electronics

MDPI

Article

Experimental Cyber Attack Detection Framework

Cătălin Mironeanu *, Alexandru Archip, Cristian-Mihai Amarandei and Mitică Craus

Department of Computer Science and Engineering, Faculty of Automatic Control and Computer Engineering, "Gheorghe Asachi" Technical University of Iași, 700050 Iași, Romania; alexandru.archip@academic.tuiasi.ro (A.A.); cristian-mihai.amarandei@academic.tuiasi.ro (C.-M.A.); mitica.craus@academic.tuiasi.ro (M.C.)

* Correspondence: catalin.mironeanu@academic.tuiasi.ro

Abstract: Digital security plays an ever-increasing, crucial role in today's information-based society. The variety of threats and attack patterns has dramatically increased with the advent of digital transformation in our lives. Researchers in both public and private sectors have tried to identify new means to counteract these threats, seeking out-of-the-box ideas and novel approaches. Amongst these, data analytics and artificial intelligence/machine learning tools seem to gain new ground in digital defence. However, such instruments are used mainly offline with the purpose of auditing existing IDS/IDPS solutions. We submit a novel concept for integrating machine learning and analytical tools into a live intrusion detection and prevention solution. This approach is named the Experimental Cyber Attack Detection Framework (ECAD). The purpose of this framework is to facilitate research of on-the-fly security applications. By integrating offline results in real-time traffic analysis, we could determine the type of network access as a legitimate or attack pattern, and discard/drop the latter. The results are promising and show the benefits of such a tool in the early prevention stages of both known and unknown cyber-attack patterns.

Keywords: cybersecurity; intrusion detection; analytical framework; information security; data security

check for updates

Citation: Mironeanu, C.; Archip, A.; Amarandei, C.-M.; Craus, M. Experimental Cyber Attack Detection Framework. *Electronics* **2021**, *10*, 1682. https://doi.org/10.3390/electronics10141682

Academic Editor: Younho Lee

Received: 18 June 2021
Accepted: 12 July 2021
Published: 14 July 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Classic security Intrusion Detection Systems/Intrusion Detection and Prevention Systems (IDS/IDPS) tend to become obsolete on their own. The wide variety of both direct and indirect attack patterns makes it next to impossible to ensure the security of all digital data and computer systems. One may employ state-of-the-art firewall and antivirus solutions to enforce security policies, but still be vulnerable to zero-day attacks that can pass through. It is no longer enough to have security check-points in the infrastructure, no matter how reliable they supposedly are. Organizations also need to make sure that their security solutions are consistently working and capable enough to detect and prevent any potential attack. Behavior-based detection of the unknown is the main challenge in cybersecurity nowadays. Galal et al. in [1] proposed a behavior-based feature model to describe the malicious actions exhibited by malware at runtime. Their approach is based on the known fact that malware variants do indeed imply code change techniques such as polymorphism and metamorphism, resulting in different syntactic structures that share similar behaviors. Only two years later, a single spam campaign, that led to a GandCrab v2.1 sample being delivered as a payload, revealed hundreds of different hashes in an attempt to evade specific file signatures. There are many such cases, but this singular example alone is enough to state that signature-based detection is no longer enough. Cybersecurity research is moving towards behavior-based detection and attempts to model threats according to an access pattern continuum scale (Table 1—adapted from the Advanced Email Security Solution webinar by the Fortinet NSE Institute). There is a no-man's-land that implies huge technological effort between the clearly defined extremes of the scale: the "Known Good" and the "Known Bad". It is the zone that is targeted by our approach. Specifically, "Probably Good" and "Might be Good" is a theoretical situation because it allows the traffic

to pass. "Completely Unknown" is a dedicated problem for sandboxing and malware analysts. "Somewhat Suspicious" and "Very Suspicious" is usually covered by security solutions providers and is partially mirrored by "Might be Good" and "Probably Good", respectively. These borders are very fragile because a zero-day detonated in a sandbox that enacts some illegitimate actions will be labelled as "Suspicious", or more directly, "Known Bad". The time-frame required to reach such a decision ranges from a few minutes to a couple of hours. This sandboxing technique involves a continuous process and, as such, just the simple usage of powerful security tools is simply not enough anymore. Thus, Bruce Schneier's well-known statement that "security is a process, not a product" [2] is more meaningful than ever before.

Table 1. Access pattern continuum scale.

Access Pattern Continuum	Security Technologies
Known Good	Domain Safelists User Safelists
Probably Good	Historical Traffic Analysis
Might be Good	Source/Sender Reputation
Completely Unknown	Sandbox behavioral Analysis Content Disarm and Reconstruction
Somewhat Suspicious	Heuristics Analysis
Very Suspicious	Greyware Scanning
Known Bad	Antivirus

Modern approaches to security architectures implement this idea of security as a process, ensuring both adequate/proportional responses to various threats and increased visibility of the whole system and of the events that occur within such a system. Dedicated tools and software agents are employed to monitor and audit both the security solutions and the underlying physical and virtual systems. Inbound/outbound traffic, behavior of IDS/IDPS, security events, server applications' requests and responses, and so on become log data that undergo thorough analyses. The purpose is to determine whether or not the chosen security tools performed well. The results provided by these analytical tasks are in turn used to validate security configuration and enforce any eventually identified weak spots of IDS/IDPS and of the target system. This process follows through almost indefinitely.

Two major ideas spawned from the security is a process concept. The first approach focuses on understanding the huge amount of gathered log data by using data-mining (DM), big data and AI/machine learning (ML) techniques, and will be further discussed in Section 2. The second pathway focuses on modelling the attack patterns and attempts to provide abstract descriptions for these attacks. Following through with this idea, the Lockheed Martin Corporation developed the Cyber Kill Chain Framework in 2011 [3]—an IT reworking of the military Find Fix Track Target Engage Assess (F2T2EA) term. The purpose of this theoretical framework is to enhance visibility into an attack and enrich an analyst's understanding of an adversary's tactics, techniques, and procedures. The model identifies reconnaissance, weaponisation, delivery, exploitation, installation, command and control, and actions on the objective to describe cyber attack processes. The authors also refer to the Department of Defense Information operations doctrine to depict a course-of-action matrix using actions of "detect, deny, disrupt, degrade, deceive and destroy" to identify the tools and means that can be used to mitigate the phases of a cyber attack [3]. A more technical approach is the MITRE ATT&CK® globally accessible knowledge base of adversary tactics and techniques that has been built on real-world

observations, initially developed in 2013 [4]. A comparison between these two techniques for modelling cybersecurity attacks and defence was made in [5].

The purpose of this paper is to present a novel approach to robust security solutions. This approach is named the Experimental Cyber Attack Detection Framework (ECAD) and is developed with the purpose to facilitate research of on-the-fly/online security applications. The vast majority of solutions for security systems focuses on offline modelling and data analytics tools. We consider that these offline results could be integrated in on-the-fly/online approaches. Suspicious traffic would be detected and any potential attacks would be prevented during the early reconnaissance stage. Thus, ECAD has the benefit of strengthening existing IDS/IDPS instruments and providing better responses to security threats.

In particular, we bring the following contributions:

- We design and implement a cyber attack detection framework that can employ multiple reconnaissance agents. These agents provide useful low-level information about the network traffic and system logs. The framework then employs a specialised module that collects these data and decides whether to allow or drop a potentially suspicious connection.
- We validate the effectiveness of the framework by testing it against various scenarios related to the reconnaissance phase of the Cyber Kill Chain Framework.

The remainder of the paper is structured as follows. Section 2 presents the literature overview for the field addressed by our solution. The next two sections share our approach: the architecture considered for ECAD (Section 3) and the practical approach to its implementation (Section 4). The initial results and discussion are given in Sections 5 and 6, respectively. The final Section 7 draws the much-needed conclusions and hints on some prospective development.

2. Related Work

The escalating number of cyber attacks and related security events, and the increase and variety of attack patterns are met with renewed research efforts invested in developing new, better, and more robust IDS/IDPS solutions. These next-generation protection and prevention systems are being developed based on results supplied by data analytics tools and are ideally meant to prevent attackers from causing any effects on their targets. By the end of 2003, Taipale summarizes in [6] a first set of such attempts and focuses on how data-mining tools might or might not be appropriate for security purposes. The study identifies two equally important trends. Arguably, inconsistent, erroneous data could lead to misleading conclusions. Moreover, data-mining tools are usually statistical methods and could yield false-positive results. This could lead, for instance, to an improper classification of network traffic and could allow a potential attacker to pass through unhindered. On the other hand, one should take into account the fact that data-mining algorithms and tools are not synonyms for the whole knowledge-discovery process. A good understanding of the data, proper modelling and selection of input data, and adequate interpretation of the results are also key stages that require appropriate consideration.

In 2011, Liu [7] outlined two distinct approaches for data-mining/knowledge-discovery-based security solutions: top-down and bottom-up. This perspective is based on whether or not a hypothesis exists with respect to the analyzed security event. A top-down model implies that such a hypothesis exists and attempts to validate it through data-mining. On the other hand, the bottom-up method first attempts to analyze the security data and determine patterns and then formulate a model/hypothesis that encompasses the induced behavior. Liu then focuses on identifying the actual data-mining tools that are most commonly used [7]. Both unsupervised and supervised learning techniques are employed to either determine groups of similar access patterns from historical data, or attempt to categorise newly acquired data into existing classes. Association rule-mining is also used alongside clustering and classification to identify potentially interesting patterns and relations between security audit data.

Most analytical approaches are based on classic DM tools and on using these results for improving IDS/IDPS solutions. The idea in itself is not something new, but it still raises significant challenges. Lee and Stolfo presented one such possible concept as early as 1998 in [8]. Their focus is on audit data, such as access logs, protocol-specific data (such as FTP and SMTP), or dedicated monitoring commands (such as tcpdump) to build classifiers or to mine association rules. The purpose of these tasks is to contribute to building a set of decision rules for a potentially automated agent-based IDS. Enhancing their research, a subsequent study shows how modified association rule-mining and frequent episodes can be improved and therefore strengthen the aforementioned IDS [9]. Data subject to this research are off-line data gathered from network and system audit data. The achieved results are the base building-blocks in determining features of intrusion patterns, features that are then used for intrusion detection models. In [10], Lee summarizes both the progress and the issues that also arise from using DM and incipient forms of ML in IDS/IDPS: efficiency and credibility. DM/ML are mostly statistical instruments that could yield false-positives and false-negatives, as previously mentioned.

A more recent survey was presented by Jin et al. in [11]. The authors identify that both descriptive and predictive DM techniques are employed in security analysis. The paper includes a comprehensive comparison between approaches—12 solutions are carefully considered—and identifies the strengths and drawbacks related to different types of attacks. The comparison criteria are based on the detection efficiency, the detection accuracy over four types of attacks, the false-positive rate for non-attack data, and also on the characteristics of the DM method(s). The authors of [11] start their analysis with solutions destined to counteract Distributed Denial-of-Service (DDoS) attacks. According to the review, classification based on one-class SVM modelling achieves good results in identifying TCP-SYN, ICMP, and UDP flooding while having a potential disadvantage with respect to the dimensionality of data. This can be addressed by a PCA preprocessing stage. An interesting result is that this classifier achieves 98% accuracy in successfully detecting DDoS attacks and 100% accuracy in identifying normal traffic patterns [11]. This suggests that rather than focusing on identifying attack patterns, one could focus on normal, clean ones ("Known Good", as depicted in Table 1) and guide the security decision based on these clearly known facts. While the actual practicality of such an approach might be questionable (i.e., how good the actual classifier is), we consider it could still be investigated further.

