as 2014, two years prior to the election, and that the Prigozhin-led IRA had started advocating on behalf of Donald Trump's candidacy as early as 2015, one year prior to the election [7]. Zannettou et al. reported that 24 accounts were created on July 12, 2016, approximately one week before the Republican National Conference (at which Donald Trump was formally nominated as the Republican candidate for the 2016 Presidential election) [34]. The study also found that the Russian Internet trolls attempted to mask their disinformation campaign by adopting different identities, changing their screen names and profile information, and in some cases, deleting their previous tweets. In their examination of tweets posted between January 2016 and September 2017, for example, Zannettou et al. found that 19% of the accounts operated by IRA trolls changed their screen names as many as 11 times, and unlike other Twitter users, often deleted their tweets in large batches, in order to start again with a clean slate [34].

Much has been said about the use of social media bots in the U.S. Presidential election and the U.K. Brexit referendum [9], [10], [13], [14]. Briefly, the transfer and transformation of information on the Internet is not accomplished by people, but rather, by algorithms—scripts which convert mathematical expressions into instructions for the Internet [37]. The Internet Relay Chat System would be an early example of where bots were being used to manage and regulate social interaction on the Internet. These bots, which still comprise an integral part of the architecture of Cloud-based social media sites such as Twitter and Facebook, are capable of interacting with Internet users, answering simple questions, and collecting data. More sophisticated bots can also be deployed to crawl the Web, scrape social media sites for data, parse the information gleaned, and even manipulate political opinion [37]. Some online stores/companies, such as AliExpress, use these AI bots for managing the extensive help systems on their site. If you have an issue, you chat with the bot. The Cambridge Analytica app, which attracted so much negative attention to Facebook in the aftermath of the 2016 U.S. Presidential election and the 2016 U.K. Brexit referendum, would be an example of an algorithm that was designed for the express purpose of collecting and evaluating behavioral data such as the likes, dislikes and political proclivities of the Facebook users whose data it harvested [40].

To express it differently, the bots (robots) described herein are Cloud-based social media accounts that are controlled by software, rather than by real people. These social media bots are estimated to comprise between 5-9% of the overall Twitter population, and to account for approximately 24% of all tweets [41]. Users of social media may spend considerable time liking and disliking bots, sometimes arguing with (or even flirting with) bots, all the while thinking that they are interacting with a real person. Stories that "go viral"—i.e., that rise to the top of Twitter feeds—are often pushed there by these social media bots through manipulation of the social media platform's algorithms [41].

The main problem with "fake news" is that its consumers tend to accept what they read at face value. According to The Pew Research Center, 12% of Americans get their news from Twitter [42]. Of those who use the platform regularly, close to 60% depend on Twitter as their source of news [26], [42], [43]. With respect to the type of "fake news" that is the subject of this present study, it can be said that the frequent tweeting and re-tweeting by bots leads to ever-increasing exposure, resulting in an "echo chamber effect" [33]. To add to the mix, evidence suggests that many individuals are unable to distinguish between factual and non-factual content found on Twitter and Facebook [44], [45]. Indeed, according to a Stanford University study, far too many are inclined to accept images or statements that they come across on social media at face value, without questioning the source of those images or statements, or for that matter, asking whether they even represent what they purport to represent [9], [44], [45].

Russian interference in the U.S. Presidential election and the U.K Brexit referendum has been well documented, and has been the subject of considerable governmental and academic research, e.g., [7], [8], [9], [10], [12], [13], [14], [20], [27], [34]. However, such Russian interference is by no means restricted to the U.S. and the U.K. To illustrate, in 2019, the European Commission—along with the European External Action Service and other EU institutions and member states—released a progress report on its Action Plan Against Disinformation. According to the Commission's progress report, evidence gathered throughout 2018 and early 2019 confirmed ongoing disinformation activities originating from Russian sources, believed to be undertaken for the purpose of influencing voter preferences and suppressing voter turnout in the EU Parliamentary elections [23], [46].

Moreover, a recent study of Canadian Twitter data suggests that Russian trolls were behind "fake news" stories that attempted to stoke fear and distrust between Muslims and non-Muslims following the 2017 shooting deaths of six worshippers at a mosque in Quebec City, leading to renewed concerns that Russian trolls might attempt to interfere in the Fall 2019 Canadian federal election [47]. With this in mind, the research team recently collected a sample of 3,500 tweets from hashtags such as #TrudeauMustGoToJail, #TrudeauMustGo, and #TrudeauMustResign, some of which were suspected of containing "fake news" which was intended to influence the outcome of the 2019 Canadian federal election. In addition, we are currently focusing our efforts on collecting Canadian-specific "fake news" Facebook items, from *The Buffalo Chronicle-Canadian Edition*, Canadian Truth Seekers, Million Canadian March, The Canadian Defence League, The Silent Majority Canada, The Angry Cousin, Proud Canadians, and Canada Proud.

This Facebook dataset presently consists of 3,737 discrete data items.

Russian-orchestrated disinformation campaigns are long-standing in nature. The Kremlin reportedly founded a school for bloggers as far back as 2009, apparently foreseeing the long-range possibilities of utilizing Cloud-based social media in political influence campaigns [48]. In fact, Russian disinformation activities have been documented in the Czech Republic and Slovakia as far back as 2013 [49], and in the 2014 election in the Ukraine, which itself followed shortly after Russia's annexation of the Crimean Peninsula [50], [51]. This is not to suggest that all known disinformation campaigns have been launched by Russia, or that such campaigns have been restricted only to those countries mentioned above. Using a combination of qualitative content analysis, secondary literature reviews, country case studies and consultations with experts, a 2019 inventory compiled by the Oxford Internet Institute found evidence of disinformation campaigns in 70 different countries around the world, including but by no means limited to Armenia, India, Malaysia, Mexico, The Philippines, Saudi Arabia, The United Arab Emirates and Venezuela [52]. In many cases, however, the campaigns are spreading pro-party or pro-government propaganda, or attacking the political opposition, and in the absence of evidence to the contrary, they could well be mounted by local (rather than foreign) actors. That said, countries other than Russia, such as China and Saudi Arabia, are believed to be making increasing use of disinformation campaigns beyond their own borders [53]. In any event, the focus of this present paper is the Russian-orchestrated attacks on the 2016 U.S. Presidential election.

A number of researchers have mobilized artificial intelligence in an effort to counter the type of disinformation warfare employed by Russia during the 2016 U.S. Presidential election and the 2016 U.K. Brexit referendum. In 2017, Darren Linvill and John Walker (from Clemson University) gathered and saved vast numbers of Twitter and Facebook postings (prior to their removal from the Internet by the respective social media platforms), thereby preserving the evidence and making the data available to the academic, cyber-security and law enforcement communities for further study [54]. Linvill and Walker investigated the Twitter handles used by the Russian IRA, both qualitatively and quantitatively, breaking them down into troll accounts and bot accounts, and into right trolls, left trolls, fear mongers, news feeders and hash tag gamers. Our research team has made extensive use of the IRA's Twitter and Facebook postings that were gathered, saved and made available by Linvill and Walker. In 2017, William Yang Wang from the University of California at Santa Barbara released his LIAR dataset, which included 12,836 statements labeled for their subject matter, situational context, and truthfulness, broken down into training, validation and test sets, along with instructions for automatic fake news detection [15]. In addition, William Wang reported that the open source software toolkit, LibShortText, developed by the Machine Learning Group at National Taiwan University, had been shown to perform well when it came to short text classification [15], [55]. The dataset provided by Linvill and Walker, and the suggestion by William Wang about using LibShortText, have both been used by us to inform and refine the machine learning and automated analysis processes described in the following sections on Methodology and Research Results.

In his above-mentioned study using the LIAR dataset, William Wang found that when it came to automatic language detection, a hybrid, convoluted deep neural network that integrated both meta-data and text, produced superior results to text-only approaches [15]. We are employing a somewhat similar approach to that of William Wang, in that we are using a combination of deep neural networks (Tensor Flow) [56], a text-reading program (Posit) that also produces meta-data or mark-up [57], [58], and the LibShortText program developed by the Machine Learning Group at National Taiwan University [55]. Employing techniques of machine learning and natural language processing, a 2018 study of Twitter troll activity in the 2016 U.S. Presidential election found that a model blending measurements of "precision" and "recall" failed to accurately classify 34% of troll posts, suggesting that such models could not be relied upon to identify and screen out fake news [48]. However, a 2019 paper, entitled "Defending Against Neural Fake News," reports on the development of GROVER, a computer model that can both generate and detect neural fake news, premised on the notion that while most fake news is presently generated by humans, the fake news of the future may be generated by machines. The authors of this paper report additionally that they have been able to discriminate fake news with an accuracy of 92%, as opposed to the more standard 73% accuracy exhibited by other fake news discriminators [59]. Our research results, reported below, come much closer to approximating those described in this 2019 study.

Some researchers have sought to identify disinformation campaigns by employing "bot" detection, instead of relying upon automated text-reading software. Essentially, much of what may be regarded as "fake news" is thought to be spread and/or amplified by the use of bots [37], [40], [41]. Thus, the goal of bot detection is to discriminate accurately between bot-generated and human-generated activity on social media. Morstatter, Carley and Liu, for example, have proposed what they call "a new approach to bot detection," again blending measurements of "precision" and "recall" [41], similar to the measurements employed in the above-mentioned 2018 study of Twitter troll activity in the 2016 U.S. Presidential election. In their 2019 study, Morstatter et. al found that they could successfully classify bot activity in 76.55% of instances. Another approach, outlined by Gorodnichenko, Pahm and Talavera, identifies suspected bot activity in the Brexit referendum and the U.S. Presidential election, by measuring such variables as when the Twitter account was first created, the number of tweets per day, the timing of daily and hourly tweeting, and the number of tweets containing the same content. This is premised on the understanding that many of these bot accounts are created for the purpose of spreading disinformation, and that bots send more messages than humans, at all times of the day (even when human activity is much reduced), and that they re-send the same messages

over and over again [60]. Using this approach, Gorodnichenko et. al found that they could classify bots and non-bots with 90% accuracy. While we have not employed bot detection in this present study, it is an avenue that we plan to explore as our work progresses.

III. METHODOLOGY

Our analysis of "fake news" messages posted by the Internet Research Agency (IRA), before, during and after the 2016 U.S. Presidential election, employed a variety of approaches, including collection of IRA posts and "real news" datasets using the Dark Crawler, plus machine analysis of large samples of the posts using Posit, TensorFlow, SentiStrength and LibShortText [55].

Although this research was geared primarily toward machine learning and the development of an artificial intelligence tool to aid in the rapid and accurate pinpointing of disinformation attacks in their early stages, we also conducted qualitative, textual analysis of 1,250 of the IRA "fake news" Twitter posts, to probe into the alleged degree of Russian involvement in the disinformation campaign [8], [13], [26], [31], assess the veracity of claims that the posts were intended to support Donald Trump's campaign for the U.S. Presidency whilst simultaneously undermining the campaign of Hillary Clinton [7], [30], [34], [35], [36], [37], [38], and investigate the degree to which some of the posts were grounded in "real news," rather than in what is commonly referred to as "fake news" [9], [19], [20], [21], [22].

A. Research Tools

The Dark Crawler is a custom-written, web-crawling software tool, developed by Richard Frank of Simon Fraser University's International CyberCrime Research Centre. This application can capture Web content from the open and Dark Web, as well as structured content from online discussion forums and various social media platforms [61] [62], [63]. The Dark Crawler uses key words, key phrases, and other syntax to retrieve relevant pages from the Web. The Crawler analyzes them, and recursively follows the links out of those pages. Statistics are automatically collected and retained for each webpage extracted, including frequency of keywords and the number of images and videos (if any are present). The entire content of each webpage is also preserved for further manual and automated textual analysis. Content retrieved by the Dark Crawler is parsed into an Excel-style worksheet, with each data element being identified and extracted. In previous studies of this nature, we have employed this same procedure to collect over 100 million forum posts from across a vast number of hacking and extremist forums, to be used for later analysis [61], [62].

The Posit toolkit was developed by George Weir of the Department of Computer and Information Sciences at the University of Strathclyde. Posit generates frequency data and Part-of-Speech (POS) tagging while accommodating large text corpora. The data output from Posit includes values for total words (tokens), total unique words (types), type/token ratio, number of sentences, average sentence length, number of characters, average word length, noun types, verb types, adjective types, adverb types, preposition types, personal pronoun types, determiner types, possessive pronoun types, interjection types, particle types, nouns, verbs, prepositions, personal pronouns, determiners, adverbs, adjectives, possessive pronouns, interjections, and particles, for a total of 27 features in all [9], [57], [58]. This process generates a detailed frequency analysis of the syntax, including multi-word units and associated part-of-speech components.

As it was configured for previous studies, the Posit toolkit created data on the basis of word-level information; thus, the limited content of the Russian IRA tweets that we were examining meant that many of the original features might have zero values. For this particular research project, Posit was extended to include analysis of character-level content, to assist with the analysis of short texts. To this end, the system supplemented the standard word-level statistics, generating an additional 44-character features for each instance of text data. These new features included quantitative information on individual alphanumeric characters, plus a subset of special characters—specifically, exclamation marks, question marks, periods, asterisks and dollar signs. The extension of Posit to embrace character-level as well as word-level data maintained the domain-neutral nature of Posit analysis. As a result of this extended Posit analysis, each data item (tweet) was represented by a set of 71 features, rather than the usual twenty-seven [1].

TensorFlow, originally developed by the Google Brain Team, is a machine learning system that employs deep neural networks [56], inspired by real-life neural systems. The learning algorithms are designed to excel in pattern recognition and knowledge-based prediction by training sensory data through an artificial network structure of neurons (nodes) and neuronal connections (weights). The network structure is usually constructed with an input layer, one or more hidden layers, and an output layer. Each layer contains multiple nodes, with connections between the nodes in the different layers. As data is fed into this neural system, weights are calculated and repeatedly changed for each connection [63].

Textual content was further analyzed using SentiStrength, which assigns positive or negative values to lexical units in the text [61], [62], [64]. This value is a measure that provides a quantitative understanding of the content of information being found online—specifically, the extent to which positive and negative sentiment is present. The program automatically extracts the emotions or attitude of a text and assigns them a value that ranges from "negative" to "neutral" to "positive."

In the case of Posit and SentiStrength, the resultant data were input to the Waikato Environment for Knowledge Analysis (WEKA) data analysis application [65]. For SentiStrength, the data, comprised of the noun keywords for each textual item, along with the associated sentiment score and the manual classification for that page, then employed WEKA's standard J48 tree classification method with ten-fold cross-validation. In this cross-validation, 10% of the data was hidden, and conditions were sought that would split the remaining 90% of the dataset in two, with each part having as many data-points as possible belonging to a single

class. Accuracy of the tree was then considered relative to the hidden 10% of the data. This process was repeated 10 times, each time with a different hidden 10% subset. WEKA produced a measure of how many of the pages were correctly classified.

For Posit, we applied the standard J48 tree WEKA classification method, plus the Random Forest classification method [65], [66], both with ten-fold validation (as described above). WEKA then produced a measure of how many of the text items were correctly classified. In the Random Forest method, classification trees (of the type found in WEKA) are independently constructed, by employing a bootstrap sample of the entire dataset, and then relying on a simple majority vote for predictive purposes, rather than relying on earlier trees to boost the weight of successive trees [67].

Finally, to better enhance the machine learning process, and to improve our future classification accuracy, we turned our attention to the LibShortText toolkit, as William Yang Wang of the University of California at Santa Barbara had indicated that this tool produced superior results when it came to the accurate classification of shorter items of text, such as tweets or brief Facebook posts [15]. LibShortText, an open source software package developed by the Machine Learning Group at National Taiwan University, is said to be more efficient and more extensible than other generalized text-mining tools, allowing for the conversion of short texts into sparse feature vectors [68].

B. Research Sample

At the beginning of the project, the research team downloaded a dataset of 2,946,219 Twitter messages (tweets) from git.hub, which had been posted online by fivethirtyeight.com. This dataset of tweets was collected and assembled by the aforementioned professors from Clemson University, Darren Linvill and Patrick Warren [54]. These tweets were described as originating from the Russian IRA, also referred to in common parlance as the Russian troll factory, a hostile foreign agency that was believed to have intentionally interfered in the 2016 U.S. Presidential election and the 2016 U.K. Brexit referendum [7], [8], [9], [10], [13], [14], [26], [27], [28], [29], [30], [31], [33].

As the various approaches used in our research (i.e., qualitative analysis, Posit, TensorFlow, SentiStrength and LibShortText) were designed to read English text, a decision was made to extract only those entries that were labeled as being "English," so in the process, we excluded languages such as Albanian, Bulgarian, Catalan, Croatian, Dutch, Estonian, French, German, Italian, Russian, Ukrainian, Uzbek, Vietnamese. As a consequence, 13 new Excel spreadsheets were created, with 2,116,904 English-speaking tweets remaining in the dataset following the removal of all non-English tweets.

Having acquired the Russian (IRA) Twitter data, we then sought a second Twitter dataset that would allow us to develop a classification model based upon comparison between "real news" and what has often been referred to simply as "fake news" [19], [20], [21], [22], [24], [25], [30]. To this end, we analyzed the textual content from the full set of IRA tweets (or "fake news") using Posit, in order to identify frequently occurring terms, and more specifically, nouns. The resultant "keyword" list was used by the International CyberCrime Research Centre's Dark Crawler, in order to retrieve a set of matching "real news" Twitter posts from legitimate news sites.

The Dark Crawler harvested Twitter feeds maintained by more "traditional," mainstream news sources, such as the *Globe and Mail*, *CBC News*, *CTV News*, the *BBC*, the *New York Times*, the *Daily Telegraph*, the *Wall Street Journal*, *Asahi Shim-Bun*, *Times of India*, the *Washington Post*, the *Guardian*, and *Daily Mail Online*, collecting tweets posted between the beginning of January 2015 and the end of August 2018 (within the approximate time frame of the IRA tweets). Tweets from the "real news" dataset that were posted after August 2018 were removed, as the data from the IRA tweets did not extend beyond that time frame. We started with 90,605 tweets, but with the removal of 10,602 tweets that had been posted in late 2018 and early 2019, we were left with 80,003 individual cases or tweets that exemplified "real" or "legitimate" news sources. For the purpose of Posit, SentiStrength and LibShortText analysis, a further research decision was made to random sample both datasets, creating two datasets of equal size, each consisting of 2,500 tweets, or roughly .001% of the larger "fake news" dataset, and 3% of the "real news" dataset. Unique identifiers were assigned to each of the data items, to ensure a means of fixed reference.

A somewhat different sample was assembled for the TensorFlow analysis, because for TensorFlow to operate effectively, a larger dataset is desirable. To achieve this, we combined the 2,116,904 English-speaking "fake news" tweets that remained (following the removal of all non-English cases) with the 90,605 "real news" tweets that were downloaded by the Dark Crawler (prior to removal of tweets that extended beyond the time frame of the IRA activities). This dataset was supplemented with 2,500 Facebook messages posted by the IRA, plus an additional "real news" set of Facebook items. Thus, a large dataset of 2,709,204 million tweets and Facebook posts was analyzed in TensorFlow following the merging of these multiple datasets.

For SentiStrength analysis and LibShortText analysis, we consolidated four smaller, 2,500 item datasets into one larger, 10,000 item dataset. This larger, 10,000 item dataset consisted of the above-mentioned set of 2,500 randomly sampled "fake news" Twitter messages derived from the dataset of 2,946,219 Twitter messages collected by Clemson University professors Linvill and Warren, the above-mentioned set of 2,500 randomly sampled "real news" Twitter messages derived from the 90,605 tweets collected by the Dark Crawler from traditional, mainstream news sources, plus 2,500 "fake news" posts from Facebook and 2,500 comparator "real news" posts [9]. The 2,500 "fake news" Facebook messages that formed part of this larger, 10,000 item dataset were posted on Facebook by Russia's Internet Research Agency between 2015 and 2017, and were

again collected and made available by Clemson University professors Linvill and Warren [54]. To secure a source of "real news" data for our comparison with the Facebook "fake news," we obtained a second "real news" dataset, this time of actual Facebook posts made available at github.com by data scientist Max Woolf. The data that we retrieved was originally comprised of 164 sets of publicly accessible Facebook status posts. From these status posts, we manually selected Facebook IDs that appeared to be associated with traditional news sources, such as *USA Today*, the *New York Times*, and *CNBC*. From these, we randomly selected 2,500 "real news" Facebook posts to serve as our comparator dataset [9].

C. Data Analysis

1) Qualtiative Textual Analysis

Qualitative textual analysis was conducted on the first 1,250 messages appearing in the above-mentioned set of 2,500 randomly sampled "fake news" Twitter posts, these 2,500 posts having been derived (winnowed down) from the dataset of 2,946,219 Twitter posts collected by Clemson University professors Linvill and Warren [54]. To express it differently, one half of the 2,500 randomly sampled "fake news" Twitter posts were read and classified manually. This process involved two experienced qualitative researchers, sitting side-by-side, reading each of the posts together, in many cases several times, until agreement on an appropriate classification was reached. Where there was disagreement, or where there was insufficient information upon which to arrive at a conclusion, the classification was designated as "undetermined." The classification for each of the 1,250 Twitter posts was recorded carefully in an Excel spreadsheet, with both researchers watching over each other's shoulder, to ensure the integrity of the data entry.

In a number of cases, the qualitative classification process included a Google search, to determine whether or not the content of the post was entirely fictional, partially true, or mostly true (i.e., grounded in "real news"). The two researchers were already familiar with some of the "real news" events that appeared and re-appeared in these posts, having conducted previous qualitative research on a different "fake news" dataset of messages emanating from the Russian Internet Research Agency, in this other case investigating fake Facebook accounts, rather than Twitter hashtags [9].

2) Posit

Following the creation and cleansing of the datasets, we extracted features from the texts using Posit, which is designed to generate quantitative data at the level of word and part-of-speech content of texts. Posit analysis was applied to each of the 5,000 tweets in order to produce a 27-item feature list for each tweet. This was supplemented by an additional feature, to indicate the "real" or "fake" characteristic of each tweet.

Previous research has indicated that Posit's domain-independent meta-data can be effective as a feature set for use in such text classification tasks [16], [17], [18]. In the present study, however, the target textual data was made up entirely of tweets. These have a limited maximum length of 280 characters, so they are inherently short and contain relatively few words. To illustrate, one of the tweets said only: "@realDonaldTrump True," while another said only: "Stay strong! #MAGA." With this shorter content in mind, Posit was extended such that the system supplemented the standard word-level statistics by generating an additional 44-character features for each instance of text data. As noted above, the result of this extended Posit analysis was that each data item (tweet) was represented by a set of 71 features, rather than the standard 27 features [1], [9].

The list of tweet features generated by Posit was formulated as an arff file format, suitable for direct input to the Waikato Environment for Knowledge Analysis (WEKA) data analysis application [65]. In WEKA, we applied the standard J48 tree classification method and the Random Forest classification method [66], [67], both with ten-fold validation. WEKA produced a measure of how many of the tweets were correctly classified.

3) TensorFlow

In this project, TensorFlow was used for processing the data with a Deep Neural Network (DNN) [56], [63]. A large dataset was initially fed into TensorFlow, in order to conduct DNN learning. The DNN results either updated an existing model or created a new model. TensorFlow then compared the same data against the constructed DNN model, and utilized that model to predict the category for each data entry.

In order to build an initial TensorFlow model, a large dataset of 2,709,204 million tweets was created by merging multiple datasets. The more data that could be collected for training a model, the better the accuracy should be. However, the individual data files were inconsistent, since they were collected from various online resources, and were formatted in very different ways. Thus, in the process of combining them into a single dataset, we opted for Microsoft Access, which allowed us to create a large, unified database table. All of the datasets were merged into this Access database, after which a class label column "*category*" was defined, denoting whether the data represented "fake" or "real" news.

The model was evaluated for its accuracy in predicting class values for the "fake" or "real" news category. To simplify the analysis, we decided to build our DNN model based on the content of the 2,709,204 tweets, without any further pre-processing. The DNN model used was a TensorFlow Estimator.DNNClassifier.

In the early stages of experimentation, we employed TensorFlow with default settings for the parameters pertaining to the number of partitions, epochs, layers, learning rate, and regularization. With respect to regularization, data was partitioned into groups according to the order in which it appeared in the dataset. Thus, if the majority of "fake news" appeared in the beginning of the dataset, it would be difficult to maintain consistent accuracy when conducting X-fold cross validation. To overcome this issue, the data was randomized as it became partitioned. Furthermore, each partition maintained the same data across all X-fold cross validation tests, so that the accuracy of the results could be compared properly.

With TensorFlow, epochs refer to the number of times the dataset is processed during training. The greater the number of epochs, the higher the accuracy tends to be. The learning rate determines the rate at which the model converges to the local minima. Usually, a smaller learning rate means it that it would take longer for the model to converge at the local minima [69]. With a larger learning rate, the model would get closer to this convergence point more quickly. The values for these parameters—i.e., the number of partitions, epochs, layers, learning rate, and regularization (L1 & L2)—were then tested to identify an optimal set of parameter values.

4) SentiStrength

For the SentiStrength analysis [61], [64], the general sentiment of the consolidated, 10,000 item dataset was first calculated without using any keywords. As there were no immediate trends identified between the "fake news" and "real news" items, keywords were generated using the top 100 nouns that appeared in the 10,000 posts. This produced a 100 x 10,000 matrix, against which we ran various algorithms in WEKA [56], again in an effort to distinguish between the "fake news" and "real news" items. This analysis included an examination of WEKA's decision trees, Naïve Bayes, BayesNet, and Multilayer Perceptron, the latter being a deep neural net algorithm, similar to that found in TensorFlow, in that it employs neurons, weights, and hidden layers [56], [63], [70], [71].

5) LibShortText

As noted earlier, LibShortText is an open source software package, developed by the Machine Learning Group at National Taiwan University. The use of LibShortText was recommended in a 2018 paper by William Yang Wang of the University of California at Santa Barbara, wherein he also described (and provided access to) his benchmark LIAR dataset. This LIAR dataset, which included 12,836 statements labeled for their subject matter, situational context, and truthfulness, was broken down into training, validation and test sets, and accompanied by instructions for automatic fake news detection [15]. For this particular research project, we employed LibShortText, but did not make use of William Wang's LIAR dataset. We plan to return to the LIAR dataset for purposes of additional machine training on short text items, as we progress in the development of our critical content toolkit.

LibShortText is said to be more efficient and more extensible than other generalized text-mining tools, allowing for the conversion of short texts into sparse feature vectors, and also for micro- and macro-level error analysis [59]. For our research project, we built a model using the default settings that came with the LibShortText software. We employed the "$ python text-train.py trainfile" command which generated a "trainfile.model" for our given training file ("trainfile"). Working with this previously built model, we set out to predict the classification labels of the test set, or "trainfile" using the instructions: "$ python text-predict.py -f testfile trainfile.model predict_result," followed by "Option -f" to overwrite the existing model file and predict_result. The LibShortText software is available for free download from the National Taiwan University at: https://www.csie.ntu.edu.tw/~cjlin/libshorttext/.

IV. RESEARCH RESULTS

A. Qualitative (Textual) Analysis

As discussed in Section III (above), qualitative textual analysis was conducted on the first 1,250 messages that appeared in the set of 2,500 randomly sampled "fake news" tweets posted by the Internet Research Agency (IRA). These tweets were read and classified manually by two experienced qualitative researchers, who read the posts together, and jointly assigned an appropriate classification for each individual tweet. The classification for each of these 1,250 tweets was recorded in an Excel spreadsheet.

One of the patterns that became apparent early in the process was that close to one-third (31.92%, n = 399) of the 1,250 tweets that were read manually consisted of what could best be described as "apolitical chatter" (see Table I, below). Tweets that were classified as apolitical chatter did not appear to be re-circulating "real news," either to targeted or untargeted audiences. Moreover, they did not appear to be supporting the candidacy of either Donald Trump or Hillary Clinton, nor were they overtly attempting to advance divisive issues, create dissension, or otherwise undermine democratic processes. Examples of apolitical chatter would include tweets such as: "#TerribleHashTagIdeas MostRomanticKissAfterVomiting," "#ToFeelBetterI get high," "#ThingsYouCantIgnore Christmas sales," and "#DontTellAnyoneBut I prefer sex with the lights on."