The next types of attacks considered by Jin et al. are remote-to-local (R2L) and user-to-root (U2R) [11]. These types of intrusions are better addressed by a combination of two classic descriptive DM techniques: clustering (K-Means Clustering, for instance) and association rule-mining (Apriori/FP-Growth and as such). Yet again, the dimensionality of data could impose on the timeliness of the results. This could also be solved by relying on Big Data concepts (Map-Reduce/stream-processing or as such), especially when real-time security solutions are in focus. It is a well-known fact that traditional classification and clustering algorithms rely on the premise of "absolute definitions": a data object either belongs to a class/cluster, or does not. This may not necessarily be an advantage when applying these methods over security data. The heterogeneity and variety of data sources and the high degree of correlations that need to be performed might not be most suitable [11], especially in on-the-fly or real-time scenarios dealing with highly complex attack patterns. Jin et al. also analyzed the potential benefits of fuzzy-based approaches and sequence pattern mining applied over security-related data.

Jin et al. concluded their survey [11] with two more important remarks that reinforced Lee's results [10] almost 20 years later. The first such note is that one must pay a great deal of attention to the actual preprocessing stages of data selection and data modelling when dealing with DM tools and algorithms. These stages definitely impact on the quality of the results and on the success of the solution. The second important idea is that a single analysis technique is usually insufficient in providing trustworthy information. Both descriptive and predictive solutions must be considered and mixed before actually

reaching a security decision. We have taken this into account in the early design stages of the ECAD framework. Details are provided in Sections 3 and 4. Our approach relies on (a) performing multiple analyses over the same data in parallel and (b) having a voting agent that makes use of these analytical results in supplying a security decision.

A novel approach, based on ML tools, is presented by Bartos et al. in [12]. The authors propose a solution, entitled the "Network Entity Reputation Database System" (NERDS), focused on estimating the probability of future attacks. The concept in itself is rather intriguing. Instead of focusing on identifying attacks and attack patterns, Bartos et al. focused on the network entities that interact with the monitored system(s). Network entities are assigned a so-called "Future Misbehaviour Probability" (FMP) score based on meta-data about previously reported alerts related to the entities and on auxiliary, relevant third-party security information on those respective entities. The purpose is to estimate whether or not an entity would behave maliciously within an upcoming time-frame which the authors call the prediction window. The FMP score is determined using supervised machine learning techniques based on the aforementioned characteristics included in a past time-frame called the history window [12]. The presented test results highlight the potential benefits of NERDS, especially when compared against deny-list searching for identifying potentially malicious entities. Moreover, Bartos et al. emphasized the generic character of the FMP score (i.e., it may be used to predict the overall behavior of an entity or it may model only particular behavior, such as the probability of a scan or a DDoS or an access attack) and on including NERDS within collaborative IDS [12].

Research efforts are also invested in both the pre- and post-processing stages of DM/ML-based security approaches. Ahsan et al. focused on enhancing ML techniques used in detecting malware from packet information to strengthen cybersecurity [13]. Their research is directed at reducing the dimensionality of data while still maintaining adequate levels of precision. The authors consider this can be achieved through eliminating highly correlated features. Dynamic Feature Selectors (DFS) based on univariate/correlated feature elimination, gradient-boosting, information gain, and the wrapper method application are presented and are then tested as preprocessing stages for five ML methods. Achieved results support the DFS approach with overall increases in both accuracy and precision. It is worth mentioning that Ahsan et al. conducted their research on NSL-KDD and UNSW-NB15 data sets [13]. Kotenko et al. analyzed how combining multiple AI/ML techniques may improve the detection of distributed cyber attacks [14]. They focus on weighted ensembles of classifiers and compare decision support solutions based on three voting schemas: weighted voting, soft voting, and AdaBoost. Kotenko et al. developed their own architecture for a distributed IDS [14]. Data collected through different sets of analyzer entities are pushed to different classifiers. Each such classifier supplies its results to weighted ensembles, and thus a decision is made with respect to the nature of the access: normal/benign, DDoS or port-scanning. Tests are conducted against the CICIDS2017 offline data set and show the benefits of voting-based approaches in combining different ML results.

Hofer–Schmitz et al. [15] has presented another, more recent study that emphasizes the importance of the data preprocessing and data modelling in attack detection. The authors target the Advanced Persistent Threat (APT) scenario. They focus on the later stages of APTs and on the whole attack in itself, rather than only on the initial intrusion phase. Supervised ML techniques are analyzed based on the feature set that constitutes the input data. Hofer–Schmitz et al. clearly prove that dimensionality reduction is a must—there are a great deal of features that yield repetitive, less, or no significant data, alongside the much-needed improvement of the response time. Another key note is that the actual datasets could be lacking with respect to the features identifying APTs. Hofer–Schmitz et al. solved this issue through a combination of facts: normal/benign data (the authors refer to this as background data) and some known knowledge on APTs. Thus, attack-identifying data are randomly inserted into the background, such that only one

attack pattern would be observable during a one-hour interval [15]. The results they obtain underline the importance of adequate feature set selection on different datasets.

The previously presented examples clearly show that DM/ML are useful in improving existing IDS/IDPS solutions, especially when combined into more comprehensive approaches. The subject addressed by our paper is focused on efficient architectures and practical implementations that would combine and use such analytical results in a real-time response scenario. Thuraisingham et al. [16] established the challenges and direction from the perspective of data-driven approaches to cyber security as a science. Their main reasoning premise is that IDPSs are nowadays useless against APTs and slow attack vectors. The aforementioned Hofer–Schmitz et al. [15] actually prove Thuraisingham reasoning a few years later. Thuraisingham et al. also establish one possible direction in developing a data-driven framework using a context model and related context representation for knowledge-sharing. This is based on the remark that "no single component will have complete knowledge about its context" [16]. This approach was one of the mottoes in our design of the ECAD framework. Another key concept is that the logic of context-reasoning must be implemented separately from the monitored system and from the actual behavior of its underlying IDS/IDPS components. Our approach, ECAD, closely relates to this concept and allows the dynamic modification and/or replacement of these reasoning modules. Attacks such as APTs could thus be detected through forensic approaches—using rules over the sensed knowledge and data that describe a potential attack—in (near) real-time. Our experiments are focused on streaming data and facts and reasoning based on what is available for the currently inquired suspicious action. Thuraisingham et al. consider that, at least from a theoretical perspective, such investigations could only be performed post-factum. There was a real challenge to solve this issue: in the context of streaming, how can there be established a set of rules while the underlying facts that feed these rules require contextual analysis induced by those same rules? This cyclic dependency led us to design the ECAD framework as a state machine with a feature of storing historical metadata for later analyses. From the state machine perspective, we use security states that affect context and assume that a certain latency is therefore acceptable for this particular case. Establishing these particular contexts was another challenge that we needed to manage. This led us to the idea of using stream-based classification in detecting attacks. We study the problems of "concept-drift" and "concept-evolution" first stated in [17]. There are real problems in a context of evolution of attacker methods or evolution of benign software usage patterns, that makes previously trained models outdated and therefore necessitates the continuous refinement of the classification model in response to new incoming data. It is clear for us that the ECAD framework must also deal with the emergence of novel classes over time. This is due to the implied fact that, for instance, new APTs would update and enhance existing attack methodologies and/or be based on entirely new ones. Given this perspective, we do not consider that the ECAD framework is a one-time deal, and we have designed it to be highly adaptable. This is detailed in Sections 3 and 4. We therefore include an analytical overview of some of the data-driven security frameworks for the remainder of this current section.

Ankang Ju et al. [18] proposed an architecture related to timely detection of comprehensive intrusions–HeteMSD (Heterogeneous Multisource Data). They differentiate between traditional attacks and targeted cyber attacks based on the complexity of the attack, on the motivation of the potential bad actor, and on the specificity of the attacked target. Based on this note, Ju et al. argued that the Kill Chain model is appropriate only for targeted cyber attacks. We consider that this is an artificial difference because complex attacks mingle both traditional and targeted approaches and make use of a high variety of techniques in exploiting a wide range of vulnerabilities. Moreover, we believe that one of the main goals of a potential attacker is to pass through unhindered. This is usually achieved by hiding malicious intentions within normal data or traffic. If we consider the case of remote system access, then a potential attacker would first scan the target to identify open network ports. One could use the Nmap tool which offers fine-grained

timing controls alongside six predefined timing templates for IDS evasion. Some of these are based on arbitrary time-frames between consecutive scans such that it would appear as a common traffic error. This implies that a potentially targeted cyber attack relies on a first reconnaissance phase similar to a traditional attack. Ju et al. also consider that vulnerability data and threat intelligence could prove to be a valuable information source for any potential security solutions [18]. We do take this into account in our approach, and we attempt to integrate these data in our practical implementation. On a last note on [18], it is not clearly specified whether or not their model is only theoretical and whether or not they rely on offline or online data. At this point, we set our solution aside by directly aiming at online data as the target of our analysis, having offline knowledge as the basic building blocks for our decision system.

A closely related idea to the first two layers of our architecture was presented in [14]. Their challenge was to develop the architecture of a distributed system capable of detecting cyber attacks based on Big Data technologies. One of their concerns was to reduce the number of lost packets, thus reducing the likelihood of losing information about a possible attack on high-speed computer networks. They are applying the method of balancing at client level, for distributing the load of network packets leaving the client-sensors and entering as input for packet classifiers at the server collector level, as a solution to reduce the number of dropped packets. Their basic architectural idea consists of several client-sensors and one server-collector. Each sensor includes one balancer and several network analyzers. At their estimated transfer rate at 600 kilopackets per second, even legit traffic may have DDoS behavior when full packet capture is performed. In our early experiments with the implementation of the ECAD framework, we extract only the traffic metadata because we investigate the behavior and not the payload, which is usually encrypted. However, this traffic behavior might still occur, and for this reason, we imply the usage of load balancers in order to scale out on demand, as described in Section 4.2.1.

The "Universal Cyber Security Toolkit" (UCST) is intended to enhance the cyber protection of the healthcare IT ecosystem and ensures the patients' data privacy and integrity. UCST is developed as part of the SPHINX project under the grant agreement H2020-EU.3.1.5.1-ID 826183. Authors have developed the architecture and its corresponding technical specifications focusing on the proactive assessment and mitigation of cyber security vulnerabilities and threats [19]. The authors determine, through a Vulnerability-Assessment-as-a-Service, a so-called "Common Vulnerability Score" that is used to dynamically assess vulnerabilities of network entities. UCST, as the core component of SPHINX, is meant to operate with online traffic data, and the initial tests have been performed over the NSL-KDD Train 20 variant intrusion detection data set. SPHINX is therefore another solution that attempts to integrate ML analytics into an online approach. However, it seems that this approach does not make use of multiple analytical tools and does not combine their respective results [20]. What sets our solution aside is that we do combine results from different sources (threat intelligence, DM/ML solutions, etc.) in an attempt to: (a) achieve good response/reaction times; (b) increase the degree of confidence in a decision; and (c) provide the means to further analyze these outcomes and better understand the results. With this purpose in focus, we have included a voting agent within ECAD and support for persistent decision storage (see Section 3). Similar vote-based decision solutions in security systems are therefore further reviewed.

We have previously mentioned that the model submitted by Kotenko et al. [14] also employs a voting approach due to the inherent errors of DM/ML tools. The authors suggest a weighted voting mechanism, having the actual coefficients assigned to the basic classifiers with respect to the correctness over the training instances. They also make use of tools that keep track of the results supplied by each individual classifier. These historical data are used both to adjust the assigned weight and to improve these classifiers. Improvement is achieved by retraining faulty modules focusing on the previously incorrectly classified objects. Babić et al. [21] also emphasized the fact that a voting mechanism should be used alongside DM/ML modules. They rely on a Triple Modular Redundancy (TMR) approach

and focus on identifying DDoS attacks. The input data for the TMR module are collected from three classifiers (EWMA, CUSUM, and KNN), where the result is binary and indicates whether an attack is happening or not. Babić et al. [21] also stated that TMR may be generalised into a n-modular redundancy (n being odd) provided that multiple analytical tools are available.

As a research and educational framework, our approach allows the opportunity to share the evidence on which the decision was based on an integrated information-sharing platform, for later studies. New knowledge gained on both existing and new attack patterns are invaluable in strengthening IDS/IDPS solutions and successfully reducing the potential impact of security breaches.

Wagner et al. presented in [22] a malware information-sharing platform and threat-sharing project platform. Their purpose is to collect and share the important Indicator of Compromise (IoC) of targeted attacks as an open-source software solution. It is worth noting that this approach is mainly developed by the Belgian Defense CERT and the NATO Computer Incident-Response Capability (NCIRC). The gathered data include both technical and non-technical information about malware and attacks and is stored as a so-called event using a standardized format. Sharing and distribution is intended for trusted parties established in a trusted environment. The MISP—Open-Source Threat Intelligence Platform and Open Standards For Threat Information-Sharing (formerly known as the Malware Information-Sharing Platform)—provides functionalities to support the exchange and consumption of information by NIDS, LIDS, log analysis tools, and SIEMs. One of the most interesting features of MISP is the correlation engine. Its purpose is to identify all relationships between events based on their attributes. This is performed through one-to-one, fuzzy hashing correlation techniques and/or Classless Inter-Domain Routing block-matching. We consider this to be a very useful approach, since it could allow security specialists to preemptively address potential threats. ECAD had therefore been designed to embed both this validated threat intelligence data source and its corresponding correlation functions.

There are many open-source projects and commercial products that offer the means to deal with threat intelligence. Solutions like IBM X-Force Exchange, Alienvault OTX Pulse, Checkpoint IntelliStore, Crowdstrike intelligence exchange, CRITs, Soltra Edge, and CIF v3, to name a few, are presented comparatively in extenso in [23].

Throughout Section 2 we have analyzed the concept of security in a digital age. It is clear that simply configuring adequate IDS/IDPS solutions is not enough to ensure a trustworthy degree of safety. Attack patterns and malicious activities vary in both diversity and number, and new threats arise almost every day. Working on the concept of security as a process, we have identified two major research investments for this field. The first one relates to using DM/ML techniques to strengthen existing solutions. The reason for having a review on DM/ML techniques is only to reiterate their prospective advantages in security solutions. This approach is furthermore justified by the basic need to first understand and know your data—security data in this case—and to understand your target results. Association rule-mining, clustering, and classification techniques are just a few of the analytical tools we have centered our attention on within this review. Usually, these are applied over what we call offline data, such as audit logs, behavior, connection patterns, and so forth. The natural evolution is to then include these results dynamically in IDS/IDPS solutions, and it offers the second analyzed research field: security frameworks. This represents the main topic presented in our paper: a research and educational framework that allows the study of utilising intelligence threat data and DM/ML results in (near) real-time reactions/responses to security threats. The literature survey therefore includes the required analysis on the necessity of reliable threat intelligence data-sources (e.g., MISP and other enumerated projects) and on some theoretical and practical approaches to security frameworks that make use of the aforementioned concepts (HeteMSD, the framework submitted in [14], and Sphinx). We have chosen to analyze these frameworks because we consider them to be somewhat similar to our own with respect to having a voting mechanism that

validates the decisions. This is indeed to be expected since DM/ML results are subject to probability.

Our contributions for this section are summarized below:

- An analysis on DM/ML approaches in cybersecurity;
- An overview of some of the frameworks that could make use of DM/ML data in strengthening IDS/IDPS solutions;
- A contextual link between the literature review and our solution.

3. ECAD Framework Architecture

It is generally accepted that an intrusion is an unauthorized mechanism designed to access system resources and/or data. Intruders can be classified as external and internal. External intruders attempt to gain unauthorized access to system resources from outside the target network. Internal intruders attempt to elevate their limited privileges by abusing it. Our research is focused on creating a security framework that provides mechanisms to detect behavior of external intruders in the early stages of a cyber attack. We consider that an attacker becomes an internal intruder after successfully gaining access to internal resources. Vertical and, at least partially, horizontal privilege escalation (e.g., lateral movement) are successfully dealt with by antivirus programs and other commercial security products. This is beyond the purpose of the paper. We focus on prevention and pro-active decisions rather than on post-factum detection and containment. The software solutions available for our approach are varied, and both open-source and proprietary software could be used in a proof-of-concept implementation. One of our main concerns is to provide interoperability and enable collaborative security approaches, and not to compete with or replace an existing product. The ECAD framework is thus designed to be pluggable by default and include and/or interact with various other systems. This interoperability will also emerge later in this section, from the description of the tools that can be used, as well as the solutions chosen for the implementation of the ECAD platform prototype.

Our aims are to: (a) provide a guideline on how to integrate tools in order to obtain the desired results; (b) provide an environment for testing and analyzing the performance of custom tools in attack detection scenarios; and (c) provide a (near) real-time solution for addressing the initial phases of cyber attacks. Thus, the ECAD architecture we submit has the potential of inducing a new methodology for preventing cyber attacks in (near) real-time. This approach focuses on analyzing live traffic data and metadata and issue feedback that would facilitate adequate responses. Therefore, the architecture encompasses two external layers that are included within any cybersecurity context to mimic real-case scenarios and to trigger responses from the framework. These layers are shown in Figure 1 and briefly explained bellow.

Attack Layer: Any kind of attack scenario can be modeled using the Cyber Kill Chain pattern presented in [3]. Nowadays, an attack scenario includes more and more sophisticated techniques that are used by both malicious ("Bad Actors" in Figure 1) and well-meaning actors ("Ethical Hacker" in Figure 1—security specialists who analyze the ability of the defended system to cope with an attack). Each such attack begins with a much-needed reconnaissance phase. During this stage, actors try to analyze their target(s) to obtain as much useful data as possible and, at the same time, try to avoid raising suspicions. These data are used during the weaponisation phase to build a payload that is served within a so-called delivery stage. A subsequent exploitation stage is then in place, when the malicious actors try to execute auxiliary commands to ensure the persistence of the delivered payload. Well-meaning actors, who know the infrastructure of the protected network, will perform the same steps in order to identify what their opponents might discover or use.

We noticed that the reconnaissance phase is usually neglected by most IDS/IDPS systems. The actions performed during this stage are considered borderline legal and cannot directly incriminate a potentially malicious actor. This is one of the key points of our approach. We do consider that the information gained from this phase could be

extremely valuable in identifying the actual target of a potential attack. Therefore, the ECAD framework has been designed to also analyze these types of data. Furthermore, we consider that operating on these two incipient stages (i.e., reconnaissance and delivery) is a mandatory first step to be taken in a (near) real-time response scenario.

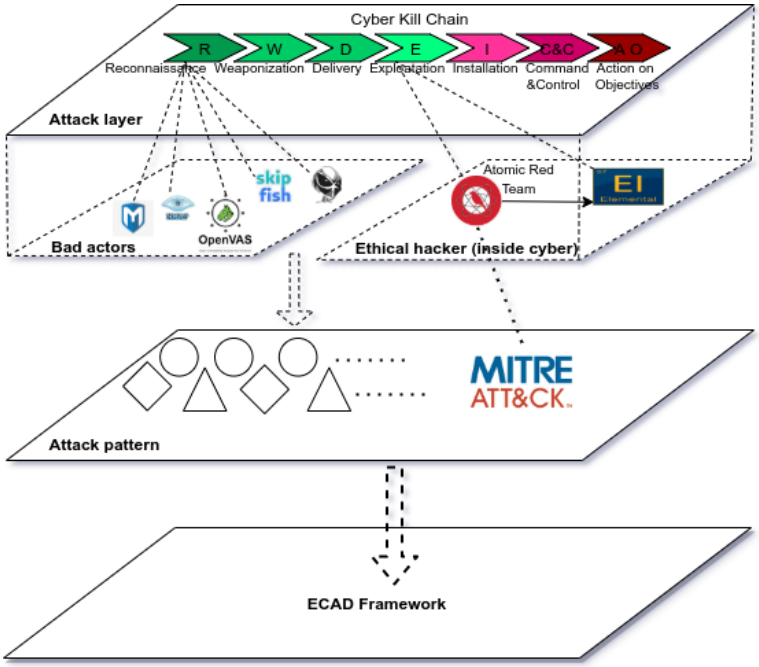

Figure 1. The operating context of the testing platform for the ECAD framework.

Attack Pattern: There are multiple scenarios and combinations of scenarios that an attacker can follow so as to disguise their action as much as possible. The actions performed during this weaponisation stage involve enriching existing successful scenarios with new, innovative techniques and delivering highly diversified attack patterns. It is a well-known fact that the actual weaponisation phase is usually hidden for the target. One should take into account that both attackers and defenders could use the same knowledge base, such as the MITRE ATT&CK® framework. Defenders use ATT&CK to test the defensive capabilities of the monitored targets and to discover unknown gaps in networks and products alike, or to perform tabletop exercises, assessments, and hands-on evaluations, as shown in Figure 1—the Ethical Hacker (Inside cyber) sub-layer. Aside from the actual beneficial security results, we also consider these data as valuable input for our own analytical layer. Relying on some clearly known, verifiable data implies strengthening the accuracy of DM/ML tools and is well within the purpose of ECAD.

The ECAD framework is comprised of four tiers, as shown in Figure 2: the Attack target (ECAD agent layer), ECAD collection/storage layer, ECAD analyzer and workload dispatcher layer, and ECAD correlation and detection layer.

Attack target (ECAD agent layer): The targeted system is the tier where the first elements of the detection chain within ECAD acts. These elements collect raw data from network traffic (Network capture sub-layer) and/or log files of protected systems (Log capture sub-layer). Additionally, at this tier, the system reacts through actions such as communication-blocking or redirecting to a controllable area, using security deception as an emerging cyber-defence tactic. This latter security approach may not be mandatory for common analytical frameworks, but we consider it does hold some potential in strengthening the actual study of cyber-attacks. Even though they are not explicitly represented in Figure 2, security deception tools like honeypots and sandboxing mechanisms allows further observation of the behavior of attackers once they have gained access to what they think is the targeted network. The actions of the attacker could be tracked trough

honeypots to reveal how the real threat actor behaves, and their actions could be detected without repercussions on the defended system. This provides support for the much-needed interoperability between ECAD and other detection and analysis systems. Furthermore, depending on the system's protected services, the Network capture sub-layer and the Log capture sub-layer components may be configured accordingly. Moreover, this tier ensures some form of data-preprocessing through filtering. At the Network capture sub-layer, only metadata obtained from levels 2, 3, and 4 of the OSI model are taken into account, depending on the actual communication protocols that are being used. At the Log capture sub-layer level, the filtering is done according to their format and the services that generate the data.