There are a number of possible explanations when trying to account for the presence of so much apolitical chatter. One explanation could be that the Russian IRA simply did not get its money's worth when hiring some of these Internet trolls. To illustrate, whichever troll (or group of trolls) was responsible for the IRA hashtag BOOTH_PRINCE generated a disproportionate number of apolitical tweets, for example: "#ThereIsAlwaysRoomInMyLifeForDrake," "#tofeelbetteri think about Iphone 7S" and "#MyAmazonWishList FEMBOTS." On the other hand, the hashtag BOOTH_PRINCE also produced some Anti-Clinton tweets, such as: "A plastic fork too cut a steak #ThingsMoreTrustedThanHillary," and another referring sarcastically to then-Democratic President Barack Obama, and to Hillary Clinton's opponent in the Democratic primaries, Bernie Sanders: "#ObamasWishList Bernie, actually." Thus, it seems more likely that the political messaging was intentionally interspersed with a lot of apolitical chatter, in an effort to make these IRA-sponsored hashtags and tweets appear more akin to the type of discourse typically found on social media.

Another possible explanation for the high number of messages that we found necessary to classify as "apolitical chatter" would be the difficulties we encountered when trying to retrieve the videos or twitter feeds that were linked to these IRA tweets. While the tweets themselves seemed relatively innocuous, at least on the surface, it is conceivable

that they may have been targeted toward specific, pre-identified groups, and may have included links to political messaging and political advertising, as has been suggested by various other observers [7], [8], [9], [10],[11], [12], [13], [14], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38].

TABLE I. RESULTS OF QUALITATIVE, TEXTUAL ANALYSIS

Classification	Frequency	%
Apolitical Chatter	399	31.92
Pro-Trump/Anti-Clinton	328	26.24
Undetermined	171	13.68
Real News	152	12.16
Pro-Clinton/Anti-Trump	81	6.48
Racist	57	4.56
Helpful Advice	35	2.80
Anti-Racist	27	2.16
Total	**1250**	**100.00**

A similar explanation might apply to the 171 tweets (13.68%) where we could not arrive at a classification decision, and where the intent of those tweets thus ended up being classified as "undetermined" (see Table I, above). To illustrate the difficulties in the classification process, amongst the 1,250 tweets that we read manually, there were seven that contained a link to "#RejectedDebateTopics," and three that contained a link to "#BetterAlternativesToDebates," both of which were reportedly subsidiary hashtags created by the Russian troll army [72], [73]. An examination of what little remained of the Internet content from these two hashtags suggested that they were mostly pro-Trump or anti-Clinton, but there were embarrassing images of—and embarrassing statements about—Trump as well as Clinton; therefore, it was impossible to determine with certainty to which variety of Internet content the readers were being directed.

Despite the pro-Trump bias of "#RejectedDebateTopics," one of the Russian IRA tweets that we examined that was linked to this hashtag could safely be classified as being Pro-Clinton/Anti-Trump, because it irreverently asked: "Which Eastern European country will Trump's next wife come from?" However, another one from the same hashtag, #RejectedDebateTopics asked: "which Kardashian is least likely to have an STD?" The Kardashians were supporters of Hillary Clinton during the 2016 U.S. Presidential election, which allowed us to classify this second tweet as pro-Trump, anti-Clinton. On the other hand, one tweet that was linked to #RejectedDebateTopics asked simply: "who killed the Kennedy's ?" [sic], presumably referring to the Kennedy family, famous for producing Democratic President John F. Kennedy, Democratic Presidential candidate Robert Kennedy, and Democratic Senator Ted Kennedy. Another tweet, this time linked to #BetterAlternativesToDebates, talked about "smoke signaling using Bill's special cigars," perhaps referring to Bill Clinton, the former Democratic President of the United States, and the husband of Hillary Clinton. It is conceivable that both of the above-mentioned tweets were pro-Trump or anti-Clinton, but it was agreed that there was insufficient information to arrive at a decision in this regard. Thus, to err on the side of caution, both were assigned to the "undetermined" category.

Many of the 1,250 tweets that were read and classified manually were actually engaged in the re-circulation (or regurgitation) of "real news" stories, and thus ended up being classified as "real news." These "real news" tweets accounted for 12.16% (n = 152) of the dataset. This is comparable to our findings in a companion research project that involved the analysis of a sample of 2,500 Russian-generated Facebook posts, wherein we learned that 13.5% of the Facebook posts were based to one extent or another on recognizable, named entities, such as people, places, and specific dates or events [9]. Many of the "real news" tweets that are reported in this present paper were innocuous news stories and did not appear to be either pro-Trump or pro-Clinton. Examples of such tweets include: "The Latest: Sister says crash victim was retired from FBI," "San Antonio loses another popular radio star after on-air announcement #art," "University of Texas-Arlington police consider roaming robot" and "Texas appeals court overturns ex-Baylor player's conviction." Again, there are a couple of possible explanations, one being that the Russian IRA did not get its money's worth from these trolls, the other being that there was a concerted effort to make these IRA-sponsored hashtags and tweets look more like the typical discourse found on social media, the latter being the more likely of the two. To express it differently, they could be described as "background noise," intended to obfuscate the real motivation behind this online activity.

There were also 35 tweets that appeared to be providing "helpful advice." Examples of such "helpful advice" tweets would include: "Free And Cheap Things To Do In #London 27-28 January 2017 More Info Here," "How to Get Magazines to Review Your Music," and "Q&A: "What are trans fats and why are they unhealthful? #news." Again, it is entirely possible that there was a concerted effort to make these IRA-sponsored hashtags and tweets look more like the typical discourse found on social media, by throwing in some "chaff" with the "wheat," or that they were put in simply to create background noise. In any event, tweets that were classified as providing "helpful advice" were few and far between, comprising only 2.8% of the 1,250 "fake news" messages that were read and classified manually.

Of the 1,250 tweets analyzed manually, the 328 tweets that overtly supported the presidential candidacy of Donald Trump, or that were blatantly anti-Clinton, comprised the second largest group overall (after "apolitical chatter"), and vastly outnumbered the 81 tweets that supported the candidacy of Hillary Clinton (or in the alternative, were anti-Trump), by a ratio of four to one (see Table I, above). An example of a pro-Trump tweet that attempted to cover all of the main talking points of Trump and his supporters in one shot would be: "OUR MAN–He will get us out of the last 8 year mess against our Religion, Jobs, Illegal & Refugee Overkill, Homeless Vets & more–NEED HIM!." Other exemplars of unabashedly Pro-Trump tweets would be: "I just spoke to @realDonaldTrump and he fully supports my plan to replace Obamacare the same day we repeal it. The time to act is now," "Because all legal citizens vote Trump!

#VoteTrump," and "Trump is making manufacturing great again."

Tweets that were intended to undermine the campaign of Hillary Clinton, whilst simultaneously buttressing the campaign of Donald Trump, were in abundance: "Hillary Clinton to Fundraise with Anti-Christ (No, not Obama a different one)," "@SheriffClarke: If Trump made me his FBI Director I would be arresting Hillary Clinton today. #Comey," and "BREAKING: Julian Assange Is Back! And He Just Put The Nail In Hillary's Coffin." The latter tweet was clearly referring to the hack of the Democratic National Committee's email server by Russia's General Main Staff Intelligence Unit (the GRU), and to the subsequent leak of potentially embarrassing internal emails on WikiLeaks [7], [8]. Another tweet, again related to WikiLeaks, stated that: "WikiLeaks CONFIRMS Hillary Sold Weapons to ISIS... Then Drops Another BOMBSHELL! Breaking News." One anti-Clinton tweet targeted her daughter, saying: "Chelsea Clinton has received another award, this time for a day's worth of work." Yet another targeted Clinton's husband, saying: "Remember when Trump got a $1 million birthday gift from Saudi Arabia? Oh wait, that was Bill Clinton!"

There were 81 tweets (6.48% of the 1,250 tweets that were manually classified) that arguably supported Hillary Clinton, or in the alternative, talked negatively about Donald Trump, but most were not so blatantly in favour of one candidate over the other as the tweets supporting Donald Trump, or those attacking Hillary Clinton. To illustrate, one tweet that was classified as being pro-Hillary, anti-Trump, announced that: "Keith Ellison Plays Race Card, Claims Trump Brings White Supremacy to the White House." This tweet could also have been classified as "real news," in that it was reporting about Democrat Congressman Keith Ellison, who was running in 2018 for the Attorney General position in the state of Minnesota [74]. This was evidently some time after the 2016 Presidential campaign, but it has been widely reported that the Russian IRA carried on with its pro-Trump, pro-Republican, anti-Clinton, anti-Democrat agenda throughout 2017 and 2018 [75], [76]. The above tweet was "generously" classified as being anti-Trump, because it mentioned "Trump" and "White Supremacy" in the same breath. However, a closer examination of the news of the day might suggest that it could have been classified as pro-Trump, given that Ellison's invocation of the "race card" was viewed by some as a sign of desperation on his part. But to err on the side of caution, and in view of the oft-repeated claims by Donald Trump that alleged Russian interference in the 2016 Presidential election is "a hoax" or "fake news" [77], [78], this tweet was classified as being anti-Trump.

Another example of erring on the side of caution would be the following tweet: "BEHNA: ABSURD! Secret Service Agent Declares She Wouldn't Take A Bullet For Trump." This too was generously classified as being "anti-Trump," because it talked about a secret service agent who apparently would not perform her obligatory duties to protect the President, due to her personal animosity toward Donald Trump. It could just as well have been classified as "real news," because the story also appeared in mainstream news sources [79]. And it could arguably have been classified as "pro-Trump," because the sender of the tweet appeared to be saying that the behaviour of the secret agent was "absurd." However, we sought at all times to maintain a neutral stance in our qualitative textual analysis. In any event, if we had taken in-between messages such as these, that seemed at least on the surface to be saying something negative about Donald Trump, and classified them instead as "pro-Trump," then the four-to-one ratio of tweets in favour of Trump would have widened measurably.

Indeed, many of the tweets that were classified as pro-Clinton and/or anti-Trump could have gone either way or could have been classified as "undetermined" in their intent. The following tweet serves to illustrate this classification conundrum: "Donald Trump's Frog Meme 'SINISTER,' Clinton Campaign Warns." This tweet talked about a warning from the Clinton campaign concerning "sinister" activity on the part of Donald Trump and was thus classified as being pro-Clinton. However, it may well have been intended as a sarcastic "dig" toward Hillary Clinton and her team, or could even have been intended to direct the followers of the tweet to a video in which Donald Trump was "poking fun" at Clinton (this was not possible to verify, as the attachment has since been removed from the Internet, presumably because of the political fallout and furor following the detection of the Russian disinformation campaign).

This is not to say that there was a total dearth of pro-Clinton, anti-Trump messages. One example of a clearly pro-Clinton tweet would be: "#ImStillWithHer; She's #MyChoice #MyPresident #MyHero." Another example of a pro-Clinton tweet would be: "I think people also assume that folks who may vote for HRC won't push her. That couldn't be further from the truth." There were also a number of tweets that were clearly anti-Trump, such as: "The Latest: GOP senator says party has gone 'batshit crazy' #Texas," "Protesters in Texas seek release of Trump tax returns," and "Designer of Make America Great Again dress is an immigrant," not to mention "#anderr LOL : Mad Max Reveals THE EXACT MONTH Trump Will be Impeached." But such overtly pro-Clinton or anti-Trump messages were comparatively few and far between, and in many cases, had to be "teased out" of the dataset.

There were quite a few blatantly racist messages in the first 1,250 "fake news" tweets (4.56%, n = 57), some of which could arguably have been categorized as pro-Trump and anti-Clinton, as they mimicked Donald Trump's portrayal of Mexicans as criminals, drug traffickers and rapists [80], favoured his "Muslim ban" [81], supported his anti-immigration stance, and generally concurred with his description of Haiti, El Salvador and certain African nations as "shithole countries" [82], all of which was reportedly intended by Trump—and by the Russian IRA—to foster an atmosphere of distrust, divisiveness and fear with respect to immigrants and racial minorities, in order to "rile up" Trump's voter base [7], [8], [32], [33], [34]. Examples of anti-Mexican or Anti-Central American tweets include: "11 dumped from Rio Grande raft rescued by Border Patrol," and "A mayor was just shot dead in Mexico on the day after she took office." Messages targeting African-Americans were

also in evidence, with exemplars including such tweets as: "When a tall ass nigga, sees a short ass nigga w/ a tall girlfriend," or "Young Black folks keep saying they're not like the ancestors. And I keep saying that's the problem," or "They steal everything. Black folks have to be wiser." Anti-Muslim messaging could be found in abundance, in tweets such as: "'The Koran is a fascist book which incites violence. This book, just like Main Kampf [sic], must be banned.'- G. Wilders," or "5-year-old girl was raped by muslim immigrants and nobody's talking about that! #IsalmIsTheProblem," or "Did you know that Muslims are now allowed to have sex with slave woman even after their death?! #BanIslam." The overall thrust of these anti-Muslim, anti-immigration, anti-refugee messages is best encapsulated in the following tweet: "See those countless women and children? Neither do I https://t.co/tZOkWo7OjZ #banIslam #Rapefugees https://t.co/XoETkXAidV."

The above-mentioned racist messages, many of which clearly supported the Trump political agenda, were counter-balanced by approximately half as many anti-racist messages (2.16%, n = 27). Anti-racist tweets included the following: "We deserve to feel safe in our cars, our businesses, our parks, our homes and our churches. #BlackSkinIsNotACrime," "New Mexico Store in Trouble for Controversial Obama, Anti-Muslim Signs," and "34-year-old African-American man in Wisconsin brought 3 different documents to DMV & still couldn't get voter ID." Again, we imagine that the inclusion of this comparatively small number of anti-racist tweets was likely intended to offset the overtly pro-Trump, anti-immigration bias that was in evidence throughout the dataset, and to make these Russian-generated hashtags and tweets look more like the typical discourse found on social media. Quite apart from that, ostensibly anti-racist tweets such as these could actually have been crafted in such a way as to stoke fear and distrust among immigrants and racial minorities (thereby suppressing their vote), with comments about their overall lack of safety, and the difficulties that they could expect to experience when attempting to register to vote.

The findings of our qualitative textual analysis of the first 1,250 messages that appeared in the set of 2,500 randomly sampled "fake news" tweets posted by the Russian IRA strongly support the oft-reported conclusion that these Twitter feeds were intended to buttress the Presidential campaign of Donald Trump, and to stoke dissension, distrust, anger and fear in the American voting populace [7], [8], [9], [13], [26], [28], [35], [36], [37], [38]. Although it was sometimes difficult to tease out, we also adduced evidence that the hashtags and tweets were indeed generated by Russian sources, which runs counter to the White House narrative about "the Russian hoax" [77], [78]. To illustrate, one tweet announced: "Nikolai Nikolaevich Ge - _ Russian realist painter famous for his works on historical and religious motifs - was born today." Another noted that "The Russian band Leningrad is bringing its smashing program titled '20 Years for Joy' to the US," while still another reported that "For the first time since 2010, the MoscowState University has returned to the top 100 of QS World University Rankings global ranking." While such news stories might have held some interest for the Russian Internet trolls, it seems unlikely that they would have been of particular interest to American users of Twitter.

B. Posit Results

As noted earlier, the Posit toolkit generates frequency data and Part-of-Speech (POS) tagging while accommodating large text corpora. The Posit analysis produced a feature set with corresponding values for each of the 5,000 tweets, that is, the 2,500 "fake news" tweets and the 2,500 "real news" tweets. The feature set was loaded into WEKA as a basis for testing the feasibility of classification against the predefined "fake" and "real" news categories. Using the "standard" set of 27 Posit features—and the default WEKA settings with 10-fold cross validation—the J48 and Random Forest classifiers gave 82.6% and 86.82% correctly classified instances respectively. The confusion matrix for the latter performance is shown in Table II, below.

TABLE II. CONFUSION MATRIX FOR POSIT: 27 FEATURES (RANDOM FOREST: DEFAULT WEKA SETTINGS)

n=5,000	**Predicted: NEGATIVE**	**Predicted: POSITIVE**	
Actual: NEGATIVE	2,190	310	2,500
Actual: POSITIVE	340	2,160	2,500

As indicated previously, Posit was enhanced with an additional 44 character-based features, resulting in a total of 71 features, rather than the standard 27 features [1]. This was done in order to address the fact that tweets have a limited maximum length of 280 characters; thus, they are inherently short, and contain relatively few words. Using this extended feature set on the 5,000 tweets—and the default WEKA settings with 10-fold cross validation—the J48 and Random Forest settings classifiers gave 81.52% and 89.8% correctly classified instances respectively. The confusion matrix for the latter performance is shown in Table III, below.

Changing the number of instances (trees) from the default value of 100 to 211 in Random Forest provided a boost to the level of correctly classified instances to 90.12%. The confusion matrix for this performance is shown in Table IV, below.

TABLE III. CONFUSION MATRIX FOR POSIT: 71 FEATURES (RANDOM FOREST: DEFAULT WEKA SETTINGS)

n=5,000	**Predicted: NEGATIVE**	**Predicted: POSITIVE**	
Actual: NEGATIVE	2,266	234	2,500
Actual: POSITIVE	276	2,224	2,500

TABLE IV. CONFUSION MATRIX FOR POSIT: 71 FEATURES (RANDOM FOREST: INSTANCES AT 211 IN WEKA SETTINGS)

n=5,000	**Predicted: NEGATIVE**	**Predicted: POSITIVE**	
Actual: NEGATIVE	2,269	231	2,500
Actual: POSITIVE	263	2,237	2,500

Our best performance results (90.12%) were obtained from the Posit classification using the 71-feature set with Random Forest (instances at 211). The "detailed accuracy by class" for this result is shown in Table V.

TABLE V. DETAILED ACCURACY BY CLASS FOR BEST POSIT RESULT

Class	TP Rate	FP Rate	Precision	Recall	F-Measure
NEGATIVE	0.908	0.105	0.896	0.908	0.902
POSITIVE	0.895	0.092	0.906	0.895	0.901
Weighted Avg.	0.901	0.099	0.901	0.901	0.901

Following these classification efforts using Posit, the two datasets (the real and fake tweets) were subjected to further analysis. The aim at this point was to determine whether any obvious characteristics in the data might skew the classification results. Several checks were made on the complexion of the two sets of data, focusing particularly on their relative content in terms of words and characters—since these features are the focus of the Posit analyses.

A comparison was made of the length of tweets in the two datasets. This revealed some differences in the distribution of tweets according to their length measured in words (Figure 1). Generally, distribution by length in words for the real news tweets rose above the curve for distribution by length in words for the fake tweets. Conceivably, this would ease the challenge of discriminating between the two datasets.

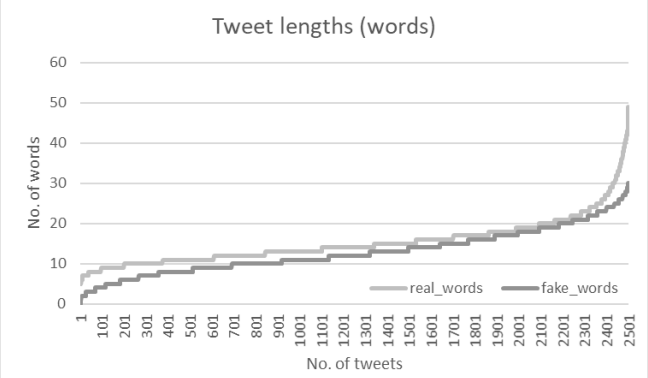


Figure 1. Comparison of Tweet Lengths (Words)

Since tweets are limited to 280 characters in length, a natural contrast was to consider the relative lengths of tweets by number of characters. This comparison (Figure 2), revealed a further distinctive trend in the real as opposed to the fake tweet content.

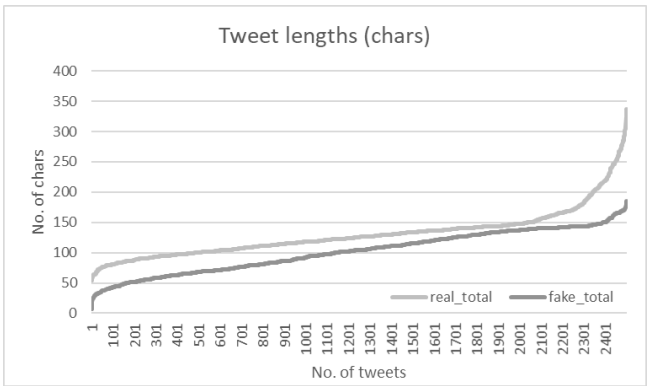


Figure 2. Comparison of Tweet lengths (Chars)

Figure 2 indicates that, as with length measured in number of words, length as measured by number of characters showed a distinctive trend for the real tweet content above the fake tweet content. As before, this may reasonably ease the task of differentiating real from fake tweets.

This post-classification analysis revealed one further notable insight on the character-lengths of real tweets. As shown in Figure 2 (above), some tweets with real content exceeded the 280-character maximum size permitted on Twitter. In total, twelve tweets in the real category of tweets were found to exceed 280 characters. Upon further investigation, this was found to be due to the presence of appended URLs in these tweets that had not been removed during the data cleaning stage. While this accounted for only 0.48% of the total real tweets, the excessive length of these tweets single them out as different from every example of fake tweet.

Additional insight on data complexion was derived from comparison of average and median values for length by words and length by characters (Table VI). This showed little difference in average and median tweet lengths in words and a wider separation in terms of characters.

TABLE VI. AVERAGE AND MEDIAN TWEET LENGTHS

	Real	**Fake**
Average tweet length (words)	15	13
Median tweet length (words)	14	12
Average tweet length (chars)	130	102
Median tweet length (chars)	125	104

A final contrast was made across the real and fake tweet datasets in terms of the use of specific characters. Two factors were considered: the presence of 'special characters' and the number of character types (i.e., unique characters) in the tweets.

The character-level Posit analysis generates several features based upon use of special characters for each data item. In this case, the special characters are full-stop, question mark, exclamation mark, dollar sign and asterisk, i.e., five possible special characters.

The contrast between real and fake tweet content in terms of how many different special characters appear in each tweet is illustrated in Figure 3 (below). This reveals notable

differences between the two types of tweet. Fake tweets avoid all special characters more commonly than real tweets. While many of both types deploy one special character, many more of the fake tweets deploy two special characters. There is less difference between the varieties of tweet at the three special character level while no tweets combine use four or five of these special characters.

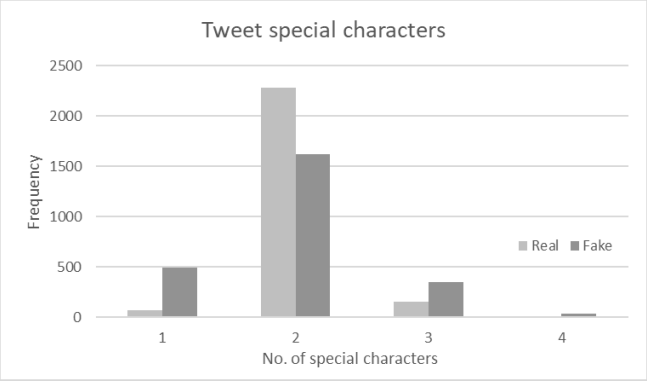


Figure 3. Comparison of Special Character Usage

Our comparison of the number of unique characters present across the tweet datasets, surveyed the presence of the alphanumeric set of characters (not case-sensitive) and the special characters noted above. This contrast is illustrated in Figure 4 (below) and further indicates a subtle difference between real and fake tweet content.

As noted earlier, the classification performance using Posit as a basis for feature generation gave a best performance match to the manual classification of 90.12%. While balanced in sample size, classification performance on this relatively small data subset of 2,500 real and 2,500 fake tweets, may have been influenced positively by the data characteristics described above. As a step toward eliminating such a potential anomaly, we deployed a much larger dataset when classifying with Tensorflow.

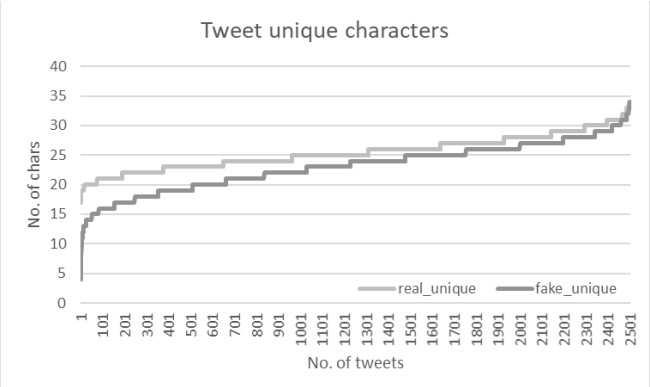


Figure 4. Comparison of Unique Character Usage

C. TensorFlow Results

Recall that in this project, TensorFlow (developed by Google Brain) was used for processing the data with a Deep Neural Network (DNN) [56], [63]. Posit analyzed a randomized sample of 5,000 tweets, that is, the 2,500 "fake news" tweets, and the 2,500 "real news" tweets. A much larger dataset, consisting of 2,709,204 million tweets and Facebook posts, was fed into TensorFlow upon commencement, in order to conduct DNN learning. Then, the DNN results either updated an existing model, or created a new model. TensorFlow next compared the same data against the constructed DNN model, and utilized that model to predict the category for each data entry. In the early stages of experimentation, using default TensorFlow parameters for number of partitions, epochs, layers, learning rate, and regularization, the accuracy results yielded an average of around 60%. Many parameter values (for each parameter: number of partitions, epochs, layers, learning rate, and regularization) were then tested to identify an optimal set of parameter values. This resulted in an increase in accuracy to 89.5%, a substantial improvement from the earlier results. These parameters are described below, with the post-training optimal values shown in Table VII.

To be able to run large numbers of experiments, we wrapped all code into a standalone function, so that large numbers of various scenarios could be designed, set up, and tested continuously. These batch jobs allowed us to evaluate different combinations of parameters. The parameters of each run, and the corresponding results, are also shown below. Tests were run using 10 partitions, with training on the first 5 partitions, and testing on the last 5 partitions.

D. SentiStrength Results

SentiStrength assigns positive or negative values to lexical units in the text [61], [64]. Recall that this value is a measure that provides a quantitative understanding of the content of information—specifically, the extent to which positive and negative sentiment is present. The program automatically extracts the emotions or attitude of a text and assigns a value that ranges from "negative"' to "neutral" to "positive." For SentiStrength analysis, we consolidated four smaller, 2,500 item datasets into one larger, 10,000 item dataset. This larger, 10,000 item dataset consisted of the set of 2,500 randomly sampled "fake news" Twitter messages, the set of 2,500 randomly sampled "real news" Twitter messages, the set of 2,500 "fake news" posts from Facebook, and the set of 2,500 comparator "real news" posts.

For initial SentiStrength analysis, the "general sentiment" was calculated (i.e., without keywords), but all scores were negative, without any apparent distinguishing trends—between "fake news" and "real news," or between Twitter items and Facebook items. We then proceeded to use keywords, by calculating the top 100 nouns out of the 10,000 posts, and running sentiment analysis again, this time with respect to the 100 identified nouns. This produced a 10,000 x 100 matrix (4 x 2,500 = 10,000 rows, one for each post, and 100 columns for each noun, or keyword). On this matrix, we ran various algorithms using WEKA [65] and TensorFlow [56], in an effort to differentiate between the four classes, that is, "fake news" Twitter messages, "real news" Twitter messages, "fake news" posts from Facebook, and the comparator "real news" posts. This too proved to be futile, as

there were too many missing values for the decision trees to handle properly, and given that the reported predictive accuracy was not much better than a random guess.

TABLE VII. TENSORFLOW PERFORMANCE RESULTS

Layers	Learn Rate	Partition	Size	Time	Accuracy
[500, 500]	0.003	0	674941	44.683	0.873
[500, 500]	0.003	1	675072	48.102	0.873
[500, 500]	0.003	2	674613	45.654	0.873
[500, 500]	0.003	3	675109	45.638	0.873
[500, 500]	0.003	4	9479	2.562	0.871
[700, 700]	0.003	0	674941	217.444	0.873
[700, 700]	0.003	1	675072	57.929	0.874
[700, 700]	0.003	2	674613	59.508	0.873
[700, 700]	0.003	3	675109	58.923	0.873
[700, 700]	0.003	4	9479	3.020	0.872
[500, 500]	0.03	0	674941	128.865	0.882
[500, 500]	0.03	1	675072	59.551	0.882
[500, 500]	0.03	2	674613	60.684	0.881
[500, 500]	0.03	3	675109	61.396	0.882
[500, 500]	0.03	4	9479	3.205	0.895

Finally, as were primarily interested in distinguishing "fake news" from "real news," we collapsed the four datasets into two classes, "real news" and "fake news," each consisting of 5,000 items. The results of this final sentiment analysis are shown in Table VIIII (below). While the BayesNet and Naïve Bayes indicated 56.85% and 58.06% of correctly classified instances respectively, these would be considered barely better than random guesses, at 50.00%. However, the MultiLayer Perceptron, a deep neural net algorithm, similar to that found in TensorFlow, in that it employs neurons, weights, and hidden layers [56], [63], [69], [71], yielded a classification accuracy of 74.26%. This would be considered "acceptable," or at least more acceptable than barely better than random guesses, but not up to the standards that we are presently seeking.