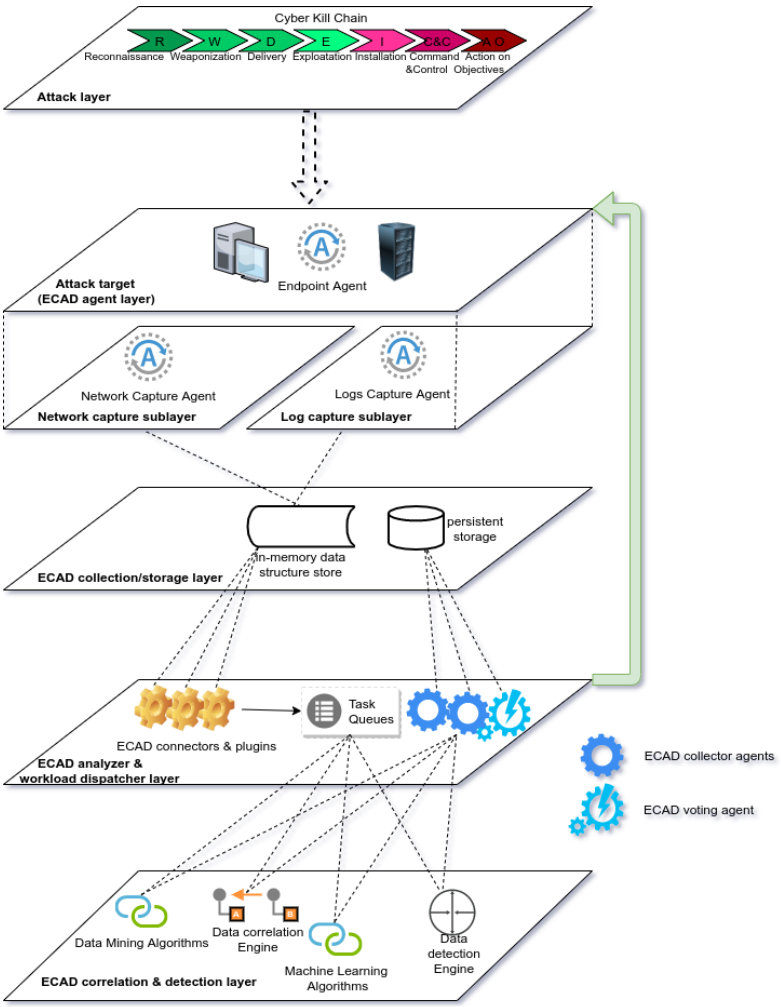

Figure 2. The four-tier architecture of ECAD.

ECAD collection/storage layer: We have designed this tier as an intermediate one between the information collection and decision-making tiers. There are two main goals for this approach. The first one is related to the fact that we need to perform multiple, independent analytical tasks over the raw incoming data from the previous tier. This is indeed required due to the fact that various DM/ML tools have different strengths and weaknesses. A (near) real-time approach must eliminate those weaknesses in order to reach a proper security decision. Moreover, different DM/ML tools might require different data models. We addressed these issues through In-Memory Databases (IMDBs). Raw data are temporarily stored within these fast storage tools. The underlying agents read their corresponding data chunks, with the last agent triggering clean-up stages. This ensures speed while minimizing the chances of data loss (i.e., all the agents are required to perform analytical tasks on the same data who would actually receive that data). The second goal is

to offer the means required to review and further investigate the results supplied by the lower tiers. We have considered persistent storage databases in order to solve this problem. The role of this persistent storage is to keep data that can be related with previous events and that have been analyzed and classified. Persistent storage improves the previously acquired raw data with the corresponding results supplied by the agents included in the lower tiers. This implies an UPDATE-like operation. Persistent storage is separated from the actual processing chain—that is, raw data are collected from a different source (IMDBs) and results are stored through a logging-like procedure. This ensures that the limitations of persistent storage do not impede on the speed of the system while still allowing the required audit information.

ECAD analyzer and workload dispatcher layer: We have designed this tier focusing on three key goals. First of all, we have stated that the same raw data should undergo multiple analyses that would compensate for the inherent probabilistic nature of DM/ML tools. Consequently, the same input data from the upper tier are modeled into different tasks. Each such task is processed independently from the others, in parallel. Once this processing is completed, the task is pushed to a dedicated queue (see Figure 2—Task Queues). This allows us to identify the actual analytical task rather quickly, without having an actual task selector module. The second goal is to be able to map the analytical results to the corresponding raw input data, issue a decision through the voting agent, and store the results for further investigations. These analytical results are received by both dedicated collector agents and the voting agent. The collector agents are responsible for persistently storing the results, while the voting agent is responsible for issuing a decision. This approach allowed us to also achieve the third goal: completely separated routes for the input and output data for this layer. This is, in our opinion, an essential property of a (near) real-time design: an asynchronous behavior through dedicated processing entities and communication channels that greatly diminishes the possibility of bottlenecks.

Further enhancing the idea of (near) real-time responses, we consider that it is better to have some initial reaction during a potential attack than a delayed one (or even not a reaction at all). The voting agent is designed to reach a decision as soon as possible, while at the same time being able to self-amend any potential false result. A schematic representation of the actions performed by this particular agent is given in Figure 3.

The core principle is that the voting agent would not wait for all these tools to offer a decision with respect to the analyzed data. Decisions are reached as soon as an acceptable confidence degree is obtained. To exemplify this behavior further, let us assume the following simple example: consider the case of a port scanning attack performed by a known entity. This entity has not yet attacked our protected endpoint(s), but has been previously reported for malicious behavior, and these reports have been acquired within the integrated malware information-sharing platform—denoted agent A1. There are other analytical modules that can provide responses based on behavior using DM/ML techniques—denoted agents A2, A3, A4. Let us assume that the weights are 30%, 10%, 20%, and 40%, respectively. A1 will provide the first response, followed by the others. If we further assume that a threshold of 40% is an acceptable confidence degree (Figure 3), then any second agent that responds would also trigger a reaction from the voting agent to the upper-layer endpoint agent. Late responses are also acknowledged and stored alongside previous ones. Moreover, if these late responses reverse the decision, then the entity is once more permitted to access the protected endpoint(s). We thus mitigate the probably good and might be good scenarios (see Table 1). Such clients would only perceive our response as a network glitch, which in turn might still be acceptable since we could actually provide a better degree of protection.

ECAD correlation and detection layer: This last tier of the ECAD framework is the actual computational one. It is meant to integrate threat intelligence data sources and analytical tools based on DM/ML techniques. Each resource is embedded within its own service. We chose this approach due to the fact that it allows parallel, independent processing over the same input data. Furthermore, it allows on-the-fly deployment of

new modules or removal of faulty ones. For instance, adding a new analysis tool implies running a new service, two dedicated agents included in the upper tier and a new task queue associated to this new service. We consider that a proper implementation of this approach theoretically implies zero downtime.

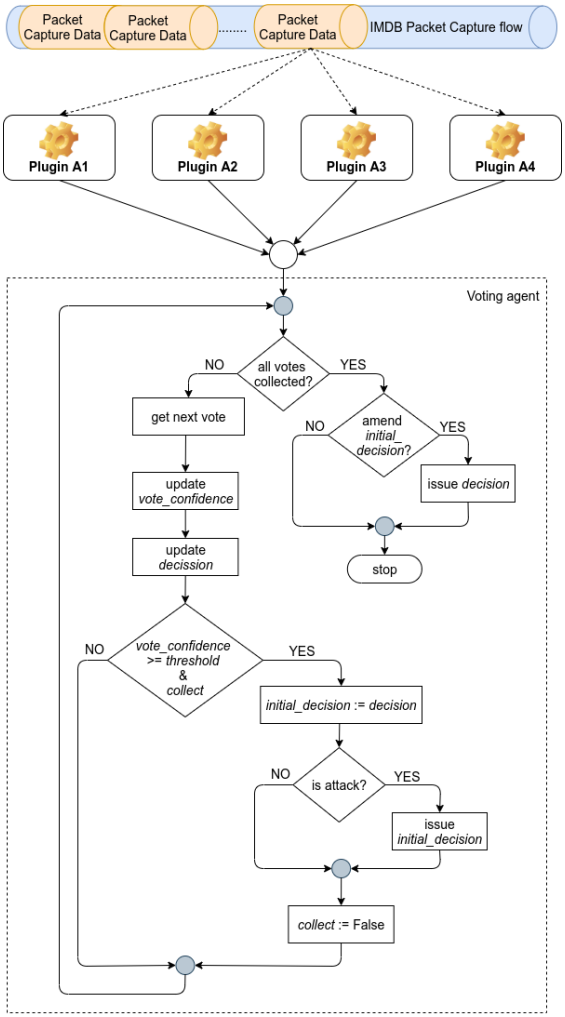

Figure 3. ECAD voting principle.

Another reason that justifies our approach is related to the previously presented voting mechanism. The enhanced flexibility of this design might allow developers to integrate faulty tools in their implementation. Additionally, some analytical tools might be more suitable for some particular types of attack. Threat intelligence data may not include the latest accurate details required to reach a decision. To counteract these potential pitfalls, we have designed this layer such that each analytical service is assigned a weight based on a given degree of confidence one has over a specific tool or another. Moreover, it is not guaranteed that all services would supply the results at the same time, as we have previously shown (see Figure 3). Services that do not supply their results in due time would still be able to contribute to the decision as a "second opinion" on the already taken course of action. We opted for this design model because it supports both self-amending a previously erroneous action and the opportunity to recalibrate the tools and improve the results.

Each analytical service needs to be designed as that it would closely follow the functions of the underlying detection component. Threat intelligence data sources are usually meant to search for correlated events, based on stored knowledge. We therefore designed such services to include the retrieval function, as well as to analyze the found

data and determine whether or not there is an undergoing attack. DM/ML techniques could be classified in descriptive and predictive groups. While predictive analyses are more suited to be used in (near) real-time solutions, descriptive ones need to be adapted for such scenarios. Both techniques are relevant through their actual result data rather than the means of achieving those results. We considered this last remark and designed these services accordingly. Sections 4 and 5 provide more details on how we actually achieved this behavior.

Section 3 is dedicated to presenting our architectural model for a framework that aims to improve security solutions and provide timely responses to undergoing attacks. We focus on the early reconnaissance and delivery phases and on prevention rather than post-factum responses. The architecture is organised on four tiers, modular by design, and allows fast integration with new security and analytical tools. We consider that this approach is suitable for modern implementation techniques and is meant to be implemented in virtualised environments, based on concepts such as Infrastructure as Code (IaC) and Software-Defined Networks (SDN).

4. ECAD Framework Implementation

An ECAD-like approach for dynamic, (near) real-time solutions in cyber security could prove to be difficult to implement on past technologies. IT concepts and tools have strongly evolved towards virtualisation and virtual components. This has made possible approaches that allow a change of perspective in the design stages. Thus, the effective work infrastructure becomes a dynamic element that can be modeled according to the runtime requirements of the developed applications. The first challenging problem was to decide how we were going to build, test, run, update, and monitor the services we were developing or integrating. We focused our study on industry-validated tools included within the Cloud Native Interactive Landscape provided by the Cloud Native Computing Foundation (CNCF). We also rely on concepts like Continuous Integration (CI) for integrating, building, and testing the code within the development environment. We choose to use the GitLab tool suite both to integrate the developed code and to enable CI pipelines to run on every commit or push. The steps defined for the CI process will dictate how we proceed towards Continuous Delivery (CD), to Continuous Deployment (CDP), and, finally, how we ensure that our developed services are continuously monitored and able to self-heal.