TABLE VIII. DETAILED ACCURACY BY ALGORITHM FOR BEST SENTISTRENGTH RESULT

Algorithm	Accuracy
Random Guess	50.00%
Decision trees	50.33%
BayesNet	56.84%
Naïve Bayes	58.06%
Multilayer Perceptron	74.26%

E. LibShortText Analysis

For LibShortText analysis, we again consolidated four smaller, 2,500 item datasets into one larger, 10,000 item dataset, identical to the one used for the SentiStrength analysis (see above). This 10,000 item dataset was split into two randomly sorted 5,000 item datasets, one for training purposes, and the other for testing purposes. For our research project, we built a model using the default settings that came with the LibShortText software [68]. On the first attempt, our classification accuracy was 80.56%, substantially better than the accuracy yielded by the SentiStrength analysis. Our second attempt resulted in a classification accuracy of 90.2%, comparable to the classification accuracy yielded by Posit, at 90.12%, albeit using a larger and more diverse dataset than the one input to Posit.

V. DISCUSSION

We were disappointed with the SentiStrength analysis, given that when we combined SentiStrength with the WEKA standard J48 decision-tree classification method in an earlier study of online extremist content, we were able to correctly classify 80.51% of the webpages [16]. In fact, with our earlier extremism study, the binary anti-extremist and pro-extremist categories had even higher degrees of correctly identified pages, with 92.7% of the pro-extremist cases and 88% of the anti-extremist cases correctly identified. This indicated to us that the decision tree worked well when it came to classifying extremist content [16]. In this present study, the MultiLayer Perceptron (a deep neural net algorithm) yielded a classification accuracy of 74.26%, which is comparable to the results of other studies that have employed sentiment analysis on tweets [59], [83]. We are hoping that further machine training, perhaps enhanced by an expanded list of keywords provided by the ongoing qualitative analysis, will improve upon these SentiStrength results.

TensorFlow epitomizes machine-learning and artificial intelligence, in that it gradually teaches itself, once provided with sufficient data and the requisite training/learning epochs. It is anticipated that the predictive accuracy of the TensorFlow component will ultimately exceed 90% once it is fully trained and fully operational. In a current trial experiment, we demonstrated that the predictive accuracy of TensorFlow does indeed improve with the amount of inputted data. For the first round of analysis, we randomly selected 10,000 Facebook items from another "real news" dataset and 10,000 items from another "fake news" dataset that we had recently generated using the Dark Crawler, next merging and shuffling the two files to create one file containing 10,000 Facebook items. In this case, the predictive accuracy of TensorFlow was only 48.65% when analyzing the content alone, and 50.4% when analyzing the content along with tagged text generated by Posit. On the other hand. TensorFlow's predictive accuracy increased to 79.84% and 79.94% (with the Posit features) when we used 90,000 Facebook items from our "real news" dataset and 10,000 items from our "fake news" dataset to create a larger file containing 100,000 Facebook items.

That said, TensorFlow requires big data and significant processing times. Thus, while TensorFlow will be instrumental in analysing the massive amount of data to be harvested, it will likely not be capable of providing the type of near-real-time alerts on hostile information activities required for our anticipated "critical content toolkit." Rather, we expect that it will provide ongoing, deep-level analysis of all of the data as it is collected, and assist in the building of new models in response to any changes in the strategies and tactics of hostile foreign actors. As a consequence, we anticipate that we will be turning to other (companion) models to enhance the prospects for near-real-time alerts.

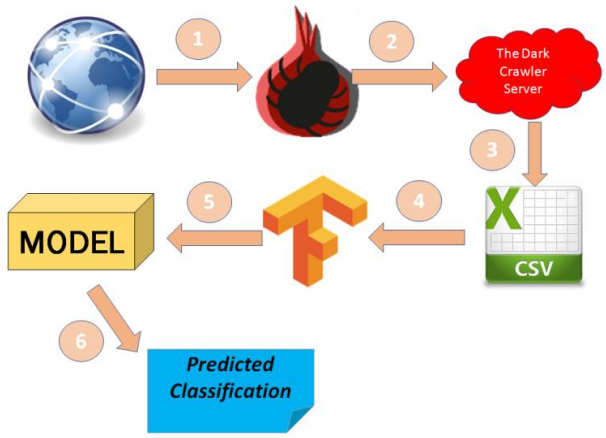


Figure 5. The TensorFlow Model

The TensorFlow model on which we are presently working (see Figure 5, above) commences with The Dark Crawler searching the Internet and downloading all relevant content onto The Dark Crawler server. The data from the stored content is then converted into an Excel file containing all of the pertinent information for each individual data item (e.g., the time and date of the message, text, or post; the hashtag, Facebook page or publication source; the forum and subforum, if taken from a forum; the Internet address, if available; the title of the text or message, if any; the body of the text or message; the number of likes, re-posts or re-tweets; etc.). This data is input to TensorFlow for deep neural network analysis, leading to the generation of a model for measuring the presence of hostile information activities on the Web—a tool which will then predict/classify social media messaging and other sources of online news as "fake" or "real."

Given the limited number of words and word varieties in most tweets, the performance of the Posit analysis using the default 27 word-level features proved to be better than expected, with 86.82% correctly classified instances using Random Forest. The addition of character-level information enhanced this performance to a creditable 90.12% correctly classified instances, again using Random Forest. This result was somewhat surprising, given that alphanumeric details seem far removed from tweet content-level [1].

The Posit toolkit is limited by the speed at which it can read and analyze large volumes of text. Posit is not as slow as TensorFlow, and when combined with WEKA, it has an initial classification accuracy that exceeds that of TensorFlow and in some cases matches that of LibShortText. Nevertheless, while Posit does not excel in reading and analyzing large text corpora as quickly as LibShortText, or in analyzing the vast amounts of data that can be input into TensorFlow for machine learning purposes, it does bring an entirely different dimension to the model that we are building, in that the Posit toolkit generates frequency data and Part-of-Speech (POS) tagging, with data output including values for total words (tokens), total unique words (types), type/token ratio, number of sentences, average sentence length, number of characters, average word length, noun types, verb types, adjective types, adverb types, preposition types, personal pronoun types, determiner types, possessive pronoun types, interjection types, particle types, nouns, verbs, prepositions, personal pronouns, determiners, adverbs, adjectives, possessive pronouns, interjections particles. As Posit recognizes and records individual words and characters, it can aid significantly in the adaptation of the overall model to the changing strategies and tactics of hostile foreign actors, and at the same time, glean unique keywords or key phrases from incoming data so that the activities of hostile foreign actors can be identified quickly and targeted more precisely.

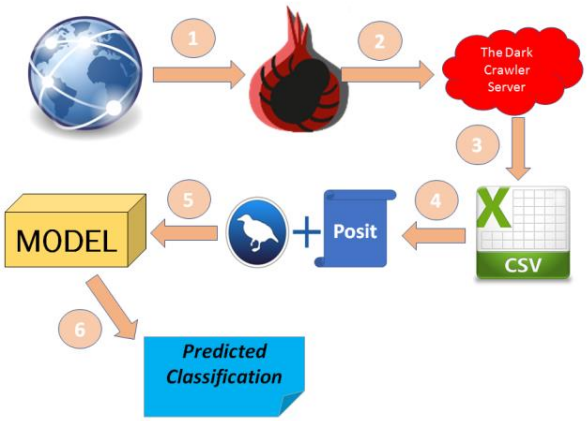


Figure 6. The Posit/WEKA Model

The Posit model that we envision (see Figure 6, above) differs from the TensorFlow Model, in that once the data is harvested, organized, and ready for input, it first goes into Posit for analysis, and then into WEKA for secondary assessment of classification accuracy. Posit (in combination with WEKA) has at times generated classification accuracy in the 98-99% range when it comes to processing various of the recently generated data sets that we have on hand.

The LibShortText results were very encouraging, with a creditable classification accuracy of 90.2%, comparable to the 90.12% classification accuracy yielded by Posit. Recently, in conjunction with our work with LibShortText, we downloaded and configured LibLinear, a companion open source software package, again developed by the same Machine Learning Group at National Taiwan University that developed LibShortText [84]. LibShortText is a text analysis program, while LibLinear is a classification program. LibLinear predicts the accuracy of the classification performed by LibShortText, much like WEKA predicts the accuracy of the classification performed by Posit. Another advantage to LibLinear is that is supports incremental and decremental learning, or to express it differently, the addition and removal of data in order to improve optimization and decrease run time. LibShortText, on the other hand, does not readily support updating of the model.

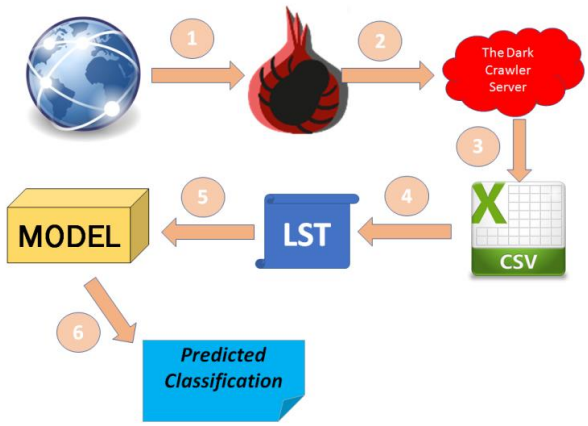


Figure 7. The LibShortText/LibLinear Model

Generally speaking, LibShortText and LibLinear have been outperforming TensorFlow and Posit in a number of our current trial experiments. To illustrate, when analyzing 1,000 randomly selected data items taken from our more recently generated "real news" datasets, contrasted with 1,000 randomly selected data items taken from our more recently generated "fake news" datasets, we found that LibShortText and LibLinear exhibited classification accuracies of 93% and 92% respectively, as opposed to Posit and WEKA at 72.7%, TensorFlow (using Posit-generated .arff content at 54.5%, TensorFlow (using content only) at 52.5%, and TensorFlow (using tagged text) at 48%. We would consider these TensorFlow numbers to be no better than tossing a coin, but these results were not entirely unexpected, as TensorFlow thrives on large data, and this experiment was conducted using only 2,000 discrete data items.

This LibShortText/LibLinear Model (see Figure 7, above) is essentially the same as the TensorFlow Model set out in Figure 5 (above), in that it commences with The Dark Crawler searching the Internet and downloading all relevant content onto The Dark Crawler server. The data from the stored content is then converted into an Excel file containing all of the pertinent information for each individual data item. This data is input to either LibShortText or LibLinear for the generation of a model for measuring the presence of hostile information activities on the Web—a tool which will then predict/classify social media messaging and other sources of online news as "fake" or "real," much more quickly than TensorFlow or Posit.

In this model, LibShortText and LibLinear can be used almost interchangeably, in most cases without unduly affecting the processing times or predictive accuracy. We did, however, encounter limitations with LibShortText on the training and testing file sizes when using only 4GB of RAM. We received "memory exhausted" notifications, and instructions to "restart python." After upgrading to 32GB of RAM, this problem was resolved. That said, the required RAM size is an issue to be borne in mind as we design the final model.

VI. CONCLUSION

Through the research process outlined above, we have: 1) developed typologies of past and present hostile activities in Cloud-based social media platforms; 2) identified indicators of change in public opinion (as they relate to hostile disinformation activities); 3) identified the social media techniques of hostile actors (and how best to respond to them); and 4) undertaken cross-cultural analyses, to determine how hostile actors seek to fuel tensions and undermine social cohesion by exploiting cultural sensitivities.

Our current research will ultimately generate an algorithm that can automatically detect hostile disinformation content. In the longer term, we will use the knowledge generated by this research project to further expand and integrate the capabilities of the Posit toolkit and the Dark Crawler, in order to facilitate near-real-time monitoring of disinformation activities in the Cloud. Further, we plan to add a feature that will permit us to capture disinformation messages prior to their removal by social media organizations attempting to delete those accounts, and/or their removal by actors seeking to conceal their online identities. Ideally, this integrated, "critical content toolkit" will be able to recalibrate itself when confronted with ever-changing forms of disinformation.

During the research process, we also downloaded 2,500 "fake news" Facebook messages that had been posted by the IRA on Facebook pages known variously as Blacktivist, Patriototus, LGBT United, Secured.Borders, and United Muslims of America. (These 2,500 Facebook messages were included in our TensorFlow, SentiStrength and LibShortText analysis). All 2,500 of these messages have been subjected to a preliminary review in the qualitative research tool, NVivo, and also, to preliminary review in Posit [9]. Early insights from this companion study revealed that many of the allegedly "fake news" items were founded to one degree or another in contemporaneous "real news" events.

Following the initial rounds of data collection described earlier in this paper, we broadened and enriched our selection of data sources, focussing primarily on Facebook, Twitter, and other web-based news sources. A "fake news" list of Facebook pages was generated by searching for Facebook pages that belonged to websites described by MediaBiasFactCheck.com as coming from "questionable sources." MediaBiasFactCheck was founded and is edited by Dr. David Van Zandt—a professor, lawyer, and current president of The New School—along with his team of volunteers. In all, we harvested 96,219 Facebook "fake news" items, posted between January 2014 and September 2019. This was recently supplemented by a set of 3,736 Canadian Facebook "fake news" items, posted from May 2014 up to the present.

Data for the expanded Twitter dataset, specifically assembled by the research team for this ongoing project, were also extracted the same way as the set of "fake" Facebook posts, that is, by using the list of 530 "questionable sources" published by MediaBiasFactCheck.com. From this, 181 Twitter accounts were identified for data collection,

accounting for 43,193 data items posted between March 2009 up to the present. Only Twitter accounts that contained a link to the websites identified as suspect by MediaBiasFactCheck.com were included in this sample.

Our third category of "fake news" was recently derived from Web sites presenting themselves as legitimate sources of real news but considered "fake." News articles were collected from four publicly available datasets: (1) *ISOT Fake News*, (2) *Getting Real About Fake News*, (3) *Fake News Corpus*, and (4) *FA-KES: A Fake News Dataset around the Syrian War*. *ISOT Fake News* was created by the Information Security and Object Technology (ISOT) research lab at the University of Victoria [85]. The dataset contains both fake and real news. The former was obtained from websites considered unreliable by Politifact, a website dedicated to fact-checking U.S. news. Real news was obtained from the website Reuters.com. In total, there were 21,417 real news and 23,481 fake news items. *Getting Real About Fake News* was created in 2016 by Megan Risdal, a Product Lead at Kaggle (an online data science community). This dataset contains 12,999 news articles from 244 sources obtained from the BS Detector Chrome extension. The articles are labeled according to their credibility as fake, conspiracy, hate, bias, satire, junk science, and "bullshit."

Fake News Corpus is an open source dataset from 2018 that contains 9,408,908 news articles, created by GitHub user "several27." News articles were obtained from a list of 745 domains from www.opensources.com, as well as the *New York Times* and webhose English news articles. For the current project, after cleansing the dataset by removing unlabelled items, we have retained 779,882 fake news items and 1,783,529 credible news items. Finally, the *FA-KES* dataset, created at the American University of Beirut with the intention of helping train machine learning models, contains 805 news articles about the conflict in Syria, of which 46 are labelled as "fake," with the remaining 378 labelled as "real" [86].

Comparator "real news" Facebook and Twitter data sets have been collected from official news sources representing the top 24 Canadian newspapers in accordance with their known circulation in 2016. We also included *Huffington Post Canada* and two TV News sources with large online followings—*CBC News* and *CTV News*. Apart from the *CBC*, *CTV* and the Canadian edition of the *Huffington Post*, we obtained data from 24 sources, for example, *The Globe and Mail*, *The National Post*, *The Toronto Star*, *Le Journal de Montreal* (French), *Le Journal de Quebec* (French), *Le Soleil* (French), *The Vancouver Sun*, *The Toronto Sun*, *The Calgary Herald*, *The Winnipeg Free Press*, *The Ottawa Citizen*, and *The Montreal Gazette*, to mention a few of the sources. In total, we recently collected 31,557 "real news" Facebook data items from these "trustworthy" news sources, dating from July 2018 to the present. We also collected 253,936 "real news" Twitter data items from these "trustworthy" news sources, dating from December 2013 through September 2019.

This vast databank of recently acquired "real news" and "fake news" (and everything in between real and fake) has been assembled for use in conjunction with our ongoing qualitative analysis, as well as to provide a basis for our ongoing quantitative analysis and machine-learning-based classification. In fact, data drawn from these new datasets were used in our recent comparison tests involving Posit, TensorFlow, LibShortText and LibLinear, as outlined above in our Discussion section. The data collection and data analysis processes are in progress and robust. We anticipate developing a "proof-of-concept" model of our "critical content toolkit" in the near future.

ACKNOWLEDGMENTS

This research project would not have been possible without funding from the Cyber Security Cooperation Program, operated by the National Cyber Security Directorate of Public Safety Canada. We would also like to thank our research assistants, Soobin Rim (TensorFlow) and Aynsley Pescitelli (NVivo).

REFERENCES

[1] B. Cartwright, G. R. S. Weir and R. Frank, "Fighting Disinformation Warfare with Artificial Intelligence: Identifying and Combatting Disinformation Attacks in Cloud-based Social Media Platforms," *Tenth International Conference on Cloud Computing, GRIDs, and Virtualization*, pp. 73-77, May 2019. URL: http://thinkmind.org/index.php?view=article&articleid=cloud_computing_2019_5_30_28006 [Last accessed: 2019.07.28]

[2] D. Ebner and C. Freeze, "AggregateIQ, Canadian data firm at centre of global controversy, was hired by clients big and small," *Globe and Mail*, April, 2018. URL: www.theglobeandmail.com/canada/article-aggregateiq-canadian-data-firm-at-centre-of-global-controvery-was [Last accessed: 2019.04.8]

[3] R. Rathi, "Effect of Cambridge Analytica's Facebook ads on the 2016 US Presidential Election," *Towards Data Science*, 2019. URL: https://towardsdatascience.com/effect-of-cambridge-analyticas-facebook-ads-on-the-2016-us-presidential-election-dacb5462155d [Last accessed: 2019:07.20]

[4] J. Russell, "UK watchdog hands Facebook maximum £500K fine over Cambridge Analytica data breach," *TechCrunch*, 2018. URL: https://techcrunch.com/2018/10/25/uk-watchdog-hands-facebook-500k-fine/ [Last accessed: 2019.07.18]

[5] M. H. McGill and N. Scola, "FTC approves $5B Facebook settlement that Democrats label 'chump change,'" *Politco*, July 12, 2019 URL: https://www.politico.com/story/2019/07/12/facebook-ftc-fine-5-billion-718953 [Last accessed: 2019.07.18]

[6] I. Lapowsky, "House Probes Cambridge Analytica on Russia and Wikileaks," *Wired*, 2019. URL: https://www.wired.com/story/congress-democrats-trump-inquiry-cambridge-analytica/ [Last accessed: 2019.07.20]

[7] Office of the Director of National Intelligence, "Assessing Russian Activities and Intentions in Recent US Elections," 2017. URL: www.dni.gov/files/documents/ICA_2017_01.pdf [Last accessed: 2019.07.28]

[8] R. S. Mueller III, "Report on the Investigation into Russian Interference in the 2016 Presidential Election," pp. 1-448, 2019. URL: www.justsecurity.org/wp-content/uploads/2019/04/Muellller-Report-Redacted-Vol-II-Released-04.18.2019-Word-Searchable.-Reduced-Size.pdf [Last Accessed: 2019.07.28]

[9] B. Cartwright, G. R. S. Weir, L. Nahar, K. Padda and R. Frank, "The Weaponization of Cloud-Based Social Media: Prospects for Legislation and Regulation," *Tenth International Conference on Cloud Computing, GRIDs, and Virtualization*, pp. 7-12, May 2019. URL: http://thinkmind.org/index.php?view=article&articleid=cloud_computing_2019_2_10_28021 [Last accessed: 2019.07.28]

[10] M. T. Bastos and D. Mercea, "The Brexit botnet and user-generated hyperpartisan news," *Social Science Computer Review*, 0894439317734157, 2017. URL: https://journals-sagepub-com.proxy.lib.sfu.ca/doi/pdf/10.1177/0894439317734157 [Last Accessed: 2019.07.28]

[11] M. Field and M. Wright, "Russian trolls sent thousands of pro-Leave messages on day of Brexit referendum, Twitter data reveals: Thousands of Twitter posts attempted to influence the referendum and US elections," *The Telegraph*, 2018. URL: www.telegraph.co.uk/technology/2018/10/17/russian-iranian-twitter-trolls-sent-10-million-tweets-fake-news/ [Last accessed: 2019.04.8]

[12] G. Evolvi, "Hate in a Tweet: Exploring Internet-Based Islamophobic Discourses," *Religions*, *9*(10), pp. 37-51, 2018. URL: https://www.mdpi.com/2077-1444/9/10/307 [Last accessed: 2019.07.21]

[13] A. Badawy, E. Ferrara and Lerman, K., "Analyzing the Digital Traces of Political Manipulation: The 2016 Russian Interference Twitter Campaign," *arXiv*, 2018 URL: https://arxiv.org/abs/1802.04291 [Last accessed: 2019.07.28]

[14] C. Shao, P. M. Hui, L. Wang, X. Jiang, A. Flammini, F. Menczer and and G. L. Ciampaglia, "Anatomy of an online misinformation network," PloS one, 13(4), e0196087, 2018. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0196087 [Last Accessed: 2019.07.28]

[15] W. Y. Wang, "'Liar, Liar Pants on Fire': A New Benchmark Dataset for Fake News Detection," *arXiv preprint arXiv:1705.00648*, 2018. URL: https://arxiv.org/abs/1705.00648 [Last accessed: 2019.07.15]

[16] G. Weir, R. Frank, B. Cartwright and E. Dos Santos, "Positing the problem: enhancing classification of extremist web content through textual analysis," *International Conference on Cybercrime and Computer Forensics (IEEE Xplore)*, June 2016. URL: https://ieeexplore-ieee-org.proxy.lib.sfu.ca/document/7740431[Last accessed: 2019.08.13]

[17] G. Weir, K. Owoeye, A. Oberacker and H. Alshahrani, "Cloud-based textual analysis as a basis for document classification," *International Conference on High Performance Computing & Simulation (HPCS)*, pp. 672-676, July 2018. URL: https://ieeexplore-ieee-org.proxy.lib.sfu.ca/document/8514415 [Last accessed: 2019.08.13]

[18] K. Owoeye and G. R. S. Weir, "Classification of radical Web text using a composite-based method, *IEEE International Conference on Computational Science and Computational Intelligence*, December 2018. URL: https://pure.strath.ac.uk/ws/portalfiles/portal/86519706/Owoeye_Weir_IEEE_2018_Classification_of_radical_web_text_using_a_composite_based.pdf [Last accessed: 2019.08.13]

[19] H. Berghel, "Lies, damn lies, and fake news," *Computer*, 50(2), pp. 80-85, 2017. URL: https://www.computer.org/csdl/magazine/co/2017/02/mco2017020080/13rRUzp02jw [Last accessed: 2019.07.29]

[20] N. W. Jankowski, "Researching fake news: A selective examination of empirical studies," *Javnost-The Public*, 25(1-2), pp. 248-255, 2018. URL: https://www-tandfonline-com.proxy.lib.sfu.ca/doi/full/10.1080/13183222.2018.1418964 [Last accessed: 2019.07.29]

[21] E. C. Tandoc Jr, Z. W. Lim and R. Ling, "Defining 'fake news': A typology of scholarly definitions," *Digital Journalism*, 6(2), pp. 137-153, 2018. URL: https://www.researchgate.net/publication/319383049_Defining_Fake_News_A_typology_of_scholarly_definitions [Last accessed: 2019.07.29]

[22] D. M. Lazer, M. A. Baum, Y. Benkler, A. J. Berinsky, K. M. Greenhill, F. Menczer and M. Schudson, "The science of fake news," *Science*, *359*(6380), pp. 1094-1096, 2018. URL: https://science-sciencemag-rg.proxy.lib.sfu.ca/content/359/6380/1094 [Last Accessed: 2019.07.21]

[23] M. de Cock Buning, L. Ginsbourg and S. Alexandra, *Online Disinformation ahead of the European Parliament elections: toward societal resilience*, European University Institute, School of Transnational Governance, April 2019 URL: https://cadmus.eui.eu/bitstream/handle/1814/62426/STG_PB_2019_03_EN.pdf?sequence=1&isAllowed=y [Last accessed: 2019.07.15]

[24] S. Desai, H. Mooney and J.A. Oehrli, *"Fake News," Lies and Propaganda: How to Sort Fact from Fiction*, 2018. URL: https://guides.lib.umich.edu/fakenews [Last accessed: 2019.07.15]

[25] N. Kshetri and J. Voas, "The Economics of 'Fake News,'" *IEEE Computer Society*, pp. 8-12, (2017). URL: https://ieeexplore-ieee-org.proxy.lib.sfu.ca/stamp/stamp.jsp?tp=&arnumber=8123490&tag=1 [Last accessed: 2019.07.15]

[26] H. Allcott and M. Gentzkow, "Social Media and Fake News in the 2016 Election," *Journal of Economic Perspectives*, 31(2), pp. 211-236, 2017. URL: https://web.stanford.edu/~gentzkow/research/fakenews.pdf [Last Accessed: 2019.07.28]

[27] W. L. Bennett and S. Livingston, "The disinformation order: Disruptive communication and the decline of democratic institutions," *European Journal of Communication*, 33(2), pp. 122-139, 2018. URL: https://journals-sagepub-com.proxy.lib.sfu.ca/doi/pdf/10.1177/0267323118760317 [Last accessed: 2019.07.28]

[28] *United States v. Internet Research Agency LLC*, Case 1:18-cr-00032-DLF, The United States District Court for the District Of Columbia, February 26, 2018. URL: www.justice.gov/file/1035477/download [Last accessed: 2019.04.8]

[29] J. J. Green, "Tale of a Troll: Inside the 'Internet Research Agency' in Russia," *WTOP*, 2018. URL: https://wtop.com/j-j-green-national/2018/09/tale-of-a-troll-inside-the-internet-research-agency-in-russia/ [Last accessed: 2019.07.15]

[30] L. Reston, "How Russia Weaponizes Fake News: The Kremlin's influence campaign goes far beyond Trump's victory. Their latest unsuspecting targets: American conservatives," *The New Republic*, 2017. URL: https://newrepublic.com/article/142344/russia-weaponized-fake-news-sow-chaos [Last accessed: 2019.07.20]

[31] K. Wagner, "Facebook and Twitter worked just as advertised for Russia's troll army: Social platforms are an effective tool for marketers — and nation states that want to disrupt an election," *Recode Daily*, 2018. URL: https://www.vox.com/2018/2/17/17023292/facebook-twitter-russia-donald-trump-us-election-explained [Last accessed: 2019.07.20]

[32] A. Marwick and R. Lewis, *Media Manipulation and Disinformation Online*, New York: Data & Society Research Institute, pp. 1-106, 2017. URL: https://datasociety.net/output/media-manipulation-and-disinfo-online/ [Last accessed: 2019.07.29]

[33] K. Shu, A. Silva, S. H. Wang, J. Tang and H. Liu, "Fake News Detection on Social Media: A Data Mining Perspective," pp. 1-15, 2017. URL: https://arxiv.org/abs/1708.01967 [Last accessed: 2019.07.29]

[34] S. Zanettou, T. Caulfied, E. de Cristofaro, M. Sirivianos, G. Stringhini and J. Blackburn, "Disinformation Warfare: Understanding State-Sponsored Trolls on Twitter and Their Influence on the Web," pp. 1-11, 2019. URL: https://arxiv.org/pdf/1801.09288.pdf [Last accessed: 2019.07.29]

[35] M. Papenfuss, "1,000 Paid Russian Trolls Spread Fake News On Hillary Clinton, Senate Intelligence Heads Told," *Huffington Post*, March 2017. URL: https://www.huffingtonpost.ca/entry/russian-trolls-fake-news_n_58dde6bae4b08194e3b8d5c4 [Last accessed: 2019.07.29]

[36] The Computational Propaganda Project, "Resource for Understanding Political Bots," 2016. URL: https://comprop.oii.ox.ac.uk/research/public-scholarship/resource-for-understanding-political-bots/ [Last accessed: 2019.07.29]

[37] P. N. Howard, S, Woolley and R. Calo, "Algorithms, bots, and political communication in the US 2016 election: The challenge of automated political communication for election law and administration," *Journal of Information Technology & Politics*, *15*(2), 81-93, 2018. URL: https://www-tandfonline-

com.proxy.lib.sfu.ca/doi/full/10.1080/19331681.2018.1448735 [Last accessed: 2019.07.18]

[38] G. Resnick, "How Pro-Trump Twitter Bots Spread Fake News," *The Daily Beast*, July 2017. URL: https://www.thedailybeast.com/how-pro-trump-twitter-bots-spread-fake-news [Last accessed: 2019.07.29]

[39] G. Krieg, "It's official: Clinton swamps Trump in popular vote," *CNN Politics Data*, December 2016, URL: https://www.cnn.com/2016/12/21/politics/donald-trump-hillary-clinton-popular-vote-final-count/index.html [Last accessed: 2019.07.29]

[40] L. Stark, "Alorithmic psychometrics and the scalable subject," *Social Studies of Science*, 48(2), pp. 204-231, 2018 URL: https://journals-sagepub-com.proxy.lib.sfu.ca/doi/pdf/10.1177/0306312718772094 [Last accessed: 2019.08.03]

[41] F. Morstatter, L. Wu, T. H. Nazer, K. N. Carley and H. Liu, "A new approach to bot detection: Striking the balance between precision and recall," *IEEE/ACM Conference on Advances in Social Networks Analysis and Mining*, pp. 553-540, August 2016. URL: https://ieeexplore-ieee-org.proxy.lib.sfu.ca/document/7752287 [Last accessed: 2019.08.03]

[42] E. Shearer and K. E. Matsa, "News Use Across Social Media Platforms 2018: Most Americans continue to get news on social media, even though many have concerns about its accuracy," Pew Research Center, 2018. URL: www.journalism.org/2018/09/10/news-use-across-social-media-platforms-2018/ [Last accessed: 2019.07.30]

[43] J. Gottfried and E. Shearer, "News Use Across Social Media Platforms 2016," Pew Research Center, URL: https://www.journalism.org/2016/05/26/news-use-across-social-media-platforms-2016/ [Last accessed: 2019.07.30]

[44] N. Kshetri and J. Voas, "The Economics of 'Fake News,'" *IEEE Computer Society*, pp. 8-12, (2017). URL: https://ieeexplore-ieee-org.proxy.lib.sfu.ca/stamp/stamp.jsp?tp=&arnumber=8123490&tag=1 [Last accessed: 2019.07.15]

[45] S. Wineburg, S. McGrew, J. Breakstone and T. Ortega, "Evaluating Information: The Cornerstone of Civic Online Reasoning," Stanford Digital Repository, 2016. URL: https://stacks.stanford.edu/file/druid:fv751yt5934/SHEG%20Evaluating%20Information%20Online.pdf [Last accessed: 2019.07.15]

[46] European Commission, *A Europe that protects: EU reports on progress in fighting disinformation ahead of European Council*, June 2019. URL: https://ec.europa.eu/commission/commissioners/2014-2019/ansip/announcements/europe-protects-eu-reports-progress-fighting-disinformation-ahead-european-council_en [Last accessed: 2019.06.15]

[47] A. Al-Rawi and Y. Jiwani, "Trolls Stoke Fear: Russian disruption a concern in Fall vote," *Vancouver Sun*, p. G2, August 2016.