4.1. Implementation Technologies

The infrastructure we have implemented makes use of two types of components: stacks (orchestration part) and images (microservices). Stacks allow us to manage the actual execution of the embedded components and monitor such an execution flow. Images allow us to incorporate the actual functionality. In order to ensure the scalability of our solution, we choose container-based infrastructure, and thus we provide the availability of the ECAD services. With respect to the industry-validated tools, we use Docker Swarm and Portainer to manage the infrastructure stacks (Figure 4).

An infrastructure stack is a collection of infrastructure resources that we define, provision, and update as a unit (following the recommendations given in [24] for IaC). Each stack is defined by a source code that declares what infrastructure elements it should include. We defined the elements of a stack in `docker-compose.yml` files: compute, storage, and network resources, and the services that our proposed infrastructure platform provides. A stack management tool would then read the stack source code at runtime and call on an orchestration program to create an accessible service for managing the entire life cycle of the applications and the corresponding infrastructure. The purpose is to assemble the elements defined in the code to provision an instance of the stack. We followed the service stack pattern—any service stack manages the infrastructure for each deployable application component in a separate infrastructure stack. Thus, we ensure that we avoid the monolithic stack approach which could impede on the performance of our solution.

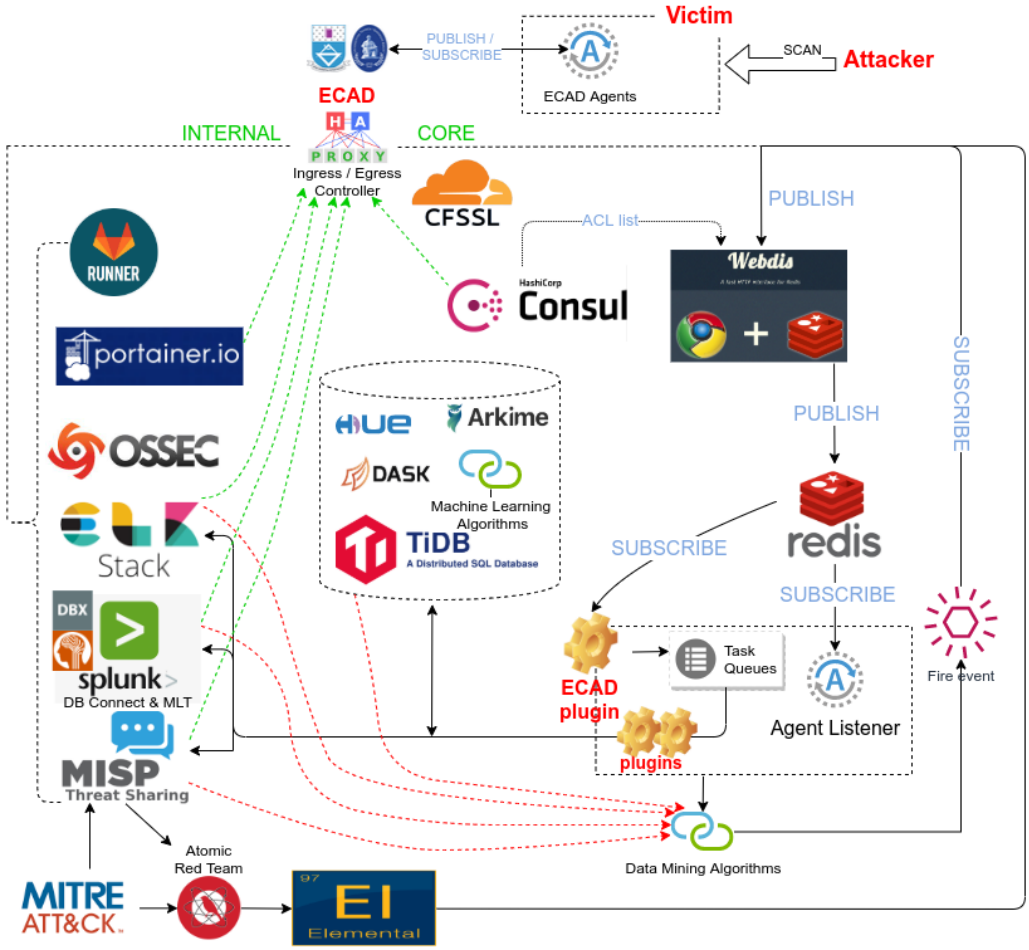

Figure 4. Sample technologies used to implement the ECAD framework and a sample data flow.

The core virtualisation tool we rely on is Docker [25]: it allows us to encapsulate entire applications and their environments within individual containers. The Docker Engine acts as the much-needed coordinator for all application containers. These container-based design patterns help us ensure that the right model is being used for the proposed system: an active entity would only make use of the strictly required components. This guaranties both resilience and easier management while maintaining adequate resource usage. As expected, some of the required container images were taken from Docker Hub (e.g., Portainer, Splunk, docker-flow-proxy, docker-flow-swarm-listener, MISP) or from private hubs, such as Elasticsearch, Kibana, Logstash, Metricbeat, Filebeat (docker.elastic.co). Alongside these, we have built our own images and registered them in gitlab.com (e.g., ecad, internal-tools). Figure 5 includes an overview of the whole ECAD framework implementation and the aforementioned container images. Blue services depict external components integrated within ECAD, while brick-coloured ones depict our own components: the certification and the `ecad listener` services.

We have previously indicated that our focus is a (near) real-time solution. Inherently, a distributed approach for achieving this goal must ensure both scalability and high availability. Container-based virtualisation in an IaC model is the first key step we have taken to support these desirable properties. Services are designed to work independently to perform logically atomic tasks. This ensures that components are called on dynamically, on a need-to-process basis. Our solution for implementing this framework thus has the ability to handle an increased load in a graceful manner with respect to the actual processing demand. High availability is strengthened due to the fact that a failing service would simply be replaced with a new replica instantiated by the corresponding orchestrator.

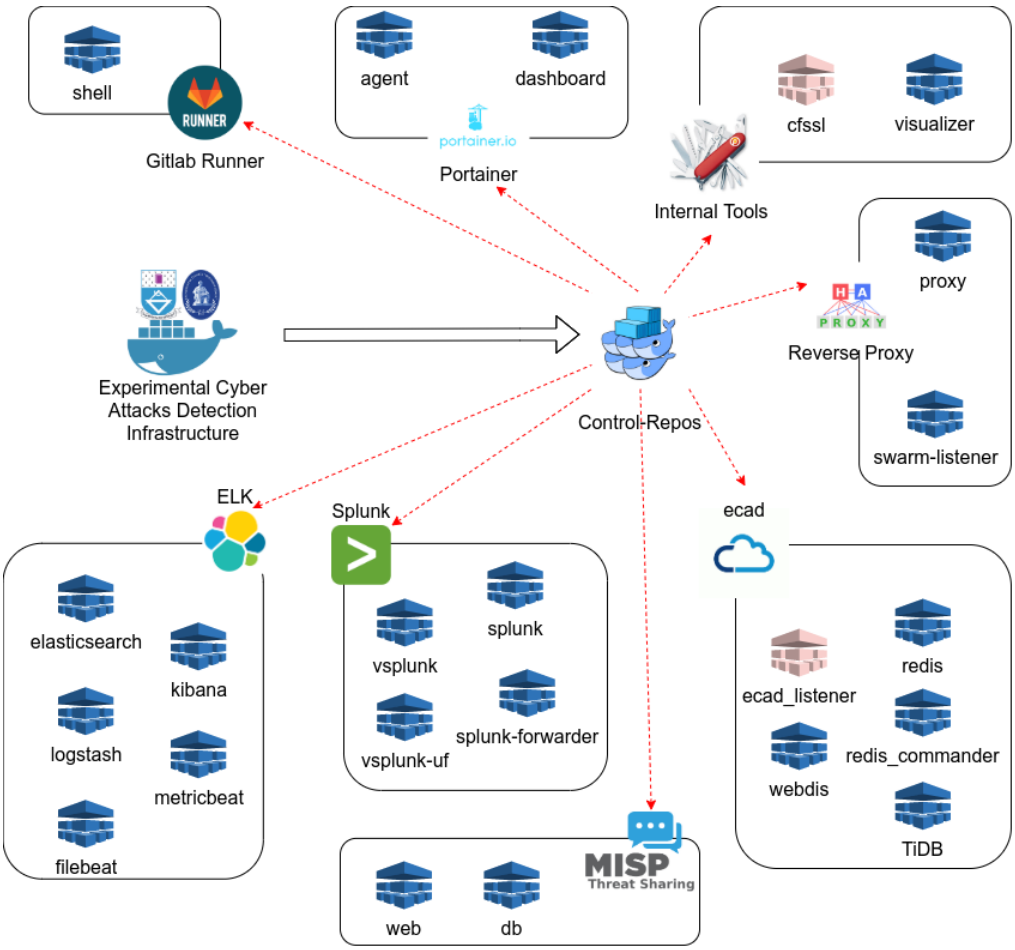

Figure 5. ECAD Stacks and Services.

Concepts and technologies like service discovery, cluster orchestration, Software-Defined Network (SDN), and IaC allow us to properly handle and efficiently make use of server cluster(s). The cumbersome tasks of load-balancing, task distribution, and fault tolerance are nowadays reduced to constraint mechanisms, such as roles and labels being defined in descriptive `docker-compose.yml` configuration files. The Docker engine module is responsible for parsing such input files and handling the set of available software services over the previously mentioned cluster(s). This allowed us to further our initial studies presented in [26,27] and include them as services within ECAD. These services are each embedded within Docker containers and are instantiated on demand—the required on a need to process basis behavior presented earlier. We also rely on Docker engine's subsidiary swarm mode feature. We defined the manager servers that handle membership and delegation, and worker ones that handle the actual instantiation of the received containers. We therefore consider we strengthen the practical approach to efficient, balanced resource usage. We preferred using a single tool suite since it ensures a more complete and cohesive integration. While there could be a lengthy discussion on whether or not other such tools (e.g., Kubernetes, HashiCorp Nomad, OpenShift, Apache Mesos) could be more efficient, we would like to point out that this implementation choice is merely our alternative for the proposed ECAD Architecture. Configuration options available nowadays are also vastly improved. For instance, one may control a desirable target state through different parameters instead of actually focusing on the intricacies of how to achieve that respective state. Cluster nodes that have been assigned as managers will then continuously monitor the actual cluster state and would amend any differences from the configured desired state. This gives us the much-needed ability to work on the actual functionality rather than on the menial underlying tasks. One example of using these facilities is related to the ecad_listener

services. These services are the so-called conduits through which data collected by the Network Capture and Log Capture agents are sent to the underlying layers. It is of utmost importance that these conduit services work properly and that they are immutable—that is, we considered exactly three replicas for preliminary testing purposes. The manager nodes would ensure this desired state, handling any eventual failings.

SDN is a cornerstone in efficient cluster management. Modern approaches to application management rely on services being dynamically created, updated, migrated within clusters, and/or scaled on demand. A direct consequence is that the underlying state data are highly volatile, making static solutions highly infeasible. Furthermore, we needed to make use of proxies as central communication hubs between all the ECAD services such that our components are truly loosely coupled. This raised yet another challenge to continuously monitor the potential cluster state changes and update the proxies' configurations accordingly. The efficient usage of Swarm networking capabilities is not enough by itself. Taking all this into account, we used a reverse proxy to act as a bridge between the external world and the ECAD services. We have defined a new software network called internal-proxy and we have attached all the required ECAD services to this new network. We did not make use of the reverse proxies' load-balancing options for the preliminary testing of ECAD. These are handled by Swarm networking tools. We are aware that in a more complex scenario (such as high-speed networks) the load-balancing feature usage could be rolled back to the reverse proxy. The next issue we needed to address was the stateful component required to handle the cluster. IaC/SDN solutions require such a stateful service to cope with the dynamic character of the virtualised cluster. Our approach was to integrate Viktor Farcic's [28] project Docker Flow Proxy: an extended version of HAProxy that loads the initial state data into memory and allows dynamic control through an HTTP API.