[48] C. Falk, "Detecting Twitter Trolls Using Natural Language Processing Techniques Trained on Message Bodies," July 2018. URL: http://www.infinite-machines.com/detecting-twitter-trolls.pdf [Last accessed: [2019.07.15]

[49] I. Smoleňová, "The pro-Russian disinformation campaign in the Czech Republic and Slovakia," *Prague: Prague Security Studies Institute*, 2015. URL: http://www.pssi.cz/download/docs/253_is-pro-russian-campaign.pdf [Last accessed: 2019.07.21]

[50] I. Khaldarova and M. Pantti, "Fake news: The narrative battle over the Ukrainian conflict, "*Journalism Practice*, 10(7), pp. 891-901, 2016. URL: https://www-tandfonline-com.proxy.lib.sfu.ca/doi/full/10.1080/17512786.2016.1163237 [Last accessed: 2019.07.21]

[51] U. A. Mejias and N. E. Vokuev, "Disinformation and the media: the case of Russia and Ukraine. Media," *Culture & Society*, 39(7), pp. 1027-1042, 2017. URL: https://journals-sagepub-com.proxy.lib.sfu.ca/doi/full/10.1177/0163443716686672 [Last accessed: 2019.07.21]

[52] S. Bradshaw and P. N. Howard, "The Global Disinformation Order: 2019 Global Inventory of Organized Social Media Manipulation." URL: https://comprop.oii.ox.ac.uk/wp-content/uploads/sites/93/2019/09/CyberTroop-Report19.pdf [Last Accessed: 2019.11.16]

[53] D. Alba and A. Satariano, "At Least 70 Countries Have Had Disinformation Campaigns, Study Finds," *New York Times*, September 2019. URL: https://www.nytimes.com/2019/09/26/technology/government-disinformation-cyber-troops.html [Last Accessed: 2019.11.16]

[54] D. L. Linvill and P. L. Warren, "Troll factories: The Internet Research Agency and state-sponsored agenda-building," Resource Centre on Media, 2018. URL: https://www.google.com/search?q=Troll+factories%3A+The+Internet+Research+Agency+and+state-sponsored+agenda-building&oq=Troll+factories%3A+The+Internet+Research+Agency+and+state-sponsored+agenda-building&aqs=chrome..69i57j69i60l3.354j0j7&sourceid=chrome&ie=UTF-8 [Last accessed: 2019.07.21]

[55] H. F. Yu, C. H. Ho, Y. C. Juan and C. J. Lin, "LibShortText: A Library for Short-text Classification and Analysis", Department of Computer Science, National Taiwan University, 2013. URL: https://www.csie.ntu.edu.tw/~cjlin/papers/libshorttext.pdf [Last accessed: 2019.08.4]

[56] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis,J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker,V. Vasudevan, P. Warden, M. Wicke, Y. Yu and X. Zheng, "TensorFlow: A system for large-scale machine learning," *12th USENIX Symposium on Operating Systems Design and Implementation*, pp. 265-283, November 2016. URL: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf [Last accessed: 2019.08.4]

[57] G. R. S. Weir, "The posit text profiling toolset," *12th Conference of Pan-Pacific Association of Applied Linguisitics*, pp. 106-109, 2007. URL: https://www.researchgate.net/publication/228740404_The_Posit_Text_Profiling_Toolset [Last accessed: 2019.08.4]

[58] G. R. S. Weir, "Corpus profiling with the Posit tools," *Proceedings of the 5th Corpus Linguistics Conference*, July 2009. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.159.9606&rep=rep1&type=pdf [Last accessed: 2019.08.4]

[59] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner and Y. Choi, "Defending Against Neural Fake News," *arXiv preprint arXiv:1905.12616*, 2019. URL: https://arxiv.org/abs/1905.12616 [Last Accessed: 2019.11.17].

[60] Y. Gorodnichenko, T. Pham and O. Talavera, *Social media, sentiment and public opinions: Evidence from# Brexit and# USElection*, No. w24631, National Bureau of Economic Research, 2018. URL: https://www.nber.org/papers/w24631.pdf [Last Accessed: 2019.11.19].

[61] J. Mei and R. Frank, "Sentiment crawling: Extremist content collection through a sentiment analysis guided webcrawler," 2015 *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 1024-1027, August 2015. URL: esearchgate.net/publication/301444687_Sentiment_Crawling_Extremist_Content_Collection_through_a_Sentiment_Analysis_Guided_Web-Crawler [Last accessed: 2019.08.4]

[62] A. T. Zulkarnine, R. Frank, B. Monk, J. Mitchell and G. Davies, "Surfacing collaborated networks in dark web to find illicit and criminal content," *2016 IEEE Confernce on Intelligence and Security Informatics (ISI)*, pp. 109-114, September 2016. URL: https://ieeexplore-ieee-org.proxy.lib.sfu.ca/document/7745452 [Last accessed: 2019.08.4]

[63] T. C. Kietzmann, P. McClure and N. Kriegeskorte, "Deep neural networks in computational neuroscience," *bioRxiv*, pp. 133504-133527, 2018. URL: https://www.biorxiv.org/content/10.1101/133504v2 [Last accessed: 2019.08.4]

[64] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai and A. Kappas, "Sentiment strength detection in short informal text," *Journal of the*

American Society for Information Science and Technology, 61(12), 2544–2558, 2010. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.278.3863&rep=rep1&type=pdf [Last accessed: 2019.08.4]

[65] M. Hall, E. Frank, H. Geoffrey, B. Pfahringer, P. Reutemann and I. Witten, "The Weka data mining software: an update," *SIGKDD Explorations*, vol. 11, pp. 10-18, 2009. URL: https://www.kdd.org/exploration_files/p2VD:\BTSync\Ricsi\Academic Work\2019\20190508 - Venice Cloud Conference11n1.pdf [Last accessed: 2019.08.4]

[66] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5-32, 2001. URL: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf [Last accessed: 2019.08.4]

[67] A. Liaw and M. Wiener, "Classification and regression by randomForest," *R News*, vol. 2, pp. 18-22, 2002. URL: https://www.r-project.org/doc/Rnews/Rnews_2002-3.pdf [Last accessed: 2019.08.4]

[68] H. F. Yu, C. H. Ho, Y. C. Juan and C. J. Lin, "LibShortText: A Library for Short-text Classification and Analysis", Department of Computer Science, National Taiwan University, 2013. URL: https://www.csie.ntu.edu.tw/~cjlin/papers/libshorttext.pdf [Last accessed: 2019.08.4]

[69] Y. N. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," *Advances in neural information processing systems*, pp. 2933-2941, 2014. URL: https://papers.nips.cc/paper/5486-identifying-and-attacking-the-saddle-point-problem-in-high-dimensional-non-convex-optimization [Last accessed: 2019.08.4]

[70] D. R. Ruck, S. K. Rogers, M. Kabrisky, M. E. Oxley and B. W. Suter, "The Multilayer Perceptron as an Approximation to a Bayes Optimal Discriminant Function," *IEEE Transactions on Neural Networks*, 1(4), pp. 296-298, 1990. URL: https://ieeexplore-ieee-org.proxy.lib.sfu.ca/document/80266 [Last accessed: 2019.08.6]

[71] W. S. Sarle, "Neural Networks and Statistical Models," *Nineteenth Annual SAS Users Group International Conference*, April 1994. URL: https://people.orie.cornell.edu/davidr/or474/nn_sas.pdf [Last accessed: 2019.08.6]

[72] S. Chadha, "Text Analytics on Russian Troll Tweets-Part 1," n.d., URL: https://www.kaggle.com/chadalee/text-analytics-on-russian-troll-tweets-part-1 [Last accessed: 2019.08.7]

[73] *CNN*, "Big Tech braces for first presidential debates, a target of Russian trolls in 2016," June 2019. URL: https://m.cnn.com/en/article/h_8d2922458b2b4c80698925b2be1ab879 [Last accessed: 2019.08.7]

[74] M. Choi, "Keith Ellison reeling after abuse allegations: The No. 2 at the Democratic National Committee is running behind in his bid for Minnesota attorney general," *Politico*, October 2018. URL: https://www.politico.com/story/2018/10/27/keith-ellison-abuse-allegations-minnesota-ag-2018-943086 [Last accessed: 2019.08.10]

[75] E. Nakashima, "U.S. Cyber Command operation disrupted Internet access of Russian troll factory on day of 2018 midterms," *Washington Post*, February 2019. URL: https://www.washingtonpost.com/world/national-security/us-cyber-command-operation-disrupted-internet-access-of-russian-troll-factory-on-day-of-2018-midterms/2019/02/26/1827fc9e-36d6-11e9-af5b-b51b7ff322e9_story.html?noredirect=on [Last accessed: 2019.08.10]

[76] T. Starks, L. Cerulus and M. Scott, "Russia's manipulation of Twitter was far vaster than believed," *Politico*, June 2019. URL: https://www.politico.com/story/2019/06/05/study-russia-cybersecurity-twitter-1353543 [Last accessed: 2019.08.10]

[77] M. Lander, "Trump Says He Discussed the 'Russian Hoax' in a Phone Call With Putin," *The New York Times*, May 2019. URL: https://www.nytimes.com/2019/05/03/us/politics/trump-putin-phone-call.html

[78] *Baltimore Sun*, "Even if Trump is right about collusion, Russia story is big (not fake) news," October 2017. URL: https://www.baltimoresun.com/opinion/editorial/bs-ed-1101-trump-russia-20171031-story.html [Last accessed: 2019.08.10]

[79] R. Dicker, "Secret Service Agent Says She Wouldn't Take A Bullet For Trump: Agency says it is taking quick action, " *Huffington Post*, January 2017. URL: https://www.huffingtonpost.ca/entry/secret-service-agent-says-she-wouldnt-take-a-bullet-for-trump_n_58887d4be4b0441a8f71e671 [Last accessed: 2019.08.10]

[80] M. Y.H. Lee, "'Rapists?' Criminals? Checking Trump's facts," *The Philadelphia Inquirer*, July 2015. URL: https://www.inquirer.com/philly/news/politics/20150709__Rapists__Criminals__Checking_Trump_s_facts.html [Last accessed: 2019.08.11]

[81] J. Hing, "This Is the Beginning of Donald Trump's Muslim Ban: Friday's executive order extended to seven countries—but that list could grow," *The Nation*, January 2017. URL: https://www.thenation.com/article/this-is-the-beginning-of-donald-trumps-muslim-ban/ [Last accessed: 2019.08.11]

[82] L. Gambino, "Trump pans immigration proposal as bringing people from 'shithole countries'," *The Guardian*, January 2018. URL: https://www.theguardian.com/us-news/2018/jan/11/trump-pans-immigration-proposal-as-bringing-people-from-shithole-countries [Last accessed: 2019.08.11]

[83] M. Bouazizi and T. Ohtsuki, "Multi-Class Sentiment Analysis on Twitter: Classification Performance and Challenges," *Big Data and Mining Analytics*, vol. 2, pp. 181-194, 2019. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8681053 [Last Accessed: 2019.11.24]

[84] C. H. Tsai, C. Y. Lin, and C. J. Lin, "Incremental and decremental training for linear classification" *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 343-352, 2014. URL: https://www.csie.ntu.edu.tw/~cjlin/papers/ws/inc-dec.pdf [Last Accessed: 2019.11.24]

[85] H. Ahmed, I. Traore and S. Saad, "Detecting opinion spams and fake news using text classification," *Journal of Security and Privacy*, vol. 1, pp. 1-15. URL: https://www.uvic.ca/engineering/ece/isot/assets/docs/SPY_Detecting%20opinion%20spams%20and%20fake%20news%20using%20text%20classification.pdf [Last Accessed: 2019.11.24]

[86] F. K. A. Salem, R. Al Feel, S. Elbassuoni, M. Jaber and M. Farah, "FA-KES: A Fake News Dataset around the Syrian War," *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 13, pp. 573-582, 2019. URL: https://aaai.org/ojs/index.php/ICWSM/article/view/3254/3122 [Last Accessed: 2019.11.24]

Reaching Grey Havens
Industrial Automotive Security Modeling with SAM

Markus Zoppelt

Department of Computer Science
Nuremberg Institute of Technology
Nuremberg, Bavaria 90489
Email: markus.zoppelt@th-nuernberg.de

Ramin Tavakoli Kolagari

Department of Computer Science
Nuremberg Institute of Technology
Nuremberg, Bavaria 90489
Email: ramin.tavakolikolagari@th-nuernberg.de

***Abstract*—Autonomous vehicles have a greater attack potential than any previous individual mobility vehicle. This is primarily due to the considerable communication demands of the vehicles, which on the one hand emerge for reasons of functionality and safety, and on the other hand for reasons of comfort. Driverless vehicles require communication interfaces to the environment, direct connections (e.g., Vehicle-to-X) and connections to an original equipment manufacturer backend service or a cloud. These communication connections could all be used as backdoors for attacks. Most existing countermeasures against cyber attacks, e.g., the use of message cryptography, concentrate on concrete attacks and do not consider the complexity of the various access options offered by modern vehicles. This is mainly due to a solution-oriented approach to security problems. The model-based technique SAM (Security Abstraction Model) adds to the early phases of (automotive) software architecture development by explicitly documenting attacks and managing them with the appropriate security countermeasures. It additionally establishes the basis for comprehensive security analysis techniques, e.g., already available attack assessment methods. SAM thus contributes to an early, problem-oriented and solution-ignorant understanding combining key stakeholder knowledge. This paper provides a detailed overview of SAM and evaluates this security technology using interviews with industry experts and a grounded theory analysis. The resulting analyses of this evaluation show that SAM puts the security-by-design principle into practice by enabling collaboration between automotive system engineers, system architects and security experts. The application of SAM aims to reduce costs, improve overall quality and gain competitive advantages. Based on our evaluation results, SAM is highly suitable, comprehensible and complete to be used in the industry.**

Keywords–Automotive Security; Automotive Software Engineering; Security Modeling; Model-based Security; Autonomous Driving.

I. INTRODUCTION

Modern vehicles are interconnected computer networks in which many electronic control units (ECUs) communicate with one another and with the environment (Vehicle-to-X communication). In recent years, car manufacturers have been producing vehicles that have an online connection and offer cloud services, such as the mobile app from Tesla, BMW iDrive or Audi Connect. In most cases, the user can actually monitor or control parts of the vehicle via a mobile application or cloud service. These convenience features are intended to attract new customers, but can also be access points for new attacks [1]. Considering the fact that autonomous vehicles will continue rather than reverse the trend towards more communication interfaces for reasons of functionality, safety and comfort, making collective research efforts in the field of vehicle security understandable; after all, human lives are at stake every time these "driving computers" are the target of attacks.

As far as security experts are concerned, it should be noted that car attackers do not target cars the same way as they attack desktop computer systems, because cars use different networks, protocols and architectures [2], [3]. In addition, vehicles often contain obsolete legacy mechanisms with unsecure and unencrypted protocols (e.g., Controller Area Network (CAN)) in their system design, because they were originally not designed in accordance with today's security principles [4], [5]. Secure automotive network architectures were not prioritized in the past due to the general prejudice that cars are secure due to their technical complexity (security by obscurity). Sluggish development processes, lack of standard guidelines and low societal pressure, due to little attack experience in practice, lead to a rather slow transformation of automotive development processes, which systematically implement security by design.

Most existing countermeasures against cyber attacks, e.g., the use of message cryptography for encrypting, authenticating or randomizing vehicle-level network messages, focus on concrete attacks and do not consider the complexity of the access options offered by modern vehicles, as shown by Zoppelt et al. [6]. This is mainly due to a solution-oriented approach to security problems.

The model-based technique SAM (Security Abstraction Model) [7] adds to the early, solution-ignorant phases of (automotive) software architecture development by explicitly documenting attacks and managing them with appropriate security countermeasures. The documentation of the attacks together with their motivation, vulnerability, attackable property and other relevant properties is put in relation to the entire system description by SAM being an annex to the domain-specific architecture description language EAST-ADL [8]. Thus, all available information about the system is linked with potential attack scenarios at an early stage of the automotive system development and cooperation between the key stakeholders is made possible. The problem-oriented documentation allows automotive system developers and security experts to gain a comprehensive picture of the overall attack situation before

developing a solution that otherwise may be too short-sighted.

Zoppelt et al. [7] presented a Security Abstraction Model (SAM) for automotive software systems. In this publication, for the first time, we present a comprehensive description of SAM, put it in context with key security challenges for autonomous driving and evaluate it using grounded-theory evaluation based on interviews.

In this paper, we show:

- A systematic discussion of the current state of the art for security techniques along the V-Model as a common software engineering practice.
- A detailed description of SAM, including all of its metamodel entities.
- An evaluation of the security technique via grounded-theory interviews with industry experts.

The rest of this paper is structured as follows: Section II reviews related work on security architectures for automotive software systems. Section III reviews the state of the art on attacks on modern vehicles and automotive security modeling. Section IV discusses possible attack scenarios and the security challenges in the automotive domain. Section V describes the Security Abstraction Model in detail, including all of its metamodel entities. In Section VI we evaluate SAM and elaborate on the interviews and qualitatively analyse the results via grounded theory. Section VII concludes the paper and gives an outlook on future work.

II. Related Work

The ISO/SAE 21434 "Road Vehicles—Cybersecurity engineering" standard [9], which is currently under development at the time of writing this paper, proposes the introduction of security work packages, security concepts and architectures along the V-Model [10]. It suggests security support before after-sales. Specifically, during product validation and production ramp-up. The standard is being delegated between an consortium of 12 countries. The scope of the standard is to define a framework to include requirements for cybersecurity processes and a common language for communicating and managing cybersecurity risk among stakeholders. Our work considers the early efforts and design principles of the ISO/SAE 21434 and integrates them into the EAST-ADL.

The SAE J3061 "Cybersecurity Guidebook for Cyber-Physical Vehicle Systems" [11], also only available as a work in progress, wants to establish a set of high-level guiding principles for cybersecurity as it relates to cyber-physical vehicle systems, including lifecycle process frameworks and information on common existing tools and methods.

Although not much final information on those standards is currently available, we join and unite many of the proposed methods and principles in our contribution and show its practical applicability in a system model.

PRESERVE was an "EU-funded project running from 2011 to 2015 and contributed to the security and privacy of future vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2X) communication systems. It provides security requirements of vehicle security architectures" [12]. The EVITA project tries to "design, verify and prototype an architecture for automotive on-board networks where security-relevant components are protected against tampering and sensitive data are protected against compromise. It focuses on V2X communications and provides a base for secure deployment of electronic safety applications" [13]. Holm [14] features a Cyber Security Modeling Language (CySeMoL) for enterprise architectures. Juerjens [15] introduces UMLSec, which allows to express security-relevant information within the diagrams in a system specification. Other solutions include INCOSE work on integrating system engineering with system security engineering [16], NIST SP 800-160 [17] and other NIST work on cyber-physical systems [18].

All these solutions have one essential downside: Other than SAM (see Section V), they are stand-alone and are not integrated into an existing system model. SAM is fully integrated into EAST-ADL. In comparison with alternative solutions, a tightly coupled solution with the system model enables a seamless integration of a security model into a system model that is extensively used in the automotive industry. This helps overall acceptance and increases probability for adoption. The tight interplay of SAM with existing system models architectural considerations and practical security considerations together.

III. State of the Art

The automotive core development process is organized according to the traditional software engineering V-Model [10]. Each phase of the V-Model stands for a coherent set of process steps in which a set of artifacts are produced. The phases are logically organized, not temporally. In the system analysis phase, requirements are elicited and documented, in the system design phase, a logical, function-oriented architectural structure is developed that is the basis for both the hardware and software development phases, which results in the implementation of the automotive system. In the following, we discuss the current state of the art for security techniques along the V-Model as a common software engineering practice. For that, we will differentiate between the four major phases of software engineering, namely analysis phase, design phase, implementation phase and software test. Moreover, we will also touch on the topic of security techniques during maintenance, since this is part of the extended V-Model.

A. Security Techniques in Analysis Phase

In the system analysis phase, requirements are elicted and documented. The current state of the practice for security techniques in this phase is to capture requirements from a specification or textual annotations of the system model. Security experts identify threats and vulnerabilities of a system, while software engineers fix bugs and implement security functionality, e.g., cryptographic functions. System architects define the architecture of the system (i.e., the software and hardware topology), taking—among other things—security requirements into consideration.

A case study conducted by Zoppelt et al. [7] has shown that textual annotations cannot fully explain security scenarios in a detailed, yet compact manner. The technical details and the relevance of the threat get lost because the software engineers could not decide for what purpose or security goal textual notes were intended. Security has an inner complexity, especially considering the requirements entailed. Requirements alone are not sufficient enough. System architects and security experts need to be able to mutually annotate the same model. Only

then they can make the necessary adjustments to the system's architecture. A better practice is to define a set of security goals and systematically derive security requirements from a common architecture model and reference architectures.

Threat modeling and risk assessment is a basis for security requirements. Studies like Kadhirvelan's work [19] have shown that new processes, standards, methods and tools are necessary for evaluating the security and safety of software-intensive automotive systems.

Results of the analysis phase (mostly requirements) are then used in the design phase to deduce attack vectors and think of solutions according to derived requirements.

B. Security Techniques in Design Phase

In the system design phase, a logical, function-oriented architectural structure is developed utilizing the requirements (for both software and hardware) from the analysis phase. Many design decisions can be derived from knowing, analyzing or building different attack vectors a potential adversary has to target the system.

Attack vectors are a path or means by which an adversary can gain unauthorized access to a target system [20] or which hurts one or more security goals. Attack vectors can be identified and extracted via attack trees. Attack trees can be used to illustrate complex attack structures. The root of an attack tree describes the main goal or motivation of an attack, e.g., controlling certain functions of the target vehicle. Every sub-attack needs to be completed in order to fulfill the parent attack in the tree. The leafs of the attack tree are atomic actions or conditions. A complete path from one of the leaves to the root of the tree represents a concrete attack vector.

Moreover, security techniques in design phase utilize various systems for attack rating and threat analyses. Those techniques, e.g., the CVSS [21], allow for an early evaluation of essential security measures. The CVSS is an acclaimed industry standard for rating vulnerabilities in computer systems. The CVSS bundles the result of threat analyses via multiple different metrics, e.g., attack complexity, security goal impacts, etc. and produces a numerical score reflecting its severity. Forcing developers to think about attack vectors and vulnerability already in design phase and conducting a CVSS analysis, ensures that errors are detected early. This way, expensive corrections can be prevented from the beginning. Unfortunately, this is not the case. The current state of the practice shows that OEMs forego this analysis at an early stage. We are trying to contribute to this problem with the solution approach presented in this paper. Alongside CVSS—which is not per-se automotive related—more scoring systems exist, e.g., the SecL levels from the SAHARA method [22].

Another significant design choice is how modern vehicles communicate critical- and safety-relevant commands between different types of ECUs. The most popular broadcast network used for communication—even today—is the CAN bus [23]. CAN bus messages are unencrypted and unsigned by default, because back in the 80's—when CAN was designed—automotive security was not perceived a major issue. Remote exploitation of a single ECU item on the CAN bus causes a major security threat because it allows an attacker to send valid (and potentially harmful) messages over the bus to critical parts of the vehicle's ECU network. Modern vehicles have a tremendous amount of remote attack surfaces like wireless protocols, mobile application support and more. Examples of specific remote technologies are the passive anti-theft system (PATS), tire pressure monitoring systems (TPMS), remote keyless entry (RKE), Bluetooth, radio data systems (3G, 4G, LTE, 5G, etc.), Wi-Fi and telematics. Typically, infotainment systems tend to feature Internet access and support for third-party applications. Various attacks [24] have shown that adversaries are able to cause serious threats by compromising a vehicle's ECU (or adding an external device) and sending malicious CAN commands to the devices listening on the bus. Once the adversary has the ability to send arbitrary CAN messages, she is able to control the braking system, engine behaviour, the air vents, (un-)lock the doors, etc. Therefore, there is a strong need to secure the vehicle before the adversary can gain access to the CAN bus. If the adversary has access to the powertrain it is already too late.

Common countermeasure decisions in design phase consider network bus separation. By conceptually and physically separating safety-relevant ECUs from the remaining network system, many attack vectors can be mitigated. In reality, many ECUs are still connected physically, but are being separated through higher protocol abstractions, e.g., Unified Diagnostic Service (UDS) or virtualized applications. Those happen in implementation phase.

C. Security Techniques in Implementation Phase

According to the V-Model, the proposed countermeasures are deployed in the implementation phase.