We have considered several implementation scenarios:

1. Create a new HAProxy image (including the updated configuration files) each time a new service is added to the framework. While this might seem a good idea for pseudo-static configurations, it did not suit one of the purposes of ECAD: dynamic integration of new services.
2. Store HAProxy configuration files onto a shared storage volume. Adding a new service would involve updating these shared configuration files and having HAProxy reload them. This approach would eliminate the need to rebuild the proxy image with every new service deployment, but it induces both traffic congestion (i.e., when reading/writing to the configuration files) and cluster volume handling issues. Additionally, technically speaking, a proxy is meant to operate in-memory.
3. Use proxies designed to work with or that are updated for Docker Swarm (e.g., the previously mentioned Docker Flow Proxy project). These software tools are integrated with the Docker Flow Swarm Listener and update their own configuration on-the-fly based on the newly acquired service state data.

We chose the third option for the current implementation of ECAD. It is a much more mature scenario, allows dynamic service discovery, and facilitates message exchange between software entities while hiding the actual physical details (such as which replica runs on which server, and so on). Basically, we operate within a software-defined network-routing mesh. We developed all the modules required for the first layer of ECAD using the Docker Flow Proxy and the Docker Swarm load balancer. We thus achieved the much-desired scalable, dynamic behavior and the real-time service integration envisioned for our architecture. There is yet another, perhaps hidden and less than obvious, advantage. SDNs allow better, more refined control over the virtualised components. We designed networks in such a manner that all services that should communicate with each other are placed inside the same network. As long as all we need is the name of the destination service, proper network design and usage is quite enough for the current stage of development. It also eliminates the need for external service registries and discovery tools such as Consul or etcd, which, in turn, implies that more physical resources are available for

the actual computational tasks. We are, however, aware that such tools could be useful for other scenarios as well, and ECAD could be enhanced. Consequently, we kept a minimal Consul module in our practical implementation (Figure 4).

Another concern in our implementation was to have encrypted communications for all systems connected to external networks. We developed an Internal Tools control repository with a dedicated service based on CFSSL. Its purpose is to generate all the certificates required for REST APIs, with its own private ROOT CA. The Internal Tools also include a dedicated module for validating or rejecting certificates. The whole service is actually an OpenSSL wrapper. We use it to secure communications over networks against eavesdropping and to authenticate and authorize the party at the other end. Whenever we integrate a new component, the Internal Tools suite is called in order to issue a new certificate. This certificate is then pushed to the corresponding docker containers, and is thus made available for that respective component. This approach ensures both the dynamic behavior of ECAD and internal and external TLS secured communications over HTTP/2. The second reason for this approach had been to better protect the internal components of the framework. As we have repeatedly pointed out, Docker is our main virtualisation tool. While being a stable and mature technology, it could induce security vulnerabilities. A potential attacker could inject a malicious service [29] that could supply false results. The desired flexibility of ECAD must not generate new security concerns, while still having the full benefits of IaC's loosely coupled modular software components. Our solution is to have certificate-based authentication and authorization for each new module before actually being integrated within our framework. Consequently, we greatly reduce the chance that a potential attacker could inject malicious analytical and/or decision components.

One final remark in securing ECAD relates to the concept of immutable infrastructure. A traditional IT infrastructure implies that servers are typically updated and modified in place. Installing, updating, or upgrading packages, and tweaking and optimizing configuration files is performed after server deployment. All these are still standard practices. These practices make servers mutable (changes occur after the initial set-up) and lead to configuration drifts: active configurations tend to greatly diverge from the initially approved and deployed ones over time. It is a well-known fact that mutability is one of the most critical attack vectors. A change could indicate that the server is compromised and it should either be quarantined, stopped, or terminated immediately. The best strategy is immutability, and simply denies all changes to the server/infrastructure. The concept of immutable infrastructure implies that no updates, security patches, or configuration changes are to be performed "in-place" on production systems. If any change is needed, a new version of the architecture is built and deployed into production. We have considered this approach from the very initial stages of our implementation—a security infrastructure must respect the immutable infrastructure paradigm. Through DevOps techniques, we used IaC scripts to create and manage an automated deployment pipeline that enables rapidly releasing software changes at scale. We maintain an accurate view over the system state through control-repos (see Figure 5). The whole infrastructure is updated should the need for a change occur (e.g., a service is updated or a new service is to be deployed). This solution is feasible due to our early adoption of IaC: all modifications are encapsulated in images which are in turn based on code. The whole infrastructure remains unchanged between any two consecutive updates, and any alteration is tested before being included in an update.

4.2. Implementation Particularities

4.2.1. Agent Layer

This layer includes two types of components. The first one is comprised of specialized and custom traffic capture applications and/or log data collectors. The ECAD agents included in this tier extract metadata from each packet that reaches the network interface, gather data from designated log files, and execute commands provided by the decision layers of the framework (Figure 6). Traffic capture may be performed at the endpoint level

or at the network infrastructure upper level by the Network capture sublayer. We use scapy as the endpoint data collector. This allowed us to develop a custom network sniffing agent focused on OSI L2, L3, and L4 metadata. Systems like Snort, Bro or Suricata could be valuable alternatives for infrastructure-related data with respect to the target scenario (i.e., if all the traffic data must undergo a full analysis set). Full packet capture systems like Arkime (formerly Moloch) or OpenFPC that can store data in PCAP format could also be used for network infrastructure-related input. Those types of systems can store the collected results using tools like ELK stack, but this might raise further challenges with respect to the certifications schema we are currently employing. Log collection and/or analysis systems such as Logstash or HIDS solutions like OSSEC can provide raw data on the monitored applications or services for the Log capture sublayer level. These tools are highly configurable, and the provided results may be represented by using a variety of formats and they supply some preprocessing capabilities that could prove useful for the the subsequent level—the ECAD collection/storage layer.

The second component is made up of the required applications that can enforce security policies provided by the framework. An example is given by the ECAD agents (IPTABLES Agent in Figure 6) that includes the reactive components needed to update firewall rules (e.g., Netfilter rules for Linux systems) according to the voting result from the decision layer of the framework (Figure 2).

The Dummy Agent noticeable in Figure 6 is a simple heartbeat-based component for health check scenarios. The Testing Agent is currently under development and is briefly discussed in Section 6.

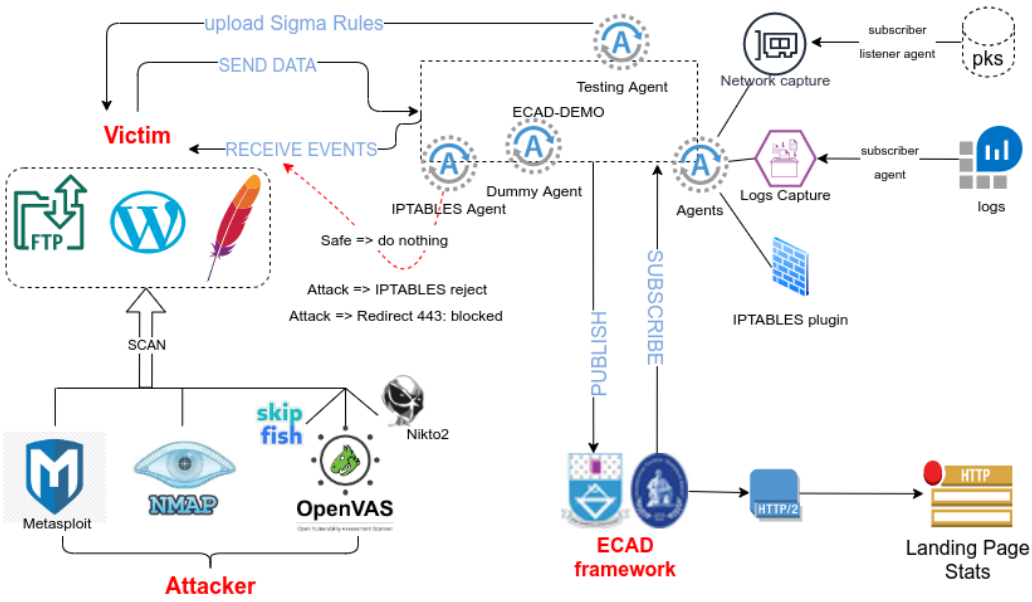

Figure 6. The testing platform for the ECAD framework.

4.2.2. Collection/Storage Layer

A solution designed for real-time traffic analysis and intrusion detection is highly dependent on scalable storage to ascertain performance and timely responses. Ensuring such storage capabilities is in itself a challenging issue, and it has been one of our concerns since the early stages of development. We designed our solution such that the data flow provided by the ECAD agent layer is distributed across the internal components of the ECAD framework. The ECAD agents use a publish/subscribe method to communicate between ECAD Framework components (Figure 4). Scalability is furthermore strengthened through using reverse proxies to route and control the data flow from the ECAD agent layer. We choose HAProxy as a reverse proxy provider in our preliminary tests, as previously stated in Section 4.1. Other solutions, such as Nginx or Traefik, may be used. We also route the traffic between all the internal components through this type of proxy.

The ECAD collection/storage layer makes use of both in-memory and persistent data stores. We chose In-Memory Databases (IMDBs) to accelerate mission-critical data transactions. The idea behind IMDB technology is straightforward: in-memory data greatly increases the overall performance by eliminating I/O operations and the need for indexes. While having potential pitfalls, such as data volatility and increased costs, the performance benefits for our approach are far more important: speeding up data transfer to the analytic layer is a must-have property of a real-time cybersecurity solution. We have chosen Redis as the first layer of data storage and Webdis to expose Redis PUB/SUB channels to HTTP clients (Figure 4). The actual data are collected from network/log capture agents included in the immediate upper layer (Figure 2). The previously mentioned persistent data store is used to keep event history, raw input data and the corresponding results obtained from the analytics layer (see Section 3). We made use of TiDB due to its many advantages.

4.2.3. Analyzer and Workload Dispatcher Layer

This layer embeds two types of plugins (or connectors) required to fulfil its purpose. An input-type set of plugins is required in order to fetch new data to be analyzed by the underlying correlation and detection layer. These plugins are subscribers to the IMDBs. The raw data and the corresponding processing form a specialized task that is fed to the appropriate task queue and thus dispatched for the actual data analysis (Figure 2).

The second set of plugins (connectors) are meant to process analytical results and to issue the corresponding decision based on the associated weighted votes. The results of the analysis are pushed to these output-type sets of plugins. There is a corresponding connector for each analytical task plus a separate decision/voting agent. The results are pushed to the persistent store by their related connectors and to the endpoint agent by the voting agent, respectively. Persistently saving both the original input data, the individual analysis results, and the intermediate and final vote is necessary to facilitate an eventual post-factum investigation. This is useful for security auditing and monitoring and can provide further useful information. We have chosen this approach because it offers completely separate data streams, and therefore there is no chance of bottlenecks between data that need to be analyzed (i.e., input data) and the provided results (i.e., output data).