On the hardware side, implementation may differ by different physical bus systems and wiring. If one or some of these applications or services become vulnerable to hacking attacks over the network, an adversary might be able to control a crucial participant in the physical network of the vehicle: the CAN bus. Another approach in the automotive domain is Automotive Ethernet, though, it is not expected to fully replace the CAN bus. CAN will continue to exist as a low-cost component, for example for connecting low-cost and computationally weak actuators and sensors with their corresponding ECUs or gateways, rather than be used as the main powertrain. As of today, the LIN-bus (Local Interconnect Network) is used for this type (low-cost, low-risk) of connection.

Cost is a limiting factor as well, when it comes to implementing expensive hardware into the vehicle. Automobile manufacturers prefer to spend more money on the salaries of programmers (fixed costs; used for entire fleet) rather than spending a cent more on a hardware part of a vehicle (variable costs; for each vehicle) because of the huge market scale. This means that hardware modules like TPMs (Trusted Platform Modules) are unattractive (cost, weight, space) as a key storing solution for each and every communicating part in the vehicle.

On software side, different protocol variations can be used to implement security measures. Some network protocols like ISO-TP, UDS and OBD2 are on a higher level of abstraction that remedy a few shortcomings of CAN. Figure 1 illustrates selected automotive protocols discussed in this paper in the ISO/OSI reference model.

Although the CAN specification describes CAN as unencrypted by default, a sound solution for encryption and

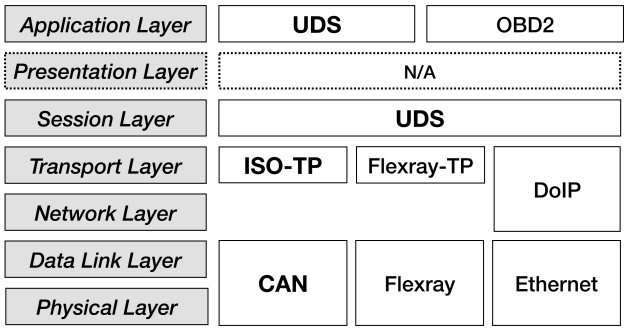


Figure 1. Selected automotive protocols classified in the ISO/OSI reference model

authentication is necessary to ensure a safe and secure distribution of critical new software over this public channel. In the automotive domain, there are not only software updates to consider, but hardware updates as well. If a workshop, for instance, replaces one of the brakes in a vehicle, they might also replace the corresponding ECU. In that scenario, how will the new cryptographic key (for message cryptography) be obtained? Common key distribution techniques like the Diffie-Hellman key exchange [25] are difficult to implement, since many of the smaller network participants are low-cost and computationally weak ECUs. These ECUs often do not feature enough memory or CPU power to perform those cryptographic algorithms and methods. Message cryptography on the CAN bus is not only hard to realize due to the strong network complexity, where key distribution is a difficult problem, but because an adversary in control of an ECU also gets access to the keys stored on that device. For some parts of the vehicle, where stronger threat models are required, e.g., keyless entry systems, emerging technologies like Password Hardened Encryption (PHE) services [26] are promising candidates for securing system components where classic challenge-and-response techniques are insufficient. Protocol implementations of UDS, for example, feature a security seed mechanism that hinders attackers from getting advanced security access. It is shown, however, that weak ciphers or a badly executed implementation of such protocols still allow for successful attacks, e.g., shown by Garcia et al. [27].

D. Security Techniques during Software Test

The implemented countermeasures for the attack vectors analyzed in design phase are tested during software test. Software test in the V-Model verifies that the software is built according to the specification given by the client. Additionally, security testing techniques are quite different to classic software test. Mostly—as is later shown in the evaluation section—security techniques are not even part of the software engineering process. Penetration testing and vulnerability assessments are familiar techniques to check a system for vulnerabilities and security measures. Currently, OEMs are starting to integrate those techniques into their existing processes. Known penetration testing techniques are to reverse engineer bus network traffic or disassembling ECU firmware images trying to get access to keys, secrets or specific messages.

E. Security Techniques during Maintenance

Vehicles have to be maintained and tested after delivery, thus over-the-air (OTA) updates are important, albeit challenging, because there is no secure OTA interface, yet. A clever solution, like PHE with an authentication scheme could resolve this issue. In addition, a fully secured and encrypted system excludes workshops and third-party service providers, which require open access, e.g., for resetting error codes, etc.

OTA updates are most often pulled and received via the infotainment unit, which has access to a 4G, LTE or 5G broadband connection. From there, each and every ECU that needs to receive an update has to get the new firmware or software patch from the infotainment unit via the CAN bus. Rolling out sensitive data, especially new firmware or security patches in case of OTA updates over the CAN bus is incredibly critical and a major liability. OEM updates must be checked and validated before they can be deployed to the range of ECUs connected to the CAN bus. Faulty network configurations and the lack of authentication checks for OTA updates and patches increase the risk of cloud and botnet attacks, e.g., Mirai [28]. Basically, cloud features and OTA updates have to be considered skeptically from the start. Even if the distribution source of the software is the OEM, attacks are still possible. A potential attacker might have found a way to distribute his malware over the OEM's infrastructure (e.g., their servers) and as a result a trust problem arises. It is fair to assume that any kind of roll-out (software updates, cloud data) is untrusted until the key distribution problem has been solved. Even if a solution for key distribution in heterogeneous CAN bus networks is developed, the number of remote attack vectors will rise harshly in comparison to the number of direct attack vectors.

IV. Automotive Attack Scenarios

This section describes the motivation of our approach. This motivation is necessary to highlight the threats and dangers of automotive attack scenarios and attack vectors. Section V will describe how to assess them in more detail with SAM.

Security goals like authenticity, integrity, confidentiality, etc, are especially important to make sure that the safety-critical software of the vehicle stays untampered. The following is a non-exhaustive list of attack vectors that cause major threats to automotive software systems:

- Injection of CAN frames from ECUs that were taken over after the remote attack (e.g., replay attacks, spamming attacks, etc.) [24], [29], [30]
- Reverse engineering of CAN frames by filtering by arbitration IDs and identifying frames via tools like cansniffer or other can-utils [31]
- Rolling out malicious (possibly unsigned) firmware to ECUs [24], [29], [30], [32], [33], [34], [35], [36]
- Gaining remote control access to the vehicle using the OEMs cloud and/or mobile application's infrastructure [33], [35], [37], [38]
- Getting SecurityAccess via Unified Diagnostic Services (UDS) [39]
- Controlling the car via Onboard Diagnostic (OBD) injection [40]
- Remotely breaking into the telematics unit [41]

- Exploiting remote keyless entry with software-defined radios [27]
- Denial of Service (DoS) attacks, e.g., as shown by Palamanca et al. [42]
- Infecting the system with ransomware. [43]

The easiest way to understand the SAM metamodel is to explain which piece of information the individual language components (on the M2 level) actually represent in a concrete M1 model. Therefore, in this section we present an already published attack, which we use to make the individual language building blocks accessible in an exemplary manner in addition to the conceptual explanation. SAM presented in this paper is a tangible solution for this kind of security analysis and security by design. All information needs to be documented in a system model that takes attack modeling for automotive software systems into account. The latest version of SAM introduces new attributes for rating these kinds of attacks.

V. Description of the Security Abstraction Model

In this section we describe our innovative contribution: a Security Abstraction Model (SAM) language specification for the automotive modeling environment as an extension for the EAST-ADL. We clarify the differences between security modeling and functional safety modeling and describe our metamodel entities of SAM to provide a comprehensive modeling environment for automotive security modeling. The entities can be used on the type-level (M1) to create functional architectures for safe and secure automotive systems. SAM is available as an open source project [44] and contains a concrete set of security modeling entities that are fully compliant to the EAST-ADL and AUTOSAR [45] specifications. As such, SAM is a proposition for an annex extending EAST-ADL with security modeling facilities, which are currently not covered by the existing language specification.

A. SAM Metamodel

For the sake of better understanding, we will use a published attack from the literature as a modeling example. In the following, brief descriptions of the attack details are given along with the SAM entity description used for representing the attack details. The full SAM metamodel is illustrated in Figure 2. Afterwards, the complete SAM model figure is shown on type-level (M1) in Figure 3.

We describe the attack in detail in the following. **The Tesla Remote Control Attack [33], [35], [37]**: This attack enables an adversary to break into the vehicle via the infotainment unit. The researchers of Tencent Keen Security Lab have demonstrated how to remotely control and steer the vehicle, how to disturb the autowipers and how to eliminate the lane detection of the vehicle.

Attack*: Represents a cyber-physical attack on the system described by an attack vector. An attack vector is a path or means by which an adversary can gain unauthorized access to a target system [20] or hurts one or more SecurityGoals. Attack vectors can be identified and extracted via attack trees. In an attack tree, child nodes are conditions which must be satisfied to make the direct parent node true. When the root is satisfied, the attack is complete. Typically, child nodes on the same level are linked with "OR" conditions. SAM uses a SubAttackGroup to also allow "AND" conditions and any other project-specific conditions between the grouped subattacks.*

The attack can be performed over a remote network connection. The vehicle user does not need to interact actively with the vehicle to allow for the attack to work. The privileges required for this attack are high. Also, impact on confidentiality, integrity and availability is high as well. As a result, an adversary can cause serious harm to passengers and other road users. As this attack is a research approach by security experts, the actual adversaries are not real-world attackers with bad intentions. They do have a knowledge-level, however, which they can provide for ill-minded attackers.

Adversary*: Attacks are performed by either an individual or the system's environment. Either way, adversaries are derivates of the system environment because they are not part of the main systems model and interact from the outside. An adversary can, however, come from within the system, e.g., from an unauthorized part or device.*

Environment*: This entity describes a collection of the environment functional descriptions. Many circumstances may be important for the attack description and a better understanding of the attack vector. Adversaries and security experts are conceptionally part of the environment. The Environment is not a newly introduced entity as it already exists in its own package, though it is extended due to the adversary's ability to use the environment for her attacks, e.g., external or real world attacks via adversarial examples [46], [47].*

In a real world scenario where adversaries with bad intentions perform the attack, the motivation behind such an attack would be to harm car occupants or other road users by crashing the vehicle. The attack therefore has a high safety relevance.

AttackMotivation*: An abstract representation of the adversary's motivations. This motivation is especially useful, when no other information, e.g., broken security goals, is available from the start. In that case, it offers an easy differentiation of the degree of severity and one can prioritise attacks according to their motivation. Moreover, it is fairly easy to find out if certain attack motivations are causing safety hazards, e.g., when tampering with safety-critical systems or modifying software-components related to the reliability of the system. The safety relevance can either be "High" (system failures), "Low" (fail-safe) or "None". It turns out that every single attack (or sub-attack) is part of a larger attack motivation. Through generalization methods we found out that there are only four higher motivations behind each attack: Harm, information retrieval, financial gain and product modification. There is also the motivation of prestige and other abstract ideals but those inherently cause consequences in at least one of the other motivations so they are not listed here. There is at least one AttackMotivation in an attack tree (its root). AttackMotivations collide with SecurityGoals. All attacks can be subsumed under one of those higher motivations:*

Harm*: A threat by an attack meant to actively or passively harm passengers and other road users, e.g., crashing the vehicle or causing a threat to other road users.*

InformationRetrieval*: A threat by an attack meant to, e.g., invade the privacy of passengers, other road users and other situational or political stakeholders, e.g., the OEM. Furthermore,*

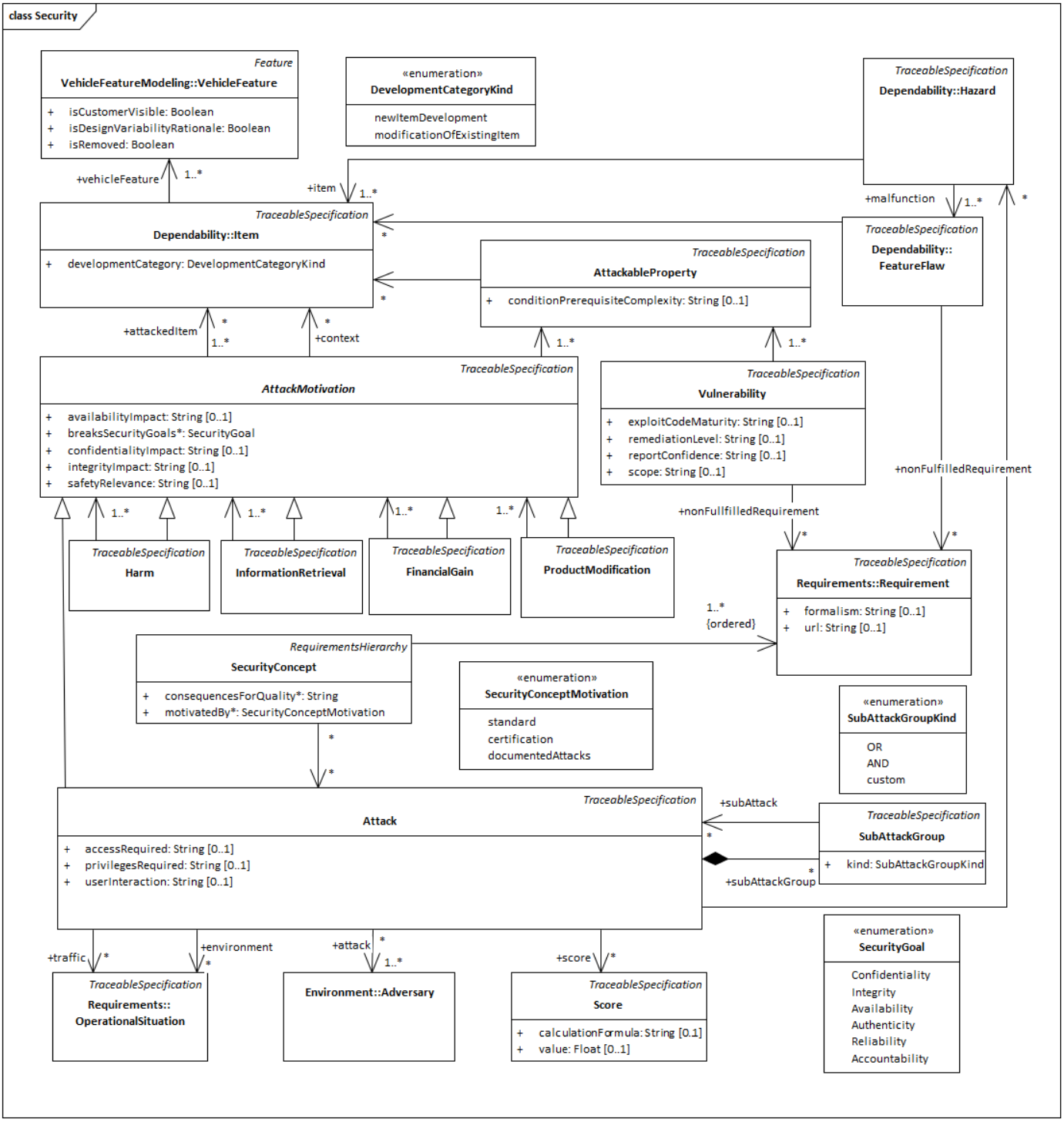


Figure 2. SAM Metamodel

getting access to other types of info, e.g., software/firmware by performing reverse engineering. Even research interest to break into the system for academic reasons is subsumed by this motivation.

***FinancialGain**: A threat by an attack meant to steal or cause financial or material gain for the adversary, service workshops or insurance companies. This usually leads to a financial loss for the owner or the OEM.*

***ProductModification**: A threat by tampering with the product's specification, e.g., getting more functionality out the car or tampering with the software in general, e.g., down-/upgrading or performance tuning.*

The affected item in the Tesla Remote Control Attack is the

AutoPilot ECU (APE).

Item: *The Item entity identifies the scope of safety/security information and the safety/security assessment. Safety/security analyses are carried out on the basis of an item definition and the safety/security concepts are derived from it.*

The exploited vulnerability in the Tesla attack is the Webkit browser framework of the infotainment unit, which offers the JSArray function and can be used for privilege escalation.

Vulnerability*: An abstract failure of a set of items, i.e., an inability to fulfill one or several of its requirements. In order to represent the weak spots in the system architecture, a vulnerability describes the weakness and affiliation to one or more Items. Vulnerabilites are concrete definitions of faulty software or configurations and requirements must be derived from them. Vulnerabilites have a scope. The scope of a vulnerability is changed, if a successful attack affects more security goals and vulnerabilities, i.e., enabling follow-up attacks.*

In case of a successful attack, adversaries can cause functional safety hazards by tampering with or disabling safety-critical functions of the Tesla vehicle.

Hazard*: The hazard metaclass represents a condition or state in the system that may contribute to accidents. The hazard is caused by malfunctioning behavior of E/E safety-related systems including interaction of these systems.*

The exploited vulnerability changes the scope of the attack, meaning that all six security goals are broken if the attack is performed successfully.

SecurityGoal*: This entity offers enumerations for common security goals [48] across any communication or data flow. These goals are: Confidentiality, Integrity, Availability, Authenticity, Reliability and Accountability.*

The exploited vehicle feature is Tesla's Autopilot, used for autonomous driving. In this case there exists a 1:1 relationship between item and vehicle feature.

VehicleFeature*: Provided by the Dependability package, a VehicleFeature represents a special kind of feature intended for use on Vehicle Level. Items consist of a set of VehicleFeatures.*

The JSArray function is the attackable property the adversary is looking for, i.e., his anchor of the attack. Attackable properties are concrete characteristics that describe the potential attack surface, e.g., if they have known security bugs and flaws like the JSArray function.

AttackableProperty*: Characteristics or certain properties of Items an adversary searches / needs for his attack to succeed, e.g., wireless communication capabilities, used ciphers, features, etc. If exploited, attackable properties allow the adversary to successfully perform an attack and define a vulnerability.*

After analysing the attack properties via the CVSS metrics, one can calculate the base score and temporal score of the attack and derive the requirement: code signing protection for over-the-air (OTA) updates.

Score*: Score is the entity for attack rating. SAM allows for any generic type of scoring system. Properties of other entities will provide all the relevant information that are needed for attack rating. The attribute calculationFormula describes which scoring system is used, e.g., CVSS, SecL, etc. Alternatively, an empirical value or expert opinion can be given if this attribute is left empty. Section V-C will explain scoring systems for SAM in broader detail.*

Requirement*: To define requirements to fix vulnerabilities, a so-called requirement is the packed result of lesson's learned and is derived from an attack. It represents a capability or condition that must (or should) be satisfied.*

The described attack is only possible when the vehicle is "Slow or Standing". Otherwise, the vehicle does not allow to use the Webkit browser. The Tesla remote control attack is possible in any operational situation of the vehicle. Once the adversary has gained full access over the system she can fully control the system over the CAN bus.

OperationalSituation*: In security modeling, it is often beneficial to know about this situation, e.g., whether the car is standing, driving or in parking mode. An operational situation is a state, condition or scenario in the environment that may influence the vehicle. It may be further detailed by a functional definition in the EnvironmentModel. Examples are: "Driving on highway", "Driving in city", "In reverse gear", "Parking", "Any", etc.*

SAM has no explicit specifications for a security concept. However, SAM proposes Common Criteria (CC) ISO/IEC 15408 protection profiles [49] as a possible solution. Common Criteria is an established standard in the security domain to provide guidance during the development of dependable systems.

SecurityConcept*: Represents the set of security requirements that together fulfill at least one SecurityGoal. An exemplary structure and classification of respective security requirements can be found in Common Criteria (CC) ISO/IEC 15408. Ideally, the security concept is motivated by analyses of the documented attacks connected to the respective item (in this case, the motivatedBy property is set to "documentedAttacks"). Otherwise, security concepts can just as well be motivated by standard or certification demands (then, the motivatedBy property is set to either "standard" or "certification".)*

SecurityConceptMotivation*: This entity offers enumerations for motivations of security requirements. These motivations are: "standard", "certification" and "documentedAttacks".*

B. Methodical Context for SAM

In order to protect and defend a system from attacks and threats it is necessary to identify and classify these threats first. The categorization of AttackMotivations already creates methodological benefits with regard to the identification of attacks. Systematic security analyses can be used to quantify the required effort for a potential attack. There is a constant battle between the attacker's efforts and the layers of security devised by system engineers. Because no system can be completely secured against all sorts of attack, system engineers compromise on varying levels of security abstractions to reach an acceptable

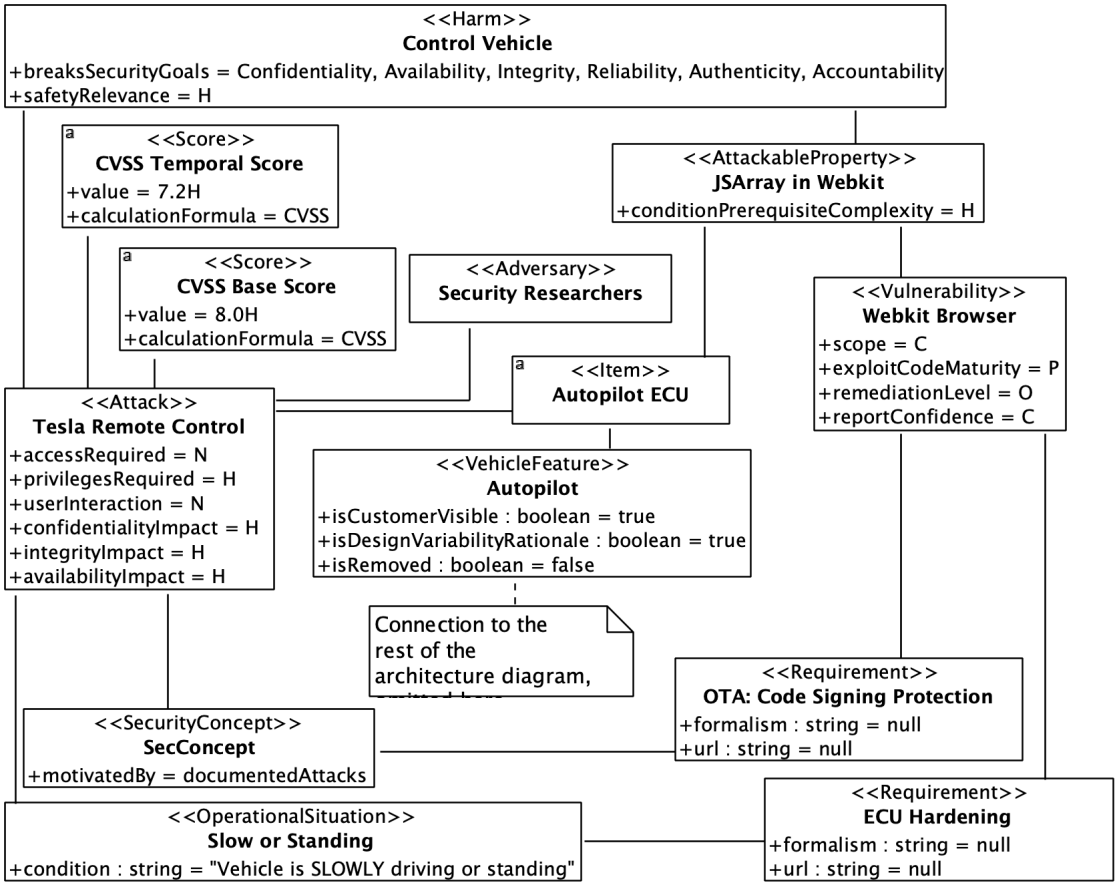


Figure 3. SAM model of Tesla Remote Control Attack—CVSS v3.0 Vector String: CVSS:3.0/AV:N/AC:H/PR:H/UI:N/S:C/C:H/I:H/A:H/E:P/RL:O/RC:C

degree of security. Hence, any security system ultimately results in a trade-off.

Although SAM does not instill security in the system design, it enforces reflection about attacks and their consequences for the system, ideally as a collaboration between system engineers and security experts. While SAM's metalevel is rather abstract, its application becomes concrete on metalevel M1. Notice that the multiplicity from AttackMotivation to Item is 1..* to 1..*, requiring the system engineer to describe at least one attack motivation for every item of the automotive system. This is an important methodical support for the discovery of threats. If a single item has no associated motivation for an attack, increased caution is required, e.g., because no attack against the item is known yet. In this case, system engineers might simply desist to scrutinize an item for possible attack motivations. With the 1..* multiplicity, however, they are forced to think about at least one attack motivation for every item. Therefore, the reason for this decision is to eagerly enforce the methodology of SAM's security approach.

The main difference between safety risks and security threats is that security threats do not happen at random (i.e., they are not bound by probability) but always occur in worst-case scenarios. For safety hazards, a statistical probability can be assumed. Cyber attacks are performed by an intelligent attacker at the most suitable time for the adversary and at the lowest defense barrier. Furthermore, it can be misleading to confuse safety goals with security goals. Security threats, however, can cause safety hazards and vice-versa. Though it is not recommended to treat them in the same way during the system design phase for reasons mentioned above. Additionally, text annotations are bad practice. Usually, the transfer from annotations in natural language is imprecise and the original intent of the security experts, which is needed to represent the system model and its security mechanisms accordingly, might be lost during the transfer. An extensive reuse of security solutions can be established by embedding SAM in the "Dependability" package of EAST-ADL and the subsequent integration into AUTOSAR. This makes it possible to keep the development effort at a minimum and to implement comprehensive safety and security solutions in a wide range of applications in the vehicle.

SAM offers the possibility to model socio-technical systems by providing the modeling entity `Adversary`. Security goals need to be fulfilled in a *socio-technical context* or a *socio-technical system*. The definition of a socio-technical system is an organized group of humans and connected technologies, which are constructed in a certain manner to produce a specific result [48]. Nevertheless, trying to improve security simply by adding cryptography to the system is a fallacy. At best, cryptography can ensure confidentiality but cannot cover security goals like availability, reliability or accountability. With our approach, we offer co-engineering processes of security and safety for automotive software engineering (security and safety by design).

C. Using Generic Scoring Systems for SAM

The latest major release of SAM [1] introduced many new attributes to the modeling entities, which allow for using well-known security scoring systems like CVSS [21]. In order to be able to keep SAM up-to-date and gain some flexibility by not making a strong commitment to one particular system we designed SAM to use any generic scoring system. When modeling attack scenarios, users of SAM can choose among their favorite. Here, we will use the CVSS. The latest version of SAM is available open source [44]. The architecture description has been completed to the extent that common scoring systems are now able to find the necessary information and thus perform their analyses. Inspired by the CVSS, which is an acclaimed industry standard for rating vulnerabilities in computer systems, we added new attributes to some of SAM's entities. The CVSS proposes three different metric groups for calculating the vulnerability scores. In the following, an explanation of the interplay between SAM and the metrics is given. The assignment of the attributes to the meta entities and partly their naming does not come from CVSS, but was developed by the athors.

The Base Metric Group reflects the intrinsic properties of Attack: from SAM's automotive-oriented perspective, this group therefore indicates the characteristics that result if the attack in question is aimed at the automotive domain in general. The entity AttackableProperty refers to the properties of the attacked item that are beyond the control of the attacker and must exist in order to exploit the vulnerability, for example, in the case of a side channel attack, the use of shared caches within a multicore system. The attribute `conditionPrerequisiteComplexity` ("Low" and "High") in the AttackableProperty refers to the complexity of encountering or creating such conditions. For example, in the case of the side channel attack mentioned above, the `conditionPrerequisiteComplexity` is "Low" because shared caches are to be expected nowadays. It would be "High" if the attack made it necessary for all tasks on all cores to use one single common cache. When evaluating this property, all user interaction requirements for exploiting the vulnerability must be excluded (these conditions are recorded in the property `privilegesRequired` of Attack instead). If the `conditionPrerequisiteComplexity` is "Low", the attack is more dangerous than if the `conditionPrerequisiteComplexity` is "High". The property `privilegesRequired` describes the level of privileges an attacker must possess before successfully exploiting the vulnerability. This metric is greatest if no privileges are required. Also, the Attack entity has been extended with the attributes `accessRequired` and `userInteraction`. The attribute `accessRequired` describes the context in by which vulnerability exploitation is possible. It must not be confused with general attack vector handling in SAM, which describes the path from attack motivation of an attack tree to one of its leafs. Whether the user or driver of the vehicle needs to interact with the system in a certain way, e.g., by pressing a button, is captured in `userInteraction`. Attacks that do not require any user interaction increase the score of the attack. **The Temporal Metric Group** allows for adjustment of the score after more information of the exploited vulnerability is available. If, for example, `exploit code` has been published or the `report confidence` of a vulnerability is confirmed, the temporal score rises. In SAM, temporal metrics are part of the vulnerability. **The Environmental Score Metrics** additionally enable the general CVSS Score (resulting from the Base Metric Group) to be adapted to the specific (automotive) company. The metrics are the modified equivalent of the base metrics weighting properties related to the concrete company's infrastructure and business risk. SAM offers a fully comprehensive basis to analyse the CVSS Base Metric Group, which means that SAM can also be used to evaluate the Environmental Metric Group. Environmental Metrics do not require any additional information beyond the Base Metrics, but merely a readjustment of the analysis perspective towards the concrete company. This means that the security scoring analysis can be carried out entirely by an analyst based on the available information provided by SAM.