The ECAD voting agent has been explained in Section 3 (Figure 3). Algorithm 1 describes the actual implementation we have used in this initial stage of development.

Algorithm 1: ECAD agent voting algorithm.

Input: votes provided by agents
Output: issue response event to the endpoint ECAD agent
$vote\_confidence \leftarrow 0$;
$collect \leftarrow TRUE$;
$decision \leftarrow NONE$;
while *not all votes collected* **do**
  get next vote from agents;
  update $vote\_confidence$;
  update $decision$;
  if $vote\_confidence \geq threshold$ *AND* $collect$ **then**
    $initial\_decision \leftarrow decision$;
    issue $initial\_decision$;
    $collect \leftarrow FALSE$;
  end
end
if $initial\_decision \neq decision$ **then**
  issue $decision$;
end

4.2.4. Correlation and Detection Layer

This layer is the one we are currently focusing our efforts on. The purpose of this tier is to include the actual analytical tools and other trusted data sources that are used in determining whether or not the monitored endpoints are under attack. These computational components are called on whenever a new task is published, and work independently on their input data to provide a vote of "allow" or "deny". What sets our approach aside from others is that we allow for the same data to be processed by different tools such as ELK, Splunk, MISP, OSSEC and some of our own methods. Each tool is given a certain degree of confidence—as expected, well-tested and trusted elements like MISP or Splunk are initially assigned higher degrees—and votes are weighted based on these scores. Initial degrees are static and specified through configuration files for the time-being.

A discussion on integrating Splunk and ELK is given in [27]. ELK had been rather easy to model and integrate as container-based services since it had been developed as a Cloud-ready tool suite, and it is highly configurable. Splunk, on the other hand, includes both a data store that allows for so-called local input and the analytical components required to work on this data. The potential pitfall is that Splunk is actually a commercial tool suite and customization is not always straight-forward. We made use of these tools to perform analyses on OSI L3, L4, and L7 data for different scenarios.

MISP is another tool suite we included at present. We have shown the potential benefits it can offer in Section 2: a rather solid correlation engine based on a vast threat intelligence data source. As we have previously stated, our intentions were to have MISP as one of the analytical tasks that would supply quick answers based on IoC information. The particular challenges of integrating MISP within our infrastructure as a standalone swarm service are related to its design philosophy. Fist of all, MISP has been created to operate on a standalone server. There are only a few projects that attempt to create a Docker container image (we have identified only three viable ones). ECAD's architecture requires that all its underlying services are also suitable in swarm mode. We addressed this issue through a custom configuration based on the Harvard IT Security project. The second challenge derives from the actual strengths MISP could offer: its correlation engine does not automatically supply alerts. This is not suitable for a (near) real-time approach. We solved this problem through two dedicated input and output analytical agents developed using PyMISP. The input one would create a new MISP event and push it to an active replica of the engine. The output one is then triggered; it will read the correlation results and formulate a decision based on this data. A brief example is presented in Section 5.

The next developed analytical agent is based on OSSEC. It is a well-known fact that this tool suite has been designed following manager–agent architecture. The agents are meant to be deployed on the protected systems and are both log-collector and reactive components (i.e., apply security policies). The manager server is the core piece responsible with file integrity checks, log, event databases, and other auditing tools. We have included both these modules in ECAD: the OSSEC agents are deployed on the monitored system, and the OSSEC manager is integrated in an analytical service. The purpose of the service is, as expected, to issue votes on the received data set (just like any other analytical component included with this fourth tier). Our reasoning for this approach derives from one of the key desirable functionalities of ECAD: multiple analytical steps performed over the same data. The OSSEC agents on the protected system are therefore responsible for this data collection stage, aside from their built-in initial reaction.

Our own analytical tools are focused on DM/ML techniques. The first set of studies we have performed relate to using association rules in (near) real-time detection scenarios, and we have developed an analytical service dedicated to this purpose. Since association rule-mining is a descriptive DM/ML technique, the actual DM/ML analysis is performed on offline data sources, such as application logs. The ECAD service is fed with high-quality association rules and maps them over search-specific data structures (tries, in our particular case). The rule in itself is divided into its respective antecedent and consequent itemsets. The antecedent represents the path in the trie, while the consequent and the confidence

are the actual stored data. Our (near) real-time approach undergoes the following steps: (i) an access pattern is sent to the analytical service; (ii) the rule trie is traversed based on the items that compose the pattern until either the current route has only one admissible branch or a data node is reached; (iii) having access to the data node, the consequent is then matched with the final part of the analyzed pattern—if a match is found, then the corresponding vote is issued to the upper-layer agents. Examples are given in Sections 5 and 6, respectively.

Section 4 presents our implementation for the proposed architecture, which is detailed in Section 3. Everything included in this section is our contribution, aside from the explicitly mentioned external tools and software packages. Following the four-tier description of our framework (Figure 2), we have chosen a modern implementation guideline, such as IaC and SDN to provide an immutable infrastructure that combines both analytical and IDS/IDPS solutions. Figures 4 and 5 provide a generic overview of this implementation. The voting agent and the dedicated input/output data streams are two key advantages of our approach. Both have been implemented to achieve and enforce the (near) real-time behavior that we consider highly important.

5. Results

The main topic of this paper is to submit a potentially new methodology for studying and preventing cyber attacks and to provide a framework to support this methodology. Our key focus is on an IDS/IDPS solution that offers near real-time reactions to potential attacks. This approach is not meant to obsolete existing security tools; it rather represents a means to integrate appropriate DM/ML instruments, security intelligence data, and so forth into collaborative defensive solutions. The ECAD framework is at present in its prototype implementation stage. Closely following the architecture described in Section 3, the practical realisation described in Section 4 heavily relies on virtualisation tools and pluggable components. We consider that these are the first important results that we have achieved through the proposed approach. All the underlying components of ECAD are running in dedicated containers and are being exposed through uniform access interfaces. This implies that we have achieved a platform agnostic solution that may be deployed on any system capable of running container software. To further demonstrate this portability, we have deployed the prototype ECAD implementation on three different environments:

1. Seven systems (with Intel® Core™ i5-7500 CPU @ 3.40 GHz, 6 M Cache, 8 GB RAM and Gigabit Ethernet);
2. Seven virtual machines running in an OpenStack Ussuri private cloud based on a CentOS 8 cluster with 8 Raspberry Pi 4 Model B Rev 1.4 (Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5 GHz, 8 GB RAM and Gigabit Ethernet);
3. Seven virtual machines running in a Proxmox Virtual Environment private cloud (8 GB RAM and Gigabit Ethernet).

All running CentOS 7 with dockers installed in swarm mode—three managers and four workers. All clusters are organized on 8 stacks, 44 containers, 21 services, 77 images, 50 volumes, and 65 networks, as reported by Portainer.

Moreover, container-based architectures support scalability through container orchestration means. We made use of these advantages: an analytical tool included in the ECAD correlation and detection layer would run as an independent task, replicated on a need-to basis. It allows for parallel analysis over the same packet capture data for a dynamic voting agent, and speeds-up the decision-making process with respect to the nature of the data being investigated. There is yet another important note on the architecture we submit. The actual data-flow we have proposed for ECAD is somewhat circular by nature (see Figure 2). Data collected by the dedicated agents are forwarded through dedicated channels to the underlying layers, while responses and reactions are obtained through different routes. Alongside the aforementioned scalability benefits, our circular approach diminishes the risk of data-flow bottle-necks and potentially further reduces the actual response/reaction times.

We enforced the pluggable behavior of ECAD by applying Viktor Farcic's [28] project Docker Flow Proxy recommendations to our particular environment. The reverse proxy we have used will thus automatically detect new components that could be included within ECAD and forward requests appropriately. As expected, the same level of automation is achieved when an unused or a faulty module needs to be taken down.

We have shown in Section 4 how we have integrated the MISP threat security intelligence data source and correlation engine in ECAD. There are two main reasons to support our decision. The first is the obvious one: since there already exists a tool that can help detect and prevent attacks, why not include, under the AGPL (Affero General Public License), such a tool within a Collaborative IDS/IDPS? It is especially useful on new server/application deployments that have gained the attention of malicious actors and could be subject to attacks. In the following, we present a use-case we have successfully resolved using this approach.

The experiment is related to malware injection performed within an APT. We assumed the aforementioned malware had already been delivered onto the monitored system. It is a common case for traditional solutions that analyze malware based on the signatures of the underlying program files. The dedicated ECAD endpoint agent identified the collateral movements with respect to application behavior and data transfer patterns. The data collected on the potential victim is modeled into an event according to the MISP attribute system. Moreover, this event is labeled with threat level 4 (i.e., the lowest threat level admissible in MISP). The event is then embedded in an analytical task by the agents included in the ECAD analyzer and workload dispatcher layer (see Figure 2). The corresponding module in the ECAD correlation and detection layer pushes this event data in MISP. This threat intelligence platform would in turn update the event through its own analytical means. We target the identified related events and their corresponding sources and update the threat level of our own event through a weighted schema based on the degree of confidence in an organisation and on the threat level of its corresponding event. The following decisions are then issued to the voting agent:

- Threat level 4 (unmodified)—enhance monitoring, no other attack-related action; this is the case of a raised suspicion that does not provide sufficient data for an actual decision, but still needs monitoring;
- Threat level 3—issue a temporary quarantine of the suspicious applications and a temporary traffic ban to/from the eventual remote IPs; this is the case of potentially materialized suspicion, actions are required to be taken, but there is still a level of uncertainty with respect to the source;
- Threat levels 2 and 1—issue permanent quarantine of the suspicious applications and permanent traffic ban to/from the eventual remote IPs; there is definitely an undergoing attack, the actions are meant to be taken immediately.

Observed response times are of the order of several tens of seconds until the actual decision is made and the corresponding actions are issued. While these results have been obtained in simulated environments and there is room for further improvement, the simulated attack had been contained. Moreover, data related to the attack and the related issued actions had been stored for further analysis. These results are also meant to exemplify our idea of having an initial reaction rather than no reaction at all, as stated throughout Section 3.

The second reason for our approach is less obvious and applied to all analytical tools, including MISP. A successful implementation of security tools does not expose its internal components and reaction/response data to any potential attacker. In our approach, ECAD embeds these decision support tools within its innermost tier, and this hides their existence from malicious entities. Moreover, each communication between these components is at least secured with respect to its transport layer and each component undergoes authentication and validation stages. This is possible due to the automated certification mechanisms we have adopted for our practical implementation.

The ECAD correlation and detection layer also includes some of our own approaches for different DM/ML tools. While these components are still undergoing testing and are not the main focus of this paper, we will present some of the preliminary results we have obtained. Reconnaissance phases performed to identify available exploits targeting Web CMS (Content Management Systems) solutions usually aim to identify specific resources that are known to be vulnerable (such as JavaScript modules or PHP script files). An attacker having obtained knowledge on such weak spots may then directly perform the delivery phase. Using association rule-mining techniques on OSI L7 log data, we have noticed that most such discovery stages directly target those vulnerable resources, without having a referrer entry in the HTTP server log. Based on this note and on previously known HTTP access logs, we have modeled an itemset as a tuple of $\{SOURCE\_IP, HTTP\_METHOD, REFERRER\_TYPE, TARGET\_TYPE\}$, having:

$$REFERRER\_TYPE \in \{NULL, VALID\}$$
$$TARGET\_TYPE \in \{INTERNAL, EXTERNAL\}.$$

The *INTERNAL* value above implies that the resource is not meant to be accessed directly. We then model the actual association rule obtained from such an itemset as having:

antecedent $\{SOURCE\_IP, HTTP\_METHOD, REFERRER\_TYPE\}$

consequent $\{TARGET\}$.