This allows for more flexibility and SAM does not have to be adapted for any future CVSS updates. All attributes used for attack assessment are of the type String. This allows for SAM to be used with generic assessment techniques and is not tightly coupled with the CVSS attribute descriptions. In the model itself, or from the model itself, a CVSS score cannot be calculated automatically anyway. Doing so would happen in a behaviour model while SAM models are structure models. But if a security analyst is familiar with the CVSS, she will be able to calculate the CVSS score with all the information that is provided by the structure model. It is therefore still possible to find related information about the attribute types ("High" and "Low", etc.) in the notes of the meta model, but does not lead to problems in case of non-compliance.

The additional benefit of having SAM models compared to directly giving the properties and a vulnerability score is that not only the CVSS (or scoring systems in general) is used, but also the possibility to construct attack trees via sub-attacks and follow-up attacks. SAM is also a method for hierarchical processing of attack vectors. In terms of substance, this goes beyond the classic attack rating. SAM makes the scoring system available to the software architect or in other words: SAM's strength lies in its ability to integrate with existing automotive architectures. What is brought together are architectural considerations with pure security considerations as regards the attack itself (attack vectors that can be derived from it, motivations, target areas) and all scoring systems that are known, which can derive all necessary information from the properties.

VI. Evaluation

To prove that SAM is feasible, we have evaluated our solution approach through "Grounded Theory" [50] interviews with two experts (in the following we will refer to them as E1 and E2) of two different automotive software companies from the industry. Both interview partners have a professional background in automotive systems engineering and/or embedded security. The interviews were structured as follows: First, we asked some general questions about automotive security modeling and current processes in the analysis, design, implementation and test phase. Afterwards, the authors gave a brief introduction and explanation of SAM. During the presentation, a real-world attack was shown and illustrated to show how to use SAM. Together with the interview partners, we jointly created a reference SAM model for a real-world usecase. Figure 4 illustrates a Function Unlock Attack. The

attack describes an attack vector that allows an adversary (here: a third party workshop) to unlock specific vehicle functions for the driver of the vehicle via tampering with the TimeServer item. The answers for the remainder of our interview were related to this modeling example. Finally, the industry experts were asked questions about the five categories: suitability, scalability, comprehensibility, completeness and tool support.

A. General Findings

Our interviews have shown that there is no clear process for security engineering, yet. Companies are in the process of creating them. Hopes are that the process will be similar to the ISO 21434 and ISO 26262 standards for safety, which is currently seen as a general guideline for automotive engineers. The SAE J3061 standard would also be a good starting point if engineers want to get started with this matter. Our interviews show that a detailed development process for security requirements is highly desired. Many projects in the industry are done by service providers, where the instructed companies integrate security into the finished standard components. On the part of Tier 1 there are also no concrete specifications for the security development process

Both experts confirmed that they are indeed working according to the V-Model. They have quite clear process models. Ideally, they get a specification, write a requirement specification, create an architecture for the software, create module specifications, derive tests from them, test the modules, then test what is integrated. Depending on what needs to be developed in the end, they then develop a piece of software or an entire ECU. Afterwards, they go up (in the V-Model) until they have finally tested all the specification requirements. They do this for all requirements.

For some requirements, security is not in the focus. Even if security is mentioned at all, it is only given implicitly by implementation instructions or which software to use. Putting a security concept in place especially for all phases of the V-Model would be desirable, even though integration might be challenging.

B. Suitability

We asked the experts how existing processes could use SAM in their current workflow or if it could even be integrated into their processes. We also explained the intention of SAM to facilitate the exchange between security experts, software architects and software engineers and asked, if it serves this purpose. We tried to find out if SAM is a suitable solution for the industry and whether or not it solves current industry questions. Finally, for this category, we wanted to know if SAM is ready for the development of autonomous vehicle systems and what could be missing. The following is a summary of the experts' answers:

Both experts agreed that they could use SAM because it is a good way of making attacks comparable. A missing process is that items have to be seen in a way that potentially there has to be an attack for each item at the time of module specification. The main problem is that the process is actually driven only once during development. Attacks, however, only happen during maintenance. A visualization (like SAM models) would help, however. A possible solution of integration SAM would be using SAM models like bug reports for security. With SAM, they would already have the constraints available. The current process currently only entails development and delivery. Teams would need a kind of "response team for security". Someone who monitors the whole systems and knows what kind of hardware-software combinations are rolled out and what kind of errors there are, e.g., SPECTRE [51]. This could suddenly affect a huge portion of products that are out in the field. The response team could classify attacks like this with SAM.

Integrating SAM into workflows is rather difficult for service providers, but on OEM side it would very well possible. Analogue to considering ASIL levels, i.e. according to ISO 26262, SAM could be well integrated on the OEM side. SAM increases transparency and simplifies understanding attacks. After a training of the team members, SAM would definitely help and make communication easier, especially because there are no alternatives. It can be, however, that one could not understand a bad SAM model without additional textual description.

Regarding suitability for autonomous driving, the experts were unsure if this is even possible to answer at this point. SAM would first have to prove itself in practice and people would have to work with it so that it could be finally used. Even if systems would become simpler, software development would probably become more complex, because there would be many more lines of code. Moreover, security has to guarantee that the safety mechanisms still work and that there is a balance of availability, security and safety. If a car goes into fail-safe mode with the slightest security suspicion, autonomous driving is not possible in this scenario.

According to one expert, some modeling entities for machine learning components could be useful. This is rather challenging, though, as machine learning components are black boxes.

C. Comprehensibility

We asked the industry experts if something about SAM is difficult to understand. Although they understood everything, an introduction with examples would help beginners getting started, because SAM is not that formal, rather practical.

D. Scalability

For this category we evaluated for what complexity or size of a software project SAM is best suited and how developers would ideally use SAM in practice.

Here, the industry partners disagreed. Although one could easily create multiple SAM models for more complex scenarios, one expert believes that SAM might not scale well for bigger projects. There would be no way to see the overall security for, e.g., a whole vehicle. To scale it, one would have to link several SAM models together. Overall models could get confusing because everything always depends on one item and vehicles are very complex. However, he said, SAM would be the right approach for smaller projects.

According to one expert, automotive engineers definitely have to allocate time for security engineering. SAM would help with that, because in the end, it would save time if everyone speaks the same language. There would have to be support for security techniques. Central offices that have an overview of which products, which software and which hardware are currently in the field could use SAM to determine that.

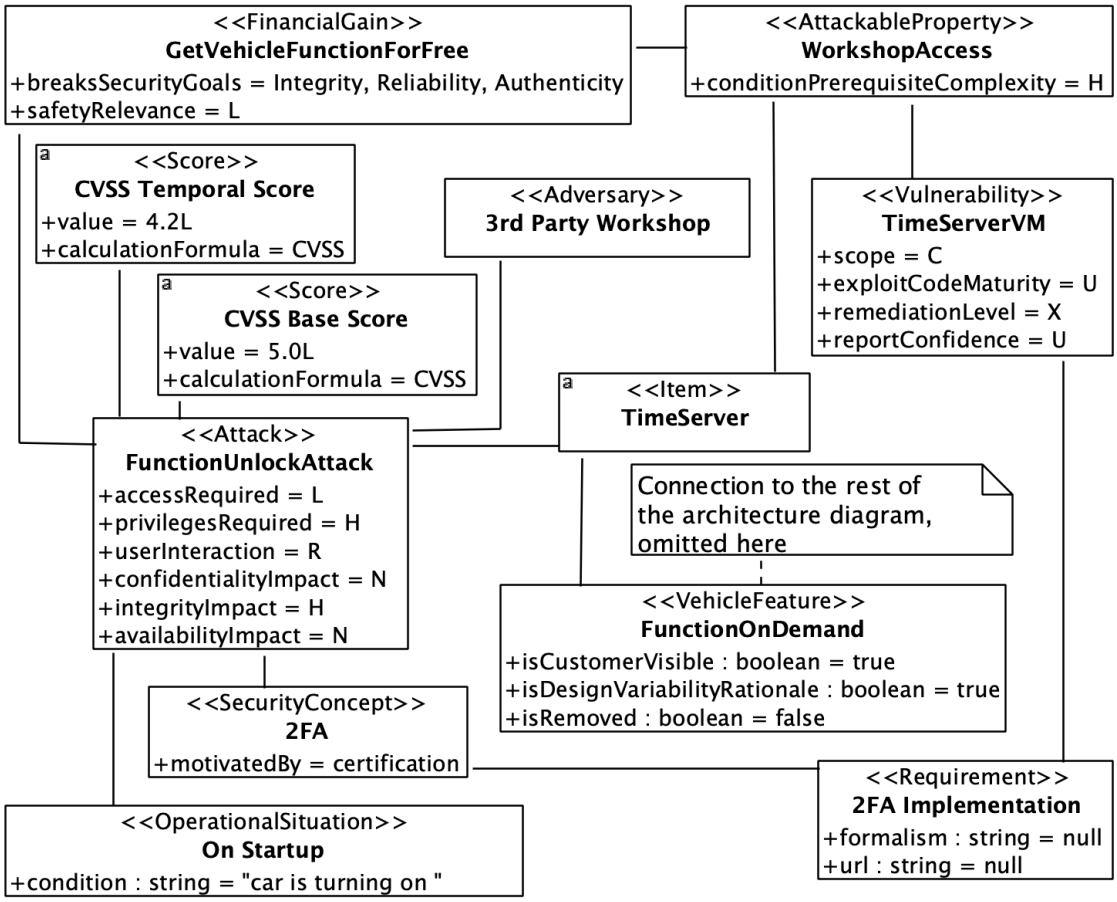


Figure 4. SAM model of Function Unlock Attack—CVSS v3.0 Vector String: CVSS:3.0/AV:L/AC:H/PR:H/UI:R/S:C/C:N/I:H/A:N/E:U/RC:U

In contrary, developers might not want to use SAM because it is on a relatively high level of abstraction. Fortunately, SAM has close familiarity to UML and developers could get used to it quickly. Traceability over requirements would be a good approach to make them adopt SAM, because they work very strongly according to requirements and the V-Model and they can validate their requirements.

E. Completeness

To make sure that SAM is indeed complete, we asked if the explanations and attributes are complete or what could be deleted from SAM. Moreover, we asked what a possible solution would look like for how to add tests to SAM.

During the discussion with the experts it became clear that some entities cover redundant information at that time. The current version presented in this paper, however, includes the latest changes, even the refined changes taken into account after the interviews. Furthermore, one expert had the opinion that the score entity might be redundant because all the relevant information is already in the entities' properties and could also be calculated that way. This is theoretically redundant information. If one would have a modeling tool now, it would display it permanently and calculate it in the background. It would not have to exist as an entity. As of right now, it is shown in its own separate entity.

Overall feedback showed that if users of SAM are already familiar with security, one could understand all the terms and explanations presented in SAM. SAM is doing a good job of illustrating the weak points. So one can deduce test cases from the model, make them more concrete and test countermeasures to achieve a certain CVSS score. One could use the score to adapt the tests and cover the fact that, for example, an attack is no longer possible and, if necessary, derive further test cases. However, SAM would actually only show the problem to engineers: the attack scenario. The resulting actions would be twofold: Fixing and testing the bug. After the bug is found, something in the architecture must be changed. Reusability is an important factor. Someone would have to test the change and then has to make sure that this bug will not happen again in future models or products. Someone could consider whether he can make the test abstract so that it will be fired in the future. However, one would need a lot more details to automatically turn it into a test. The tester should, however, be able to look at the model and write a test from it himself.

F. Tool Support

There is no seamless tool support from SAM/EAST-ADL to AUTOSAR as of right now. We asked whether or not it is needed. Furthermore, we asked if an automatic generation of SAM models (e.g., from tests or code) would be useful. Finally, we asked for remaining comments, hints and feedback.

One expert was not sure if SAM must fit directly to AUTOSAR. SAM would be useful for modeling attacks, AUTOSAR is used for modeling software. Tool support, where

one could create SAM models with, or where the score is calculated automatically would be helpful, though, and improve usability. If SAM is easy to use and a suitable tool would exist, developers would use it for sure.

The other expert disagreed and told us that an automatic generation of SAM models would not be necessary at the moment. He also said that the industry does not know whether it will stay with AUTOSAR or not. However, SAM can be used independently at the moment. His experience with AUTOSAR told him that he would not build too much onto it. One could automatically generate or update SAM models as post-processing after modeling, just as one can already generate code or header files, templates, etc. Leaving SAM the way it is right now, i.e., without automatic generation, someone has to think about security himself. It has to be thought of from the beginning.

General advice we received, suggested a "Getting Started" or "HowTo" document. This would be helpful for acceptance in practice. Finally, SAM should be used more on the OEM side, because vehicles are very complex products and only the product owner currently have a complete overview over the vehicle's components. With SAM, someone can sufficiently document security thoughts and research, including evaluation.

G. Overall Results

Our industrial evaluation has shown that SAM is indeed suitable as a solution approach and integration of the methods used into industry processes is feasible. Moreover, SAM is easy to understand according to our interviewed industry experts. Although SAM might not scale for bigger software projects, it establishes a process to get started with on a smaller scale. Also, our evaluation shows that SAM is indeed complete and offers enough tools, methods and descriptions for threat modeling and attack rating. Evaluation results regarding tool support were twofold. Further investigations on a practical tool support are necessary.

Table II shows a summary of the evaluation results. The symbol keys are explained in Table I.

TABLE I. Symbol key for the evaluation results table

Symbol	*Meaning*
++	High agreement or interest (E1 and E2 agreed)
+	Notable agreement or interest (E1 or E2 agreed)
o	Moderate agreement or interest (neither E1 or E1 showed interest)
-	Limited agreement or interest (E1 or E2 disagreed)
--	Low agreement or interest (E1 and E2 disagreed)
+/-	Mixed agreement or interest (E1 and E1 were of different opinion)

TABLE II. Summary of the evaluation results

Categories and codes	*Results*
General	
Process: Along the V-Model	++
Missing process for security	++
Suitability	
Processes can use SAM	++
Integration in process	++
Exchange between security experts, architects and engineers	++
Solves relevant industry challenges	o
Ready for autonomous driving (AD)	+
Is something missing for AD?	+/-
Comprehensibility	
SAM is easy to understand	++
Scalability	
Scales for all sizes of projects	+/-
Developers have time to use SAM	+/-
Completeness	
Nothing should be removed from SAM	o
Descriptions and entities are complete	++
Use SAM for testing	++
Tool Support	
AUTOSAR tool support	+/-
Automatic generation of SAM models	--

VII. Conclusion and Future Work

We have presented a detailed description of the Security Abstraction Model, including all of its metamodel entities. We have indications that the approch is feasible. The security technique has been evaluated with industry experts and a grounded theory analysis. The resulting analyses of the evaluation show that SAM puts the security-by-design principle into practice by enabling collaboration between automotive system engineers and security experts. Future work will concentrate on the bottom-up approach, i.e., improving embedded security and network security on the application layer and cryptographic protocol design, e.g., utilizing PHE. Next steps need to develop automotive software solutions to actually be included in the security concept. Our research focuses particularly on a PHE authentication scheme for secure authentication in autonomous car sharing scenarios and fingerprint entry systems. Our work aims to support security by design in the automotive industry and SAM offers the necessary insights and fundamentals to continue conducting relevant research in this domain.

Acknowledgment

This work is funded by the Bavarian State Ministry of Science and the Arts in the framework of the Centre Digitisation.Bavaria (ZD.B).

M.Z. was supported by the BayWISS Consortium Digitization.

References

[1] M. Zoppelt and R. Tavakoli Kolagari, "UnCle SAM : Modeling Cloud Attacks with the Automotive Security Abstraction Model," in CLOUD COMPUTING 2019, The Tenth International Conference on Cloud Computing, GRIDs, and Virtualization, Venice, Italy, 2019, pp. 67–72.

[2] S. Tuohy, M. Glavin, C. Hughes, E. Jones, M. Trivedi, and L. Kilmartin, "Intra-Vehicle Networks: A Review," pp. 534–545, 2015.

[3] W. Zeng, M. A. Khalid, and S. Chowdhury, "In-vehicle networks outlook: Achievements and challenges," IEEE Communications Surveys and Tutorials, vol. 18, no. 3, 2016, pp. 1552–1571.

[4] ISO/IEC, "ISO/IEC 15408-1:2009 - Evaluation Criteria for IT Security," vol. 2009, 2009, p. 64.

[5] A. Happel and C. Ebert, "Security in vehicle networks of connected cars," 15. Internationales Stuttgarter Symposium: Automobil- und Motorentechnik, no. March, 2015, pp. 233–246.

[6] M. Zoppelt and R. Tavakoli Kolagari, "What Today's Serious Cyber Attacks on Cars Tell Us: Consequences for Automotive Security and Dependability," in International Symposium on Model-Based Safety and Assessment, M. Papadopoulos, Yiannis and Aslansefat, Koorosh and Katsaros, Panagiotis and Bozzano, Ed., Springer. Springer International Publishing, 2019, pp. 270–285. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-32872-6_18

[7] M. Zoppelt and R. Tavakoli Kolagari, "SAM: A Security Abstraction Model for Automotive Software Systems," in Security and Safety Interplay of Intelligent Software Systems. Springer, 2018, pp. 59–74.

[8] H. Blom, H. Lönn, F. Hagl, Y. Papadopoulos, M. Reiser, C. Sjöstedt, D. Chen, and R. Tavakoli Kolagari, "EAST-ADL–An Architecture Description Language for Automotive Software-Intensive Systems–White Paper Version 2.1. 12," Hyperlink: http://www.maenad.eu/public/conceptpresentations/EAST-ADL_WhitePaper_M2 [retrieved: December 2018], vol. 1.

[9] ISO/SAE, "ISO/SAE CD 21434 - ROAD VEHICLES – CYBERSECURITY ENGINEERING," https://www.iso.org/standard/70918.html.

[10] W. Dröschel, W. Heuser, and R. Midderhoff, Inkrementelle und objektorientierte Vorgehensweise mit dem V-Modell 97. München: Oldenbourg, 1998.

[11] SAE, "SAE J 3061 - Cybersecurity Guidebook for Cyber-Physical Vehicle Systems," https://www.sae.org/standards/content/j3061/.

[12] N. Bißmeyer, S. Mauthofer, J. Petit, M. Lange, M. Moser, D. Estor, M. Sall, M. Feiri, R. Moalla, M. Lagana, and F. Kargl, "PREparing SEcuRe VEhicle-to-X Communication Systems," 2014.

[13] O. Henniger, L. Apvrille, A. Fuchs, Y. Roudier, A. Ruddle, and B. Weyl, "Security requirements for automotive on-board networks," in 2009 9th International Conference on Intelligent Transport Systems Telecommunications, ITST 2009. IEEE, 2009, pp. 641–646.

[14] H. Holm, M. Ekstedt, T. Sommestad, and M. Korman, "A Manual for the Cyber Security Modeling Language," 2013, p. 110.

[15] J. Jürjens, "UMLsec: Extending UML for Secure Systems Development," in International Conference on The Unified Modeling Language. Springer, 2002, pp. 412–425.

[16] INCOSE, "Systems Engineering Handbook," in Systems Engineering, no. August, 2000.

[17] R. Ross, M. McEvilley, and J. Carrier Oren, "Systems Security Engineering: Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems," vol. 160, no. November 2016, 2016.

[18] J. Lee, B. Bagheri, and H.-a. Kao, "A Cyber-Physical Systems architecture for Industry 4 . 0-based manufacturing systems," Manufacturing Letters, vol. 3, 2015, pp. 18–23.

[19] S. P. Kadhirvelan and A. Söderberg-Rivkin, "Threat Modelling and Risk Assessment Within Vehicular Systems," Chalmers University of Technology, no. August, 2014, p. 52.

[20] V. L. Thing and J. Wu, "Autonomous Vehicle Security: A Taxonomy of Attacks and Defences," in Proceedings - 2016 IEEE International Conference on Internet of Things; IEEE Green Computing and Communications; IEEE Cyber, Physical, and Social Computing; IEEE Smart Data, iThings-GreenCom-CPSCom-Smart Data 2016, 2017, pp. 164–170.

[21] Common Vulnerability Scoring System [retrieved: April, 2019]. [Online]. Available: https://www.first.org/cvss/

[22] G. Macher, A. Höller, H. Sporer, E. Armengaud, and C. Kreiner, "A combined safety-hazards and security-threat analysis method for automotive systems," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9338, 2015, pp. 237–250.

[23] Bosch, "CAN Specification," Robert Bosch GmbH, 1991.

[24] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," Defcon 23, vol. 2015, 2015, pp. 1–91.

[25] W. Diffie and M. Hellman, "New directions in cryptography," IEEE transactions on Information Theory, vol. 22, no. 6, 1976, pp. 644–654.

[26] R. W. F. Lai, C. Egger, M. Reinert, S. S. M. Chow, M. Maffei, and D. Schröder, "Simple Password-Hardened Encryption Services," in 27th {USENIX} Security Symposium ({USENIX} Security 18). Baltimore, MD: {USENIX} Association, 2018, pp. 1405–1421. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/lai

[27] F. D. Garcia, D. Oswald, T. Kasper, and P. Pavlidès, "Lock It and Still Lose It—On the (In)Security of Automotive Remote Keyless Entry Systems," Proceedings of the 25th USENIX Security Symposium, 2016, pp. 929—-944.

[28] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: Mirai and other botnets," Computer, vol. 50, no. 7, 2017, pp. 80–84.

[29] C. Valasek and C. Miller, "Adventures in Automotive Networks and Control Units," Technical White Paper, vol. 21, 2013, p. 99.

[30] C. Miller and C. Valasek, "CAN Message Injection," 2016, pp. 1–29. [Online]. Available: http://illmatics.com/can message injection.pdf

[31] can-utils repository on GitHub [retrieved: April, 2019]. [Online]. Available: https://github.com/linux-can/can-utils

[32] T. Bécsi, S. Aradi, and P. Gáspár, "Security issues and vulnerabilities in connected car systems," in 2015 International Conference on Models and Technologies for Intelligent Transportation Systems, MT-ITS 2015, 2015, pp. 477–482.

[33] S. Nie, L. Liu, Y. Du, and W. Zhang, "Over-the-Air : How We Remotely Compromised the Gateway , Bcm , and Autopilot Ecus of Tesla Cars," Defcon, vol. 1, 2018.

[34] T. Zhang, H. Antunes, and S. Aggarwal, "Defending connected vehicles against malware: Challenges and a solution framework," IEEE Internet of Things Journal, vol. 1, no. 1, 2014, pp. 10–21.

[35] Tencent Keen Security Lab, "Experimental Security Research of Tesla Autopilot," 2019, p. 38.

[36] C. Smith and S. Francisco, THE CAR HACKER ' S HANDBOOK A Guide for the Penetration Tester About the Contributing Author About the Technical Reviewer, 2016.

[37] S. Nie, L. Liu, and Y. Du, "Free-fall: hacking tesla from wireless to can bus," Defcon, 2017, pp. 1–16.

[38] Tencent Keen Security Lab, "Experimental Security Assessment of BMW Cars: A Summary Report," 2018.

[39] J. den Herrewegen and F. D. Garcia, "Beneath the Bonnet: A Breakdown of Diagnostic Security," in European Symposium on Research in Computer Security. Springer, 2018, pp. 305–324.

[40] Y. Zhang, B. Ge, X. Li, B. Shi, and B. Li, "Controlling a Car Through OBD Injection," in Proceedings - 3rd IEEE International Conference on Cyber Security and Cloud Computing, CSCloud 2016 and 2nd IEEE International Conference of Scalable and Smart Cloud, SSC 2016, 2016, pp. 26–29.

[41] I. Foster, A. Prudhomme, K. Koscher, and S. Savage, "Fast and Vulnerable: A Story of Telematic Failures," 9th USENIX Workshop on Offensive Technologies (WOOT 15), 2015.

[42] A. Palanca, E. Evenchick, F. Maggi, and S. Zanero, "A stealth, selective, link-layer denial-of-service attack against automotive networks," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), M. Polychronakis and M. Meier, Eds. Cham: Springer International Publishing, 2017, vol. 10327 LNCS, pp. 185–206.

[43] T. Ring, "Connected cars - The next target for hackers," Network Security, vol. 2015, no. 11, 2015, pp. 11–16.

[44] SAM repository on Bitbucket [retrieved: April, 2019]. [Online]. Available: https://bitbucket.org/east-adl/sam

[45] AUTOSAR Enabling continuous innovations https://www.autosar.org [retrieved: July, 2019]. [Online]. Available: https://www.autosar.org/

[46] J. Hayes and G. Danezis, "Machine Learning as an Adversarial Service: Learning Black-Box Adversarial Examples," vol. 2, 2017.

[47] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in Machine Learning: From Phenomena to Black-Box Attacks Using Adversarial Samples," 2016.

[48] F. Dalpiaz, E. Paja, and P. Giorgini, "Security requirements engineering via commitments," in 2011 1st Workshop on Socio-Technical Aspects in Security and Trust (STAST). IEEE, 2011, pp. 1–8. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6059249

[49] H. C. A. Van Tilborg and S. Jajodia, Encyclopedia of Cryptography and Security. Springer Science & Business Media, 2011. [Online]. Available: http://link.springer.com/10.1007/978-1-4419-5906-5

[50] B. G. Glaser, A. L. Strauss, and E. Strutzel, "The discovery of grounded theory; strategies for qualitative research." Nursing research, vol. 17, no. 4, 1968, p. 364.

[51] P. Kocher, J. Horn, A. Fogh, , D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in 40th IEEE Symposium on Security and Privacy (S&P'19), 2019.

Enabling Financial Reports Transparency and Trustworthiness using Blockchain Technology

Van Thanh Le, Claus Pahl, Nabil El Ioini,

Faculty of Computer Science,
Free University of Bolzano,
Bolzano, Italy
Email: {vanle,cpahl,nelioini}@unibz.it

Gianfranco D'Atri

Department of Mathematics and
Computer Science,
Calabria University,
Rende, Cosenza, Italy
Email: gdatri@mat.unical.it

***Abstract*—Financial report activity standardization becomes more essential in Financial Services and Technology, to facilitate the digitization of the process of communicating and acquiring business information. The eXtensible Business Reporting Language (XBRL) is one of the first steps towards this vision by providing a general digital format of financial reports with different rules and tags. Analyzing XBRL reports allow us to verify the quality and transparency of the data and well as have the full history of the stored transactions. Currently, checking and storing reports are independent for each organization and country, which is less transparent for the public and investors, who might be interested in checking company's records before investing in them. In this paper, we propose a blockchain-based solution where all reports analysis activities and results are recorded into a shared ledger to guarantee their transparency and trustworthy. Specifically, we design and implement a prototype to evaluate and store financial statements using Ethereum blockchain following special metrics. Moreover, we examine different architectural decisions in terms of cost and performance and look at their advantages and disadvantages.**

Keywords–Blockchain; XBRL; Financial Reports; DLV; ASP; On-Chain Computation.

I. Introduction

Financial statements are formal records of the financial activities that companies use to provide an accurate picture of their financial history. Their main purpose is to offer all the necessary data, which allows for an accurate assessment of the economic situation of a company and its ability to attract investors.

The precision of financial reports can not be underestimated, any missing numbers or assets in a balance sheet could have a tremendous effect on a business, for instance, a company actually lose their profit because they miss their tax. The accuracy also supports to find mistakes of expenses or internal process early, monthly reports can show the problem to restructure procedures or wrong activities. Moreover, by proposing a report in detail and correctness, businesses get more trust from the community and attract more investors. In a normal balance sheet, there are more than 100 fields needing to be filled that will be a challenge for accountants if they do in the traditional way, so we need a special tool to support the process that will check any missing point, or in other words, to verify the validity of a report, as a suggestion for company managers.

Financial reports contain sensitive data that refer to the company status, following the timeline, the data even affects decision making for future work. For example, a manager wants to examine the importance of an asset to decide whether he will buy more, in case the asset is frequently deprecated every year, his decision will change. However, the owner of the assets could update its historical price in the database to hide the downward trend that makes the storage becomes unreliable. Even the distributed database is stable, high availability and performance but it is still under controlled by authorities or any third parties, thus, in the case, blockchain could bring benefits for this model.

To this end, our goal is to investigate how blockchain can be used to address these limitations to restore trustworthiness in the financial reports. Our contribution is two-fold (i) provide methodologies to automatically evaluate and validate the consistency of the generated reports in case of off-chain and on-chain, *(ii)* use Ethereum smart contract to store financial reports and track all updates that might take place in the future. Additionally, an initial set of experiments is presented to illustrate the cost and time factor of the proposed approach.

This paper develops on our previous work [1] by providing i) An off-chain evaluator based iDLV ii) a trustworthy financial report storage that is a baseline for our continuing work.

The remaining of this paper is organized as follows: Section II illustrates our reference scenario. Section III provides background information about the used technologies. Section IV discusses the main related work studies connected to our work. Section V describes the system architecture. Section VI presents the implementation details. Section VII experimentally evaluates the cost and performance of our approach and Section VII gives our conclusions.

II. Reference Scenario and Problem Statement

A. Scenario

Our reference scenario focuses on the Italian legislation. The financial statements are governed in Italy by Articles 2423. Following the Italian Civil Code, submitting financial reports is mandatory, and it needs to be done through the website of Chamber of Commercial (*webtelemaco.infocamere.it*). Every enterprise must prepare financial statements for each year or two consecutive financial years. These essential reports contain *i)* Balance Sheet, *ii)* Income Statement, *iii)* Cash Flow

Statement, and *iv)* Explanatory note. Depending on the size of the business, we can have different forms of submissions. For example, micro-enterprises are able to skip cash flow statements and explanatory notes. In our scope, we mainly focus on the balance sheet, since it is the most common statement for every company to submit.

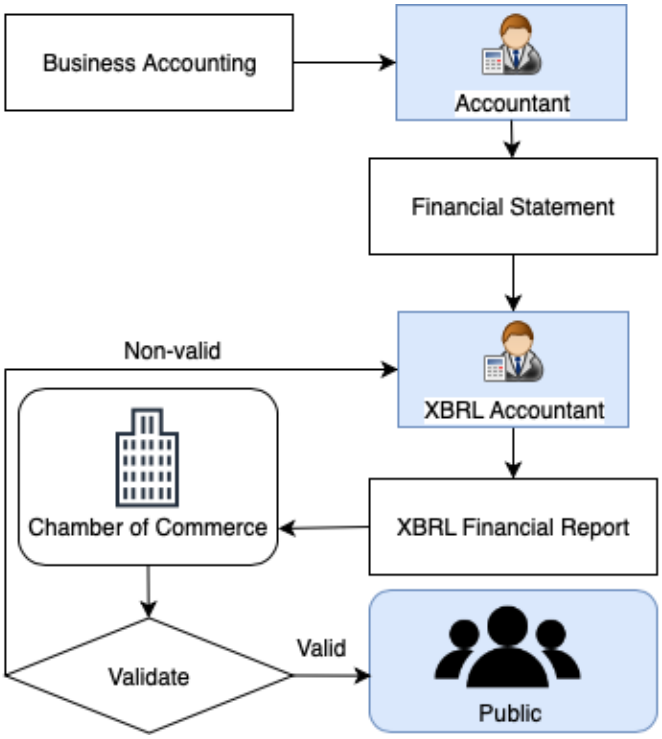


Figure 1. Financial data flow

The use case we refer to is described in Figure 1, main actors and components are:

- Business accounting is the systematic recording, analyzing, interpreting and presenting of financial information.
- An accountant is a person who records business transactions on behalf of an organization, reports on company performance to management, and issues financial statements.
- Financial statements are a collection of reports about an organization's financial results, financial condition, and cash flows.
- XBRL Accountant is a person who works with financial statements and XBRL tools to fill data to XBRL financial reports.
- XBRL is the format for delivering financial reports in an interactive digital format.
- Validating XBRL is the way we check the consistency of XBRL asset values, for example, in Italy, the validator is TEBENI, the tool provided by the Italian Chambers of Commerce.

A company *A* wants to make a financial report for this year, the manager assigns the task to the Chief Accounting Officer (CAO). The CAO then collects business accounting information from accountants as account payable, receivable, expense for wages, etc. After that, a draft version of the financial statement is made, and start preparing for XBRL version, this is a standard digital format for financial reports (XBRL stands for eXtensible Business Reporting Language). The draft is sent to a specialist called XBRL accountant. The specialist will find a suitable taxonomy following their country and business status, for example, the latest XBRL taxonomy version is *PCI_2018-11-04*, followed by [2]. The staff will map and tag financial statement elements to corresponding elements in the chosen taxonomy and use tools like Microsoft Excel or up-to-date tools in [3] to fill the data to make the instance document.

After getting an XBRL report, internal validation is necessary as a pre-check. The validation contains two steps [4]: *i)* validation of the markup and *ii)* validation of calculations. Enhanced Validation and Strict XBRL Validation are two kinds of validating calculations, in Enhanced Validation, it checks in detail child elements of a parent element, if one of them misses or the summary of calculation does not fit for both parent and child elements, an error will be released. Strict one accepts some missing elements. Because there are few companies can fulfill entire fields in taxonomy so, in our investigation, we accept some missing tag like the Strict Validation.

When the validation process finished, the CAO and the company director board will review and give permission to send the XBRL report to the Chambers of Commerce [5]). This is an association or network of entrepreneurs designed to promote and maintain the benefits of its members. The office is also considered as a board of trade that includes groups of businessmen sharing their interests even in the international scope. A chairman will be chosen to negotiate and debate with the government for policies in financial aspects and overall economic environment. In the Chambers of Commerce, there is a brief check for submitted reports before publishing it into the website of the office for the public (e.g *http://www.registroimprese.it/*). In order to access the database on the website, in Italy, it costs around 3 EUR for a report and data researching companies retrieve these data for their analysis and audit.

Independent auditors will examine financial reports after publishing on behalf of investors or customers, they give a composed report containing their opinion about whether the financial statement is fairly stated and comply in all material respects.

In our view, we will follow the scenario to the work of Chamber Of Commerce, auditors can access our public database to investigate, even can refer our evaluation strategies.

B. Problem statement

We are focusing on the processes when XBRL files are created. Evaluating files is a complicated endeavor since it requires the validation of many steps by the local authority. In case the files do not pass the validation process, they need to be corrected, then the whole process needs to be executed again. A standard evaluation system is needed for both companies and governments.

Moreover, the traditional process is not transparent even when the XBRL files are published in a public database. The files can still be updated after years to cover mistakes from the accountants, thus, the database should not be controlled by companies and all changes need be traced.

III. BACKGROUND

The following section introduces the different technologies used in the definition of the proposed architecture.

A. XBRL

Financial reports contain sensitive data that might have a huge impact on the organization's future in terms of investments and collaborations, which mandates careful management and control mechanisms able to capture any inconsistencies or manipulation of the published reports. The first step towards this goal started with the introduction of the eXtensible Business Reporting Language [6], which is the world-leading standard for financial reporting. It facilitates inter-organization communication and enables automatic reports processing and analysis. XBRL relies on XML and XML based schema to define all its constructs. Its structure consists of two main parts:

1) XBRL instance, containing primarily the business facts being reported (see Figure 2).

```
<rp:RevenueTotal unitRef="EUR">5000</rp:RevenueTotal>
<rp:CostOfSales unitRef="EUR">3000</rp:CostOfSales>
<rp:GrossProfit unitRef="EUR">2000</rp:GrossProfit>
```

Figure 2. Facts example

Each fact has the following components:
Concept name: contains namespace prefix of taxonomy schema like rp:CostOfSales.
Id: defines the unique fact but its optional.
Context Reference: shows the context of fact like working year or location.
Unit refers to unit information of the fact value
Fact value: presents for the value of an asset

2) XBRL taxonomy, a collection of arcs, which define metadata about these facts and their relationship with other facts (see Figure 3 as example of calculation linkbase).

```
<loc xlink:type="locator"
  xlink:href="taxonomy#rp_RevenueTotal" />
<loc xlink:type="locator"
  xlink:href="taxonomy#rp_CostOfSales" />
<loc xlink:type="locator"
  xlink:href="taxonomy#rp_GrossProfit" />
<calculationArc xlink:type="arc"
  xlink:from="rp_GrossProfit" xlink:to="rp_RevenueTotal"
  weight="1" />
<calculationArc xlink:type="arc"
  xlink:from="rp_GrossProfit" xlink:to="rp_CostOfSales"
  weight="-1" />
```

Figure 3. XBRL Linkbase example

Totally, a taxonomy schema has five types of linkbase:
Label linkbase: provides human-readable strings for concepts, multiple languages are also supported here for each language.
Reference Linkbase: is intended to contain relationships between concepts and references to authoritative statements.
Calculation Linkbase: contains mathematical relationships among numeric items.
Definition Linkbase: associates concepts with other concepts using a variety of arc roles to express relations between concepts in taxonomies
Presentation Linkbase: contains hierarchical presentation relationships among concepts

Figure 4 depicts XBRL structure and the relations between the different components.

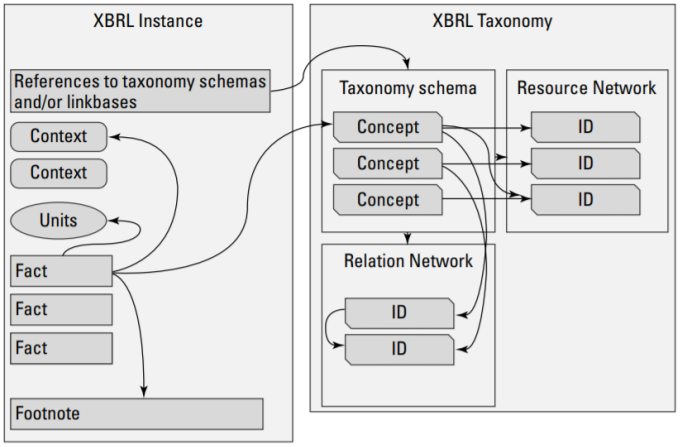


Figure 4. XBRL Structure

B. I-DLV

As the complexity of XBRL structure increases, it could reach a high number of definitions, which makes it impractical to check and validate manually. Thus, several tools have been developed to automate the validation process, Answer Set Programming (ASP) [7] is a form of declarative programming oriented towards difficult search problems, highly used in both academia and industry. ASP programs consist of rules by the form:

$$< head >: - < body > .$$

The rules are called facts, the symbol $: -$ means if, if the body is true, the head will exist. $< body >$ includes $b1 \vee b2 \vee .. \vee b3$, and $< head >$ are $h1 \wedge h2 \wedge \ldots \wedge h3$, so when all of b value is true, one of h will be chosen to do other computations.

The possible use of an ASP language for analyzing XBRL financial reports was explored by Gianfranco d'Atri in [8]. The tokenization and standardization of data supported by the XBRL Consortium allow extensive and meaningful use of AI techniques to support economic analysis and fraud detection.

I-DLV [9] is a new intelligent grounder of the logic-based Artificial Intelligence system DLV [10], it is an ASP instantiator that natively supports the ASP standard language. Beside ASP features, external computation in I-DLV is achieved by means of external atoms, whose extension is not defined by the semantics within the logic program, but rather is specified by means of externally defined Python programs, the so-called external atom in the rule bodies, which are also one of the most outstanding of I-DLV. Because of these features, in the paper, we applied DLV queries to analyze and absorb valuable knowledge from financial reports.

C. Blockchain technology

1) Distributed Ledge Technology: Distributed Ledger Technology (DLT) is an innovative data structure relying on a

decentralized and distributed peer-to-peer network to exchange data and a consensus protocol to keep data consistent. The main goal of DLT is to solve the double spending problem and provide a single source of truth with no trustworthy or centralized authority.

2) Blockchain: Blockchain [11] is a distributed ledger technology, built out of a linked list of boxes called blocks, linked together by hash codes. Each block references the previous block by including its has in its header. The blocks contain transactions that represent the information managed by the network (e.g., financial transactions, identity transactions ...). The Bitcoin blockchain is considered to be the first blockchain implementation, however, today, there more than a hundred implementation with different flavors and target different domains.

The main building blocks of a blockchain are [12]:

- Transactions, which are signed pieces of information created by the participating nodes in the network and then broadcast to the rest of the network. Every transaction must be verified by nodes before recorded into a public ledger, the verification node needs to ensure that the spenders own the crypto-currency via the digital signature, and has sufficient amount of currency in their account.
- Blocks, that are collections of transactions that are appended to the blockchain after being validated.
- A blockchain is a ledger of all the created blocks that make up the network.
- The blockchain relies on public keys to connect the different blocks (similar to a linked list).
- A consensus mechanism is used to decide, which blocks are added to the blockchain.

Generally, there are three types of blockchain platforms: public, consortium, and private [13]. In the public blockchain, all participants can execute and validate transactions. In consortium blockchain, the identity of the participants is known, but they do not necessarily trust each other. The network is moderated by one or more participants to keep access under control. Different participants might have different roles. In a private blockchain instead, the whole network is managed by one single organization. In our context, we apply public blockchain to publish financial reports to the public, where all participants could check business working status.

3) Ethereum: Ethereum [14] is a general purpose blockchain platform that enables the deployment of distributed applications. The main feature of Ethereum is the introduction of smart contracts, which are computer code residing in the blockchain and which gets executed once certain conditions are met. Smart contracts enable the development of decentralized autonomous organization (DAO), that uses smart contracts as functions to enforce governance mechanisms.

Since Ethereum is a public permissionless platform, it relies on mining to generate the next blocks of the chain. The transactions need to pay a fee in Gas [15], which is a unit that measures how much work of a node for an action or a task.

IV. System Architecture

The goal of the proposed architecture is to provide an end to end solution that leverages different technologies for managing financial reports and trustworthy publishing and updating.

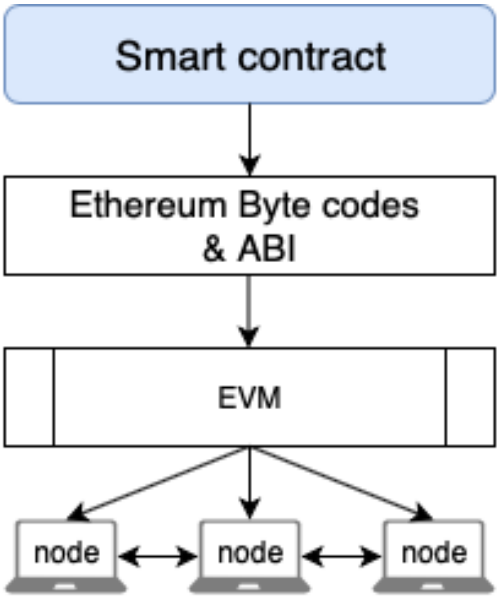


Figure 5. Smart contract execution

Figure 6 depicts an overview of the proposed architectures. It is divided into three main components: XBRL Reader, XBRL Evaluator, and XBRL Storage but in different approaches with blockchain integration that guarantees various levels of traceability.

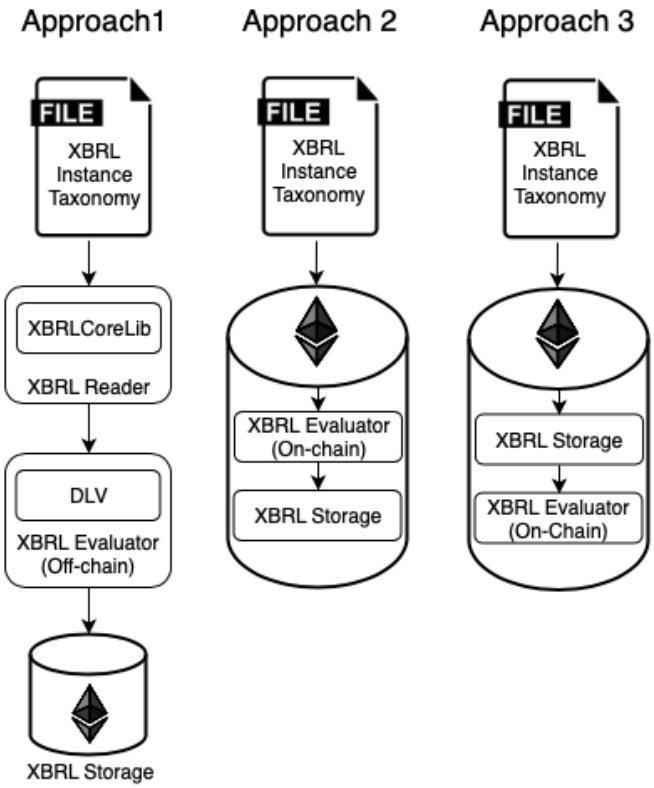


Figure 6. Alternative architectural designs

A. Components

We will start with each main component to understand its roles in approaches.

1) XBRL Reader: XBRLReader is responsible for validating the XBRL formatting by checking that all the schema is fully described. It takes as input an XBRL Instance that contains facts and a link to the taxonomies to be used.

The output of XBRLReader is a list of facts and arcs that are given to the XBRL Evaluator.

2) XBRL Evaluator: Facts and arcs from the first step are evaluated in the module, only in the first approach (see Figure 6 - Approach 1), the part is separated with DLV solver, in the others, the Evaluator is injected inside blockchain as a function. We define the needed aspects to investigate in a financial report here:

- Calculation consistency will check each value of facts, even if the value is aggregated from other asset's values like the example $GrossProfit = RevenueTotal - CostOfSales$, we will compare the result of $RevenueTotal - CostOfSales$ and $GrossProfit$ value with a threshold, the check applies for all the assets in the report, this kind of check also shows the errors inside reports where the difference between the actual value and the calculated value is greater than the threshold.
- The rate between interest and debt: a financial report normally shows data in 2 consecutive years, it could calculate changes of $interest/debt$ ratio during the years, if the index is too high, an alert is crucial for the company because it could be a potential sign for bankruptcy.
- Financial item comparison: From many reports in a year, we also compare financial item values among businesses to find, for example, the company has the highest revenue, or even filter companies do not pay for the warehouse cost.
- Benford's law checking: Benford's law [16] is an observation about the frequency distribution of leading digits in real-life data sets. The law states that a set of numbers is said to satisfy Benford's law if the leading first digit d $(d \in 1, .., 9)$ occurs with probability (see Figure 7):
  $P(d) = \log_{10}(d+1) - \log_{10}(d) = \log_{10}(\frac{d+1}{d})$
  The complicated formula is explained in [17] about stock prices example with distributions. Benford's law could check the whole data set or each financial report.

We note that the evaluation process can result in valid reports meaning that they satisfy all the pre-defined evaluation criteria or invalid reports that violate one or more requirements. At this point, it is up to the report owner to decide whether to publish the report or not. We also note that if invalid reports are published, they can be updated subsequently (e.g., add more information) to a valid state.

3) XBRL Storage: Storing financial data in a trusted location is a necessity to keep data safe and to be able to trace all the updates occurring over time. The main pieces of data of interest in our scenario are the financial facts and arcs. Blockchain is used as the backend storage where each fact and arc are stored in separate transactions. Once transactions are validated (i.e., added to the blockchain), the data becomes available to the users of the network who can view them, and any updates can be traced.

d	$P(d)$	Relative size of $P(d)$
1	30.1%	
2	17.6%	
3	12.5%	
4	9.7%	
5	7.9%	
6	6.7%	
7	5.8%	
8	5.1%	
9	4.6%	

Figure 7. Benford's law for the first digit [16]

B. Structural design

We propose an architecture that relies on two main perspectives i) off-chain for a standalone computation to analyze financial data ii) on-chain for enabling public execution in blockchain for possible functions. We consider three typologies (see Figure 6) demonstrating possible models with pros and cons for our financial data controller.

1) Approach 1: the financial reports are read by the XBRL reader, where all facts and arcs are extracted, then the Evaluator runs iDLV solver to analyze them and generate the results. The report owners can review the results before publishing them in the blockchain. Afterward, the owner can still update the inserted reports, and customers or the public can track the changes via blockchain logs or historical data.

- Advantages: the off-chain evaluator reduces validation time, especially when we are dealing with big data analysis with many financial reports at once.
- Disadvantages: The evaluation is executed as a black box and only the results are published. All the calculations are hidden, which might affect the level of trust from the public about the results.

2) Approach 2: facts and arcs also are results from the Reader, and directly they are evaluated with on-chain functions that are composed in a smart contract, after finishing the computation, if the owner is satisfied with the results, they can push it into blockchain. Any changes are accepted but still under the evaluation and can backtrack.

- Advantages: the Evaluator is implemented on the blockchain, therefore, everyone can verify its logic. To update the evaluation function, we only extend the current smart contract.
- Disadvantages: Sending a whole financial report (around 2MB) at once to a view function requires more time and it might cause the local node to crash

(e.g., the view function contains an infinite loop), which leads eventually to out_of_gas error.

Because of the negatives, approach 2 will not be implemented for testing.

3) Approach 3: this approach reverses the data storage and evaluation phases in comparison to approach 2. This allows data to be stored in advance without affecting the evaluation time.

- Advantages: data can be stored on-chain much prior to the evaluation process. Therefore, the evaluator can access the data without having to wait for it to load.
- Disadvantages: The computation runs on local nodes for evaluation that makes the performance becomes so slow for the whole financial report.

Generally, the positive and negative aspects of each approach are showed in Table I.

TABLE I. APPROACH QUALIFIES

Strategies	*Performance*	*Cost*	*Trust*	*Feasible ?*
Approach 1	high	high	low	yes
Approach 2	low	low	high	no
Approach 3	low	low	high	yes

C. Working use cases

To illustrate the interaction between the different components, we have defined a set of use cases addressed by the proposed architecture. All scenarios assume that the user has a company registered in the system, the user then chooses an XBRL file and the evaluator shows four possible outcomes. Fig. 8 depicts a sequence diagram that covers most of the scenarios.

- If all aspects are satisfied (valid), the user publishes the data into the blockchain.
- If one of the evaluation criteria is violated, the user is advised to review the report and submit it later.
- If one of the evaluation criteria is violated, the user can still publish it into the blockchain but it will be flagged as invalid.
- Invalid reports already in the blockchain can be updated by their owners (e.g., update report values). The evaluator will check them again, if the updated report is accepted, the flag will change to valid. We note that if valid reports are updated with incorrect values they will be also flagged invalid.
- Other users or any third party organizations could view and evaluate any reports.

V. IMPLEMENTATION

The implementation of the proposed approach is conducted using a three layer architecture. Each of the layers is detailed in the following subsections. The current implementation is a standalone application that interacts with the blockchain network. For the Etherum network, we rely on Blockchain network instance deployed at the University of Calabria, Italy called Unical coin [18] with the following configuration in Table II.

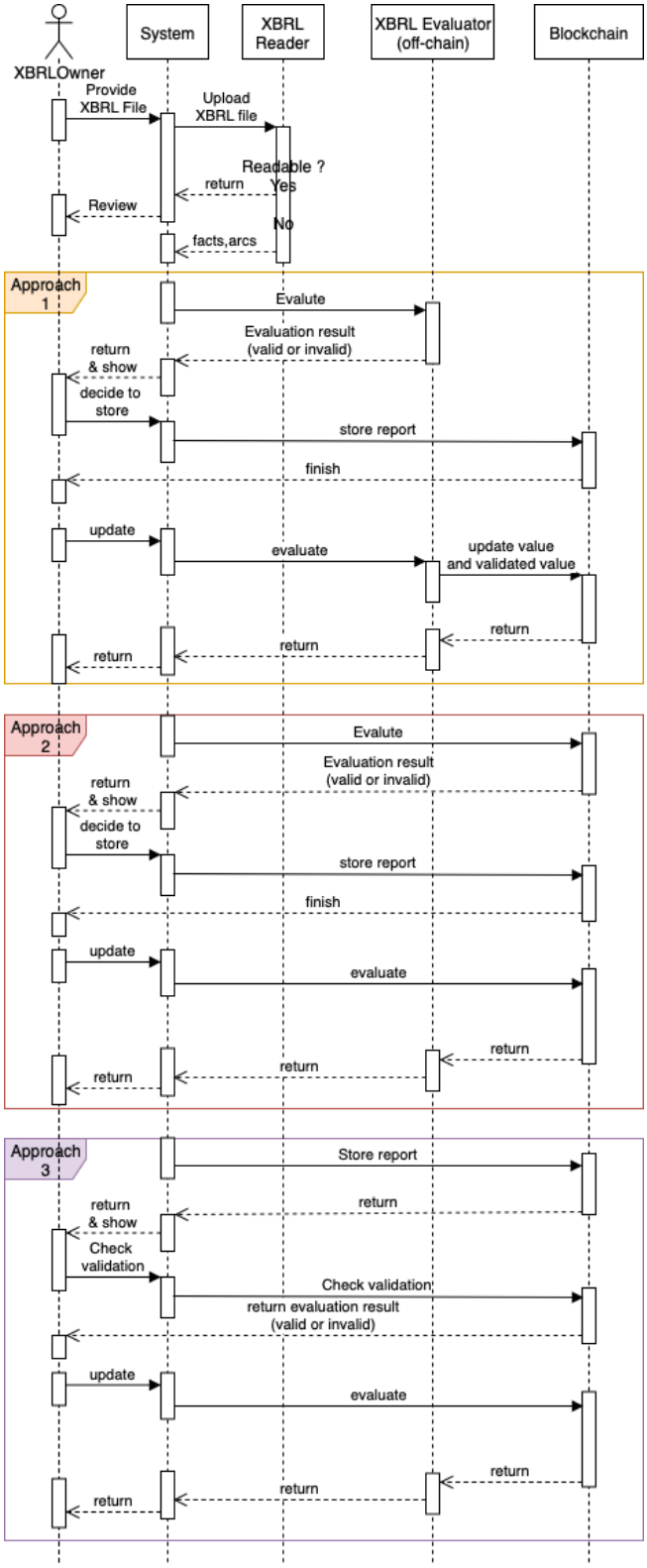


Figure 8. Report evaluation sequence diagram

The full implementation of the proposed approach can be found in our Github repository [19]. The two chosen approaches share XBRL Reader and XBRL Storage as a common part that we will describe in advance, only the evaluator for each solution are different from process and input.

TABLE II. NETWORK CONFIGURATION

Properties	*Value*
Original network	Ethereum
Difficulty	0x90000
Gas limit	0x2fefd8
Running nodes	4
Network speed	53 Mbps in download and upload

A. XBRL Reader

XBRL Reader uses XBRLCore [20], a library to read and extract data. It receives as input an XBRL file and extracts all the relevant information for the validation process, which include both XBRL instances and XBRL taxonomies (arcs) according to the XBRL 2.1 Specification. XBRLCore also has it own validation but it does not fit to the newest taxonomy (for example with group of item). For example, facts: $RevenueTotal$: $5000EUR$, and $CostOfSales$: $3000, GrossProfit$: 2000 and arcs: $GrossProfit = RevenueTotal - CostOfSales$, could be presented as Figure 9.

```
fact(revenueTotal, "5000", eur).
fact(costOfSales, "3000", eur).
fact(grossProfit, "2000", eur).
arc(grossProfit, costOfSales, "-1").
arc(grossProfit, revenueTotal, "1").
```

Figure 9. Facts and Arcs example

B. XBRL Storage

Financial data from the evaluator are published into blockchain via web3js and built smart contract. web3.js [21] is the Ethereum compatible JavaScript API, which implements the Generic JSON RPC specification, which is a collection of libraries, which allow you to interact with a local or remote Ethereum node, using an HTTP or IPC connection. Smart contract will make the skeleton to store data of a report, a company has many reports, each report has its own facts and arcs (see Figure 10).

Functions facilitate users to fill data into the structure (see Figure 11). As explained about the gas limitation above, adding each fact and arc one by one will reduce the burden on the network.

C. XBRL Evaluator

1) Approach 1: XBRL Evaluator stores facts and arcs together with the queries in a query file to examine indices in the reports, also report where there is the error by i-DLV by calling from Java Runtime:

```
idlv xbrlFile.dlv calculation.py
```

xbrlFile.dlv includes the list of facts and arcs, queries (see Figure 12), and calculation.py includes utility functions such as real numbers operations and list functions (see Figure 13). After running the command above, it prints $invalidDocument$ if the data is not correct otherwise it prints $validDocument$. The code computes the assets' values by *i)* choosing each fact and its relation (arc) *ii)* multiple weight with asset value of each arc, and *iii)* sum these values to get expected asset value to compare with the actual value from fact. If they are not equal, $checkFact$ returns $false$ and $isValidDocument$ is also $false$, in other words, the document is not valid,

```
struct Fact {
    string concept;
    string context;
    int value;
    string unit;
    string factgroup;
}
struct Arc {
    string conceptFrom;
    string conceptTo;
    int weight;
    string calLinkBase;

}
struct Report {
    string reportId;
    string date;
    string validated;
    Fact[] facts;
    Arc[] arcs;
}
struct Company {
    address companyAddress;
    string companyName;
    Report[] reports;
}
Company[] public companies;
```

Figure 10. Companies structure

```
function ownCompany();
function ownReport( reportId);
function registerNewCompany();
function getCompany();
function addReport(report);
function addFact(fact);
function addArc(arc);
function updateFact(fact);
function updateArc(arc);
```

Figure 11. Storage functions

otherwise, it is accepted. With queries, we can verify one or many documents at once with all defined metrics.

```
chooseArc(F1, F2, V) :- fact(F2,V2,U), arc(F1,F2,W),
    &times(F1,V2,W;V).
invalidFact(F) :- chooseArc(F, _, V), &checkFact(F,V
    ;"False").
invalidDocument :- &checkDocument(;"False").
validDocument :- &checkDocument(;"True").
```

Figure 12. Query example

2) Approach 3: XBRL Evaluator will integrate with the XBRL Storage as a view function in blockchain (see Figure 14). View functions do not cost any fee in execution but the amount of gas is still calculated and is still limited in one block, this is also a method of Ethereum to prevent infinite loop. To evaluate a financial report, we need to scan the fact and arc list with two for and four if that is cumbersome for a local node, therefore, we break one big function into two smaller function calls. $calActualFactValue$ finds related arcs and the check it with $checkArc$ before returning each chosen fact actual value, and then we can compare it with the real

```
listFacts = {};
isValidDocument = True

def times(F, X, Y):
    fx = float(X)
    fy = float(Y)
    if F not in listFacts :
        listFacts[F] = 0
    listFacts[F] += fx * fy
    return str(fx * fy)

def checkFact(F, X):
    fx = float(X)
    if fx == listFacts[F] :
        return True
    else :
        isValidDocument = False
        return False

def checkDocument() :
    return isValidDocument
```

Figure 13. Calulcation.py example

value in the dataset.

The result of the evaluation is not necessarily published into the blockchain since anyone can verify via their local nodes. Thus we can save more energy and Ether cost for the transaction of validated values.

```
function checkArc(
    uint companyIndex,
    uint reportIndex,
    uint findex,
    uint aindex) public view returns (
        int rs
    ) {
}
function calActualFactValue(
    uint companyIndex,
    uint reportIndex,
    uint findex) public view returns (
        int realvalue, int rs
    ) {
}
```

Figure 14. On chain evaluator functions

VI. Evaluation

A. System configuration

We demonstrate the testing environment:

- Processing core: Intel Core i7-4710HQ (2.50 GHz, 6 MB L3 Cache).
- Chipset: Mobile Intel HM86 Express Chipset.
- RAM: 8.00 GB.
- Operating system: Ubuntu 16.04.4 LTS
- Hard Disk Drive: Solid State Drive (SSD) 1 TB Toshiba
- Application: Java web project maven eclipse

Two important aspects to evaluate when considering blockchain based solutions are cost and performance, we also evaluate defined financial metrics to review the accuracy of our methods.

B. Cost evaluation

We tested our system using 200 valid XBRL files, 22 invalid files (valid in calculation consistency) provided by different business providers and are annual financial reports. The calculation is performed by the XBRL Evaluator, which implements the required mechanisms to check the soundness and completeness of the given files. The tests consider all the implemented functions of the smart contract. These tests have been run on a test blockchain network and can be reproduced by calling a set of REST endpoints. Endpoint returns the amount of gas consumed while executing transactions. The amount of gas used is multiplied by the gasPrice to obtain the costs in Ether. The Ethereum to Euro conversion factor to these prices allows computing the monetary cost. Table IV presents the cost of executing the various contract functions.

TABLE III. Costs of smart contract functions execution

Function	*Ether cost (GWei)*	*Euro cost (€)*	*Avg Time (ms)*
registerNewCompany	0,00032	0,059	7022
addFact	0,01	1,83	7579
addArc	0,01	1,83	7579
addReport	0,0012	0,22	11705
updateFact	0,01	1,83	7579
updateArc	0,01	1,83	7579
updateValidatedValue	0.0012	0,22	12325

We note that on average an annual report contains around 129 facts and 61 arcs (192 transactions) which would cost approximately 0.74 ETH (118.86 EUR at 28 August 2019 followed by [22]).

C. Time evaluation

In terms of time execution for XBRL Storage, we simulated the scenario used in our approaches, that is the process of publishing reports (addReport, addFact, addArc, updateValidatedValue). Figure 15 shows the average execution time for the whole process. The *x* axis represents the total number of facts and arcs as used in the process.

The results depicted show that the execution time for storage is linear relative to the number of transactions. However, other factors affect the execution time, mainly the variation of gas price, which affects what transactions will be picked by the miners first and the size of the network (i.e., how fast the transactions are broadcasted).

Considering the time execution in evaluation, each strategy performs differences as shown in Figure 16. The off-chain one uses iDLV solver so the time is much faster than the other, the maximum time evaluation in the case is only around 300 ms in contrast of on-chain calculator, that is more than 30000 ms (30 seconds) and the value will increase following the growth of a number of facts and arcs. In a real report with around 129 facts and 61 arcs (192 transactions), blockchain takes around 17031ms (17 seconds) while it is only 5516ms (5 seconds) in iDLV.

When executing transactions, we set the gas price as default at 20 gwei, but following [23], execution time could change

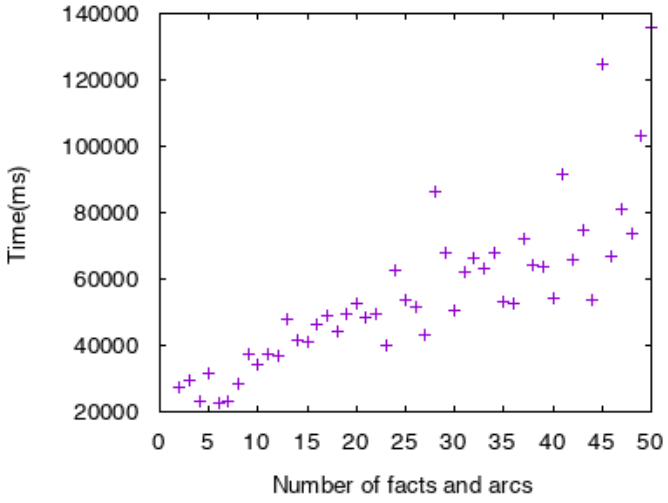


Figure 15. Storage execution time.

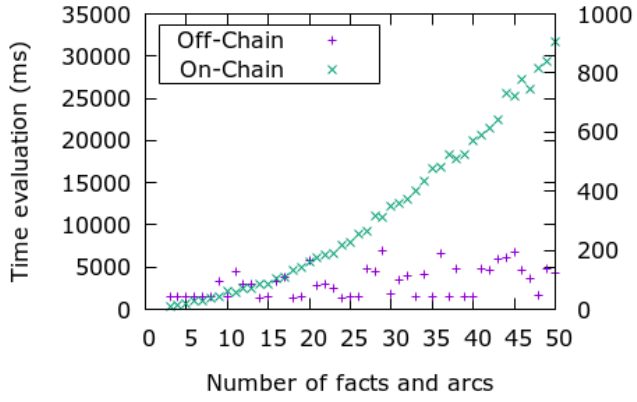


Figure 16. Evaluation time.

based on the gas, the suitable gas price is 43 gwei. We are using the private Ethereum chain so doing experiences with gas price changes are not feasible in the environment, and we will leave the question for future work when we can test the price in the real network.

D. Metric evaluation

As discussed about the four possible metrics we need to investigate, our dataset contains 120 financial reports in XBRL format, in the context of 2000 and 2015. Following the service provider, in these reports, there are:

- 100 valid files: the reports are accepted in any aspect to publish to a public database.
- 20 invalid files: the 10 reports do not meet the calculation consistency requirement and the other 10 reports miss important fields.

With the data set, we tested our algorithms that result in:

- Calculation consistency: the consistency is calculated by comparing the expected value with its actual value of all facts in reports. As the example above, the expected value is determined by:
$ExpectedValue = \sum subFactValue * arcWeight$
Since building financial report is complicated, that is cumbersome to correct all values, so we accept the fact if the difference between its expected and actual value is smaller than a threshold, we set a default threshold in our model is 0.5, and the difference is calculated by:
$valueDifference = \frac{expectedValue - actualValue}{actualValue}$

  With the configuration, we have 83 per 120 files are correct, that means the service provider decision and our outcomes are similar for each file. The other 37 files have differences in qualifying since our strategy does not check some special essential fields that need advice from experts, moreover, they also have a middle man to fix unaware mistake to make sure the reports are clean before publication.
- The rate of debt and interest: we compared the differences between the rates from two consecutive years and a threshold that is fixed by 0.5, and all of the reports have the higher values, thus the rate threshold needs to be concerned to re-defined.
- Financial aspect comparison: by retrieving all values from our reports, we have a big database, to be explicit, in the context of 2016, there are more than 4000 financial items that need to be reviewed. *itcc_ci_totalecreditiversosociversamentiancoradovuti* is the total receivables from shareholders for payments must be shown to highlight both the portion that has already been called by the company, and the portion still to be called, we choose the item as an example, and the result showed the item is missed in 88 files, is zero in 23 files and has values in only 14 files.
- Benford's law: we checked all first digits in reports, choose two big groups in each report as $conto\_economico$ and $stato\_patrimoniale$, and also review the law suitability in numbers of each report. The law suitable comparison is made by
$scarto = \sqrt{\frac{\sum_{i=1}^{9}(P_i - B_i)^2}{9}}$.
The $scarto$ will be calculated in Table IV following each first digits from 9 to 1, the negative and zero value are also countable, the table also presents the differences between the actual percentage in the dataset and the benford's law.

TABLE IV. Costs of smart contract functions execution.

Number	*Count*	*Percentage (P_i)*	*Benford's law(B_j)*	Differences
9	750	4.72	4.6	0.12
8	806	5.08	5.1	-0.02
7	873	5.5	5.8	-0.3
6	985	6.21	6.7	-0.49
5	1303	8.21	7.9	0.31
4	1469	9.25	9.7	-0.45
3	2001	12.61	12.5	0.11
2	2820	17.76	17.6	0.16
1	4867	30.66	30.1	0.56
Negative	1731			
Zero	4945			
			Scarto	0.33

For all financial documents, the $scarto$ value is 0.33 as Table IV, but for the two groups, the $scarto$ of $conto\_economico$ is 0.96 and $stato\_patrimoniale$ is 1.33 that are compared with the stardard law in Figure 17.
We qualified Benford's law in each report, the result is

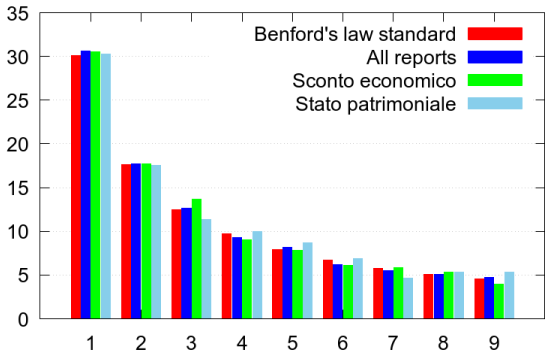


Figure 17. Compare first digits of groups with Benford's law

even worse when the maximum *scarto* for one report is 1193.66 and the minimum one is 45.64.
The result from the larger amount of data gets closer to the law, even with the whole dataset, it is only 0.33 in differences but get higher with smaller dataset, we can experience that the Benford's law could check the data of a company in case the financial report is exported every month, and the checking can run after several years.

VII. Related Work

Providing trustworthy financial data is a challenging endeavor. Over the years different tools have been developed to analyze the financial information generated by companies in order to check its consistency and integrity. However, since most of the proposed tools rely on third party organizations, issues related to trustworthiness and privacy still need to be solved.

Recently blockchain has found applications in different domains including IoT [24] [25], finance [26], health care [27], smart mobility [28] and others. In the literature, a number of studies considered the implication of blockchain on financial services and accounting. Byström [29] argues that blockchain can help corporate accounting in many ways, especially in terms of trustworthiness in accounting information and data availability in a timely manner. In [30], the authors discuss how blockchain can be an enabler technology for accounting ecosystem auditing and transparency. In [31], Colgren discusses the advantages that blockchain can bring to companies by allowing fast and public access to companies financial statements. In [26], Bussmann has given a more general overview of the potential disruption of blockchain on the Fintech market.

For banking services, Ye Guo [32] suggests that blockchain is able to replace the banking industry as external and internal issues like economic deceleration and increasing credit risk and non-performing assets. Thus, blockchain could synchronize and verify financial transactions to eliminate the problems of subsequent reconciliation. Q.K.Nguyen [33] indicated that blockchain could potentially reshape the economy, but the banking industry requires high speed of transaction execution with high scalability that is still a limitation of blockchain. VAT Fraud detection and prevention is a recommendation from [34] that uses access keys to get the authorization when purchasing cross border products. In [35], Reverse factoring and dynamic discounting are two approaches that could get benefits from blockchain, which becomes a financier to guarantee the future payment or holding the cash from buyers to optimize the working capital.

Applying blockchain as storage, Sven Helmer et al. built MongoDB database functions into Ethereum in [36], that separates the driver and database to reduce the cost transactions. The main goal of their approach is to keep all data on-chain. Shafagh et al. in [37] proposes a solution for blockchain auditable storage of IoT data by two layers of control plane and data plane, in the architecture, we can define access rights based on blockchain and control data from distributed off-chain storage. A searching function is applied with distributed hash table (DHT), nevertheless their strategy is not implemented yet and they also did not specify, which tools are used. Another approaches for public storage, [38] storj extends the function to enable people to rent other free storage and bandwidth, based on blockchain framework, it uses encryption, file sharing to store files on a peer-to-peer network. IPFS [39] stands for Interplanetary File System, it works on HTTP protocol and uses DHT to store the data following content-addressing technique that set a permanent link for any uploaded content, content-addressing allows us to verify the data too and any users in the network can access the content by its address. The system now is still unstable and has a lot of bugs, access content from other computers still take more time than traditional storage, we can expect IPFS future release with more stable versions.

We found few articles discussing about on-chain computation, Xu et al via [40] showed the comparison of on and off chain solutions for data storage, computation, even based on different blockchain models. Keeping data inside blockchain is more expensive but this is a one-time cost for permanent storage. Current blockchain technologies as Ethereum and Bitcoin are suitable with simple computation, Digital Asset Modeling Language (DAML) [41] is a new coming solution for complex computing, this is a programming language for financial institutions. DAML is designed to solve the issues of the agreement without revealing its content, the model is optimized for a private environment and the current version can interact with Hyperledger [42]. Our purpose is to make a public and trust network for the community, so DAML is not reasonable for our adaption.

Considering big data in blockchain, there are many promise use case presented in [43] about blockchain application for big data like IoT or personal data area, Karafiloski et al. showed current application as examples to extend for data analysis. L. Yue et al. in [44] proposed a big data blockchain model with several levels of data access and collection right, the smart contract in the model will automatically encapsulate predefined states and conversion rules, trigger executions. Mystiko [45] is quite comprehensive to build a whole blockchain network from scratch with up-to-date and high level technologies like cassandra for storage, kafka for communication. The high scalability, availability and elastic search enable it to be big data friendly. However, the project is implemented for a private chain that is suitable for multi party solutions, while we need a public chain for high level of trust and public verification, thus, applied strategies in Mystiko could be considered for our next extended version.

In terms of tools related to XBRL, several tools are in use, however, they are not able to guarantee the long term

trustworthiness of the reports on produced. With regard to the analysis of financial reports in XBRL format, Arelle [46] is an open source platform for XBRL financial reports format analysis. Users can view the structure of a document and use features with a GUI. Arelle provides many services that can be integrated with other technologies. Altova [47] is also well-known based on the XML development. With the help of Altova, users can present XBRL maps and relationships inside, including facts, context, and arcs. These tools have their own evaluation tools but just check with basic concepts even with some specific documents, so the result is not consistent. Moreover, considering the transparent characteristics of financial documents, we need a better approach that guarantees transparency of the whole validation process.

VIII. Conclusion and future work

In this paper, we have presented the design and a prototype implementation of a blockchain based financial reports ledger. The main goal of the proposed approaches is to increase trust and transparency in published financial reports, which can have a great impact on inter-organizational transactions. From the side of authorities, they do not have to wait for the long process of validation or storing reports, companies will submit and verify it via the system automatically, and for investors or any users want to check reports or changes in facts, they only follow logs to reveal differences.

Using ASP as a core computation in the first approach makes flexible and easy to maintain but get less trust since the public can not read clearly the full code after deployment, another strategy with on-chain computation get higher trust but take more time in evaluation (around 100 times in comparison with the off-chain version), the trade-off seems more acceptable with the public. Combination the two methods could be a solution in our future work when the companies can review and fix mistakes with a draft of report evaluation by ASP and the customers only need the on-chain function to check the validity without off-chain solver investigation.

Although the study is limited to the Italian context and does not provide a cross-analysis with other systems, the goal here is to shed some light on the great potential of using distributed ledger technologies in financial reports validation, storage, and traceability. The proposed approach has been applied to the niche area of financial reports, but the same approach may have much wider applications in numerous contexts.

For future work, we are investigating the automatic correction of invalid XBRL documents such as typing mistakes and facts missing value. Moreover, financial statements should be based on the cash flow statements from organization to organization. When we have all data flow, we can provide end to end trustworthiness and reliability.

Additionally, since we are currently working in the context of smart mobility infrastructure in the 5G CARMEN project [48], which focuses on crossing organizational boundaries use cases. Services such as cross border insurances and multimedia content access require trustworthy collaboration among different countries to generate and manage payments. This has a great effect on the cash flow and therefore, financial reports of international companies. Our work in this paper can contribute to trust management by providing a standard and trustworthy mechanism for financial transaction validation.

References

[1] G. D'Atri, V. T. Le, C. Pahl, and N. El Ioini, "Towards trustworthy financial reports using blockchain," in Proceedings of The Tenth International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2019), Venice, Italy, 2019, pp. 37–42, ISBN: 978-1-61208-703-0.

[2] Agenzia per l'Italia digitale, "XBRL - Standard formato elettronico editabile per la presentazione dei bilanci," 2019, URL: https://www.agid.gov.it/it/dati/formati-aperti/xbrl-standard-formato-elettronico-editabile [accessed: 2019-12-04].

[3] XBRL International, "Tools and Services," 2019, URL: https://www.xbrl.org/the-standard/how/tools-and-services/ [accessed: 2019-12-04].

[4] M. E. Phillips, T. E. Bahmanziari, and R. G. Colvard, "Six Steps to XBRL," 2008, URL: https://www.journalofaccountancy.com/issues/2008/feb/sixstepstoxbrl.html [accessed: 2019-12-04].

[5] Investopedia, "Chamber of Commerce," 2018, URL: https://www.investopedia.com/terms/c/chamber-of-commerce.asp [accessed: 2019-12-04].

[6] XBRL Organization, "An Introduction to XBRL," 2001, URL: https://www.xbrl.org/the-standard/what/an-introduction-to-xbrl/ [accessed: 2019-12-04].

[7] Wikipedia contributor, "Answer set programming," URL: https://en.wikipedia.org/w/index.php?title=Answer_set_programming&oldid=898338706 [accessed: 2019-12-04].

[8] G. D'Atri, "Logic-based consistency checking of xrbl instances," IJACT, vol. 3–6, 2014, pp. 126–131.

[9] J. Zangari, "idlv," 2018, URL: https://github.com/DeMaCS-UNICAL/I-DLV/wiki [accessed: 2019-12-04].

[10] W. T. Adrian, M. Alviano, F. Calimeri, B. Cuteri et al., "The asp system dlv: Advancements and applications," KI - Künstliche Intelligenz, vol. 32, no. 2, 2018, pp. 177–179. [Online]. Available: https://doi.org/10.1007/s13218-018-0533-0

[11] M. Crosby, P. Pattanayak, S. Verma, and V. Kalyanaraman, "Blockchain technology: Beyond bitcoin," Applied Innovation, vol. 2, no. 6-10, 2016, p. 71.

[12] C. Cachin, "Architecture of the hyperledger blockchain fabric," in Workshop on Distributed Cryptocurrencies and Consensus Ledgers, 2016.

[13] N. El Ioini and C. Pahl, "A review of distributed ledger technologies," in On the Move to Meaningful Internet Systems. OTM 2018 Conferences. Springer International Publishing, 2018, pp. 277–288, ISBN: 978-3-030-02671-4.

[14] Ethereum Foundation, "Ethereum," 2013, URL: https://www.ethereum.org/ [accessed: 2019-12-04].

[15] G. Wood, "Ethereum yellow paper," 2014, URL: https://github.com/ethereum/yellowpaper [accessed: 2019-12-04].

[16] Wikipedia contributors, "Benford's law — Wikipedia, The Free Encyclopedia," 2019, URL: https://en.wikipedia.org/w/index.php?title=Benford%27s_law&oldid=912469307 [accessed: 2019-12-04].

[17] L. Pietronero, E. Tosatti, V. Tosatti, and A. Vespignani, "Explaining the uneven distribution of numbers in nature: the laws of benford and zipf," Physica A: Statistical Mechanics and its Applications, vol. 293, 2001, pp. 297–304.

[18] Unical Coin Team, "Unicalcoin," 2017, URL: https://github.com/marcuzzu/UnicalCoin [accessed: 2019-12-04].

[19] V. T. Le, "Trustable system for XBRL," 2001, URL: https://github.com/levanthanh3005/TrustableSystem-for-XBRL [accessed: 2019-12-04].

[20] Y.seki, "XBRL Core," 2006, URL: https://sourceforge.net/projects/xbrlcore/ [accessed: 2019-12-04].

[21] Ethereum Foundation, "web3.js - Ethereum JavaScript API — web3.js 1.0.0 documentation," 2019, URL: https://web3js.readthedocs.io/en/v1.2.1/ [accessed: 2019-12-04].

[22] Softo ltd, "Currencio — Cryptocurrency Converter," 2001, URL: https://fcurrencio.co [accessed: 2019-12-04].

[23] GitBook, "ETH Gas Station API," 2019, URL: https://docs.ethgasstation.info [accessed: 2019-12-04].

[24] C. Pahl, N. El Ioini, S. Helmer, and B. Lee, "An architecture pattern for trusted orchestration in iot edge clouds," in Fog and Mobile Edge

Computing (FMEC), 2018 Third International Conference on. IEEE, 2018, pp. 63–70.

[25] C. Pahl, N. El Ioini, and S. Helmer, "A Decision Framework for Blockchain Platforms for IoT and Edge Computing," in Proceedings of the 3rd International Conference on Internet of Things, Big Data and Security, no. IoTBDS, 2018, pp. 105–113, DOI: 10.5220/0006688601050113.

[26] O. Bussmann, "The future of finance: Fintech, tech disruption, and orchestrating innovation," in Equity Markets in Transition. Springer, 2017, pp. 473–486.

[27] M. Mettler, "Blockchain technology in healthcare: The revolution starts here," in e-Health Networking, Applications and Services (Healthcom), 2016 IEEE 18th International Conference on. IEEE, 2016, pp. 1–3.

[28] V. T. Le, C. Pahl, and N. El Ioini, "Blockchain based service continuity in mobile edge computing," in The 6th IEEE International Conference on Internet of Things: Systems, Management and Security. IEEE, 2019.

[29] H. Byström, "Blockchains, real-time accounting and the future of credit risk modeling," Lund University, Department of Economics, 2016.

[30] J. Dai and M. A. Vasarhelyi, "Toward blockchain-based accounting and assurance," Journal of Information Systems, vol. 31, no. 3, 2017, pp. 5–21.

[31] T. D. Colgren, "Xbrl, blockchain, and new technologies," Strategic Finance, vol. 99, no. 7, 2018, pp. 62–63.

[32] Y. Guo and C. Liang, "Blockchain application and outlook in the banking industry," Financial Innovation, vol. 2, no. 1, 2016, p. 24, URL: https://doi.org/10.1186/s40854-016-0034-9 [accessed: 2019-11-11].

[33] Q. K. Nguyen, "Blockchain - a financial technology for future sustainable development," in 2016 3rd International Conference on Green Technology and Sustainable Development (GTSD), 2016, pp. 51–54, DOI: 10.1109/GTSD.2016.22.

[34] R. Ainsworth and A. Shact, "Blockchain (distributed ledger technology) solves vat fraud," SSRN Electronic Journal, 01 2016, DOI: 10.2139/ssrn.2853428.

[35] Y. Omran, M. Henke, R. Heines, and E. Hofmann, "Blockchain-driven supply chain finance: Towards a conceptual framework from a buyer perspective." IPSERA 2017 - Budapest - Balatonfüred., 04 2017.

[36] S. Helmer, M. Roggia, N. El Ioini, and C. Pahl, "Ethernitydb – integrating database functionality into a blockchain," in ADBIS, 09 2018, pp. 37–44, ISBN: 978-3-030-00062-2.

[37] H. Shafagh, L. Burkhalter, A. Hithnawi, and S. Duquennoy, "Towards blockchain-based auditable storage and sharing of iot data," in Proceedings of the 2017 on Cloud Computing Security Workshop. ACM, 2017, pp. 45–50.

[38] Storj Labs, "Storj : A Decentralized Cloud Storage Network Framework," Tech. Rep., 2018, URL: https://storj.io/storjv3.pdf [accessed: 2019-12-04].

[39] Protocol Labs, "IPFS Document," Tech. Rep., 2019, URL: https://docs.ipfs.io [accessed: 2019-12-04].

[40] X. Xu, I. Weber, M. Staples et al., "A taxonomy of blockchain-based systems for architecture design," in 2017 IEEE International Conference on Software Architecture (ICSA). IEEE, 2017, pp. 243–252, DOI: 10.1109/ICSA.2017.33.

[41] D. Asset, "Introduction of DAML," 2019, URL: https://docs.daml.com/daml/intro/0_Intro.html [accessed: 2019-12-04].

[42] T. L. Foundation, "Hyperledger," 2019, URL: https://hyperledger.github.io [accessed: 2019-12-04].

[43] E. Karafiloski and A. Mishev, "Blockchain solutions for big data challenges: A literature review," in IEEE EUROCON 2017 -17th International Conference on Smart Technologies, July 2017, pp. 763–768, DOI: 10.1109/EUROCON.2017.8011213.

[44] L. Yue, H. Junqin, Q. Shengzhi, and W. Ruijin, "Big data model of security sharing based on blockchain," in 2017 3rd International Conference on Big Data Computing and Communications (BIGCOM), Aug 2017, pp. 117–121, DOI: 10.1109/BIGCOM.2017.31.

[45] E. Bandara, W. K. Ng, K. De Zoysa, N. Fernando, S. Tharaka, P. Maurakirinathan, and N. Jayasuriya, "Mystiko—blockchain meets big data," in 2018 IEEE International Conference on Big Data (Big Data), Dec 2018, pp. 3024–3032, DOI: 10.1109/BigData.2018.8622341.

[46] H. Fischer and D. Mueller, "Open source & xbrl: the arelle® project," in 2011 Kansas University XBRL Conference, 2011, pp. 29–30.

[47] Altova, "XBRL Development Tools," 2018, URL: https://www.altova.com/xbrl-tools [accessed: 2019-12-04].

[48] 5G-Carmen, "5G for Connected and Automated Road Mobility in the European UnioN," 2018, URL: www.5gcarmen.eu [accessed: 2019-12-04].

www.iariajournals.org

International Journal On Advances in Intelligent Systems
issn: 1942-2679

International Journal On Advances in Internet Technology
issn: 1942-2652

International Journal On Advances in Life Sciences
issn: 1942-2660

International Journal On Advances in Networks and Services
issn: 1942-2644

International Journal On Advances in Security
issn: 1942-2636

International Journal On Advances in Software
issn: 1942-2628

International Journal On Advances in Systems and Measurements
issn: 1942-261x

International Journal On Advances in Telecommunications
issn: 1942-2601