These rules are then mapped over search-specific data structures that allow parsing the antecedent down to the point where the consequent may be uniquely identified. A navigation pattern that reaches an INTERNAL target with a NULL referrer included in the antecedent is considered to be malicious and such an agent would immediately yield an attack alert. This implies that our endpoint agent on the monitored entity would therefore inject a DROP rule into the firewall and would thus at least hinder the potential attacker. This implies that such a potential approach would not supply the complete results required to perform a successful attack.

We are testing a similar approach for detecting and preventing HTTP flood attacks. These may be perceived either as reconnaissance (i.e., determine information on the defensive tools), or delivery phases. We also make use of association rules for one of the ECAD analytical agent, but this time, the consequent of the rule (i.e., TARGET item) has a different set of values: (i) VALID—meaning a legitimate URL and (ii) INVALID—meaning a randomly generated URL that identifies no actual resource. The case we address is the one of brute-force HTTP GET flood. Rules that end with $TARGET = INVALID$ consequent indicate a definite denial-of-service attempt. Alongside this agent, there are also dedicated ones that analyze OSI L3 data in order to check the corresponding underlying TCP connection patterns. A host attempting to establish a high number of connections with the same target network service (i.e., the HTTP server for this scenario) represents an indication of an attack. Through the described voting algorithm (see Algorithm 1), the decision is issued to DROP all connections to that particular host. While these initial results are promising enough, another interesting remark may be formulated: we deny any further connections with any attacking host. This implies that all attacking hosts would be denied access, should the monitored system be subject to a distributed attempt (i.e., DDoS) on the same network service. The reaction time is indeed higher, but we still have the benefit of recovering rather quickly.

One more important result that we have observed is that responses/decisions/reactions are received from the ECAD Voting Agent (see Figures 2 and 3) within, at most, minutes after the suspicious access occurred. This is possible due to the design of the ECAD Voting Agent: the first reaction is based on a confidence threshold and it still allows for possible amendments if the initial decision was incorrect. Our approach could be rephrased as: Can we act on a suspicious action pattern before it is actually finalized? While, at present, these data have been observed in simulated conditions, and while response/reaction times still

need further thorough assessment, we consider them to be promising. As we previously mentioned throughout this paper, our main target with ECAD is to detect and isolate/act on malicious actions during the reconnaissance phase. In doing so, we are able to slow down or actually impede potential attackers during the very initial stages of their actions. We do not target the clearly "Known Bad" (see Table 1)—for these scenarios, there already are trustworthy security tools that solve them.

6. Discussion

We have designed and implemented the ECAD framework as a research and educational tool that allows the study of cyber attacks and the corresponding actions to prevent them. The main focus had been on the concept of (near) real-time responses to online threats through means of data analyses and DM/ML techniques. We strongly believe that prevention during the early reconnaissance and delivery phases is a key property of any proper security suite. We would like to point out that the correctness of the results is still undergoing testing, as this framework is in the initial development phases. We focused only on getting the workflow done and having the decision system up and running. We pointed out throughout Sections 3 and 4 that it is better to have a reaction during an active attack than a post-factum analysis or no reaction at all. The voting algorithm we have submitted and developed is based on this idea. Moreover, the solution we have proposed has a self-amending behavior—that is, the initial decision reached by our voting mechanism may be modified on-the-fly provided that there are enough results that revert the original course of action.

We have shown through the literature survey presented in Section 2 that complex attacks require several stages of setting up and preparation prior to the actual actions. Potentially harmful actors need to first assess the target system's defence capabilities, to identify the strengths and the weaknesses of the underlying security measures and of the target applications. While some of these initial stages may be hidden to the victim (such as the weaponisation phase with respect to the Cyber Kill Chain), actions taken during reconnaissance and delivery are almost always observable provided that adequate monitoring instruments are in place. This is the proactive approach we base our reasoning on, and we make use of such data in at least hindering the actions of bad actors. We consider that an attack cannot be successfully performed if the aggressors do not acquire enough information on their target. This implies that the agents included within the ECAD framework would actually be able to stop a cyber assault before any actual damage is incurred on the target. This proactive reasoning could also prove to be successful on some classic attack patterns, such as DDoS. We presented one such study on preventing HTTP GET flood attacks using association rule-mining techniques in Section 5. While we have not focused on a thorough assessment of reaction times, we noticed that the attack had been dealt within a few minutes. This is a significant result, especially if we consider the most recent report published by ENISA in [30] on DDoS attacks. Of such attacks, 52% are performed in under 15 min. This 15 min time-frame is usually observed for brute-force, aggressive DDoS which could also be solved through existing IDS/IDPS solutions. The remaining 48% is reported for more persistent attack patterns and could pass through classical defence mechanisms. The preliminary results we have obtained emphasize the potential benefits of our approach even for this second category of sustained DDoS attacks.

The pluggable architecture we have developed for ECAD is another important advantage of our approach. This allows the dynamic integration of new detection and prevention modules. Let us consider the example of brute-force FTP attacks. Bad actors employ novel, more aggressive distributed techniques and are able to avoid detection through constantly changing their IP addresses after a predetermined, limited number of unsuccessful attempts. The solution utilised in HTTP GET flood attacks could be modified through changing the meaning of the association rules presented in Section 5: the antecedent of the rule could include a sequence of failed attempts, and the *TARGET* item in the rule consequent could point to a value included in the targeted users' pool. The flexible im-

plementation of ECAD would require: (a) a new module that includes the corresponding patterns deployed within the correlation and detection layer, and (b) a new job dispatcher and a new queue for the analyzer and workload dispatcher layer. This is made possible through our IaC-based solution. Other existing threat intelligence data sources or analytical tools (such as ELK, Splunk, MISP, or OSSEC) could supply adjacent results for the voting agent to either strengthen or weaken the attack decision. These multiple different analyses performed over the same input data are another particularity of our solution (see Section 4.2.4). It allows us to take different perspectives of the same data into account, having different meanings and targeting different components of an attack, instead of just focusing on multiple tools that belong to the same analysis category.

We would like to also discuss the concepts of (near) real-time detection (and prevention) vs. post-factum detection. We consider that post-factum detection begins with the onward exploitation phase with respect to the Cyber Kill Chain model. This implies that the reconnaissance and the delivery phases had been successful—that is, the attacker already gained some sort of access to the monitored system. We argue that this involves a late decision solution and does not comply with the purpose of the ECAD framework. We address the concept of first-hand prevention, and we emphasize that this is merely the first step to be taken. It is not the only step, and it does require further analyses to be performed over the monitored system, as well as auditing the chain of decisions that has triggered a given course of action. We address this through the persistent storage tools we have made use of (refer to Sections 3 and 4—the collection/storage layer). This represents one key element of Bruce's Schneier "security-as-a-process" concept. It also relates to "security as a science" and "science driven by data" notions [16]. We make use of existing data (by employing different analytical agents and tools) and we produce even more data for further investigations and potential improvements. We have repeatedly acknowledged throughout this paper that good data-driven solutions rely on a profound understanding of the input information available and on the results that are to be obtained.

More and more data are always a challenge in itself, and we need to focus future research efforts on this as well. We must ensure that every processor or service we distribute in the swarm does not receive a larger volume of data than it can process. A less-than-optimal data split between tasks could severely impede on the (near) real-time characteristic of ECAD. A possible solution is to develop a system that aggregates usage metrics and gives us the ability to issue alerts or make certain decisions when a given threshold has been exceeded. We identified a possible solution for this—Prometheus, but it is not the only tool that should be considered. Results supplied by Prometheus could be used in Auto-Scaling Docker Swarm Services.

We have previously stated that ECAD has been designed and implemented such that it would consider multiple analyses over the same data (access pattern, application behavior, etc.). Our preliminary testing phases have been focused on OSI L3, L4, and/or L7 data. However, the Network capture agents are also collecting OSI L2 information (see Section 4.2.1). This could prove to be yet another useful feature encompassed in ECAD. MAC address spoofing attacks are rather difficult (if not next to impossible) to detect. Harmful actors could try and use this course of action in order to gain unauthorized network access. A correlation analysis including L2 MAC addresses, probable time-frames for the physical location of the monitored devices (e.g., AP migration in WiFi environments), OS identification tags, and so on could prove beneficial in detecting suspicious elements. Such an approach would enforce the prevention capabilities that ECAD could offer in scenarios that rely on the Zero-trust security policy model. This model implies that complex networks' security is always at risk to both external and internal factors. A practical use case for this ECAD development is the well-known bring-your-own-device (BYOD) policy within such Zero-trust environments. Of course, the second valid use case would be the actual research that could be performed over this extensive set of data.

Another desirable enhancement of ECAD would target the concept of proactive defensive techniques. It is our belief that most IDS/IDPS suites are rather reactive with

respect to detecting threats. Moreover, performance evaluation and auditing are also made post-factum. A possible approach to counter-acting this behavior is employing teams of security specialists to mimic attacks on the protected systems and their components. The purpose is to identify potential weaknesses and issue corresponding corrections and amendments for the affected security modules and/or faulty application modules. Projects such as Elemental (developed at the Berkeley University of California) provide the means to integrate threat libraries, Atomic Red Team test suites, and other such relevant data. The ECAD framework includes two main modules to support such a study: the external Ethical hacker (inside cyber) sublayer (see Figure 1—Section 3) and the MITRE ATT&CK® framework within the internal correlation and detection layer. These modules could be configured to perform controlled attacks and identify the vulnerable components of the monitored system and applications. The ECAD Testing agent (see Figure 6—Section 4.2.1) is included specifically for these tasks. Alongside the previously mentioned benefits, such testing scenarios could be used to strengthen the DM/ML techniques that are being used within our framework. Instead of merely reacting to incoming threats, one could make use of this framework to actually conceive and model new attack patterns and better prepare for an eventual real-world intrusion.

7. Conclusions

This paper is meant to present our approach to cyber security frameworks. Our practical solution is named Experimental Cyber Attack Detection (ECAD) and we have developed it to provide an insight on how different security tools, threat intelligence data sources, and DM/ML components could be orchestrated to detect and prevent cyber attacks. We provide an architectural model and a possible solution to implement it using novel, state-of-the-art techniques, such as IaC and SDN. The framework in itself is highly pluggable and provides excellent opportunities for both the research and study of attack patterns and any eventual responses issued to counter-act such intrusive actions. Moreover, we have focused on a communication protocol agnostic approach with respect to the actual analyzed traffic patterns. We consider that IDS/IDPS solid solutions should not focus on a single protocol suite, especially in such cases that the monitored target supplies multiple network services. Initial results are promising for the studied attack patterns.

We focused our study on (near) real-time approaches and addressed this behavior through (a) a circular route for input and output data and (b) a self-amending voting mechanism that issues verdicts on suspicious traffic and application patterns. Our approach is based on modelling suspicious activities with respect to the Cyber Kill Chain concept. Detection is mainly organised alongside the reconnaissance and delivery phases. We have determined that these two stages are less explored in strengthening prevention solutions and that they could prove to be a valuable source of data. While prevention in itself is a highly challenging desideratum, we have shown that our approach could at least severely hinder potential intruders. Moreover, another potentially new idea we have submitted is that we analyze the same data input with multiple different tools independently. This offers different perspectives on the same source, each such new result supplying new data that characterises potentially harmful actions. We consider this is a major advantage of our approach since it could provide better insights for both classic and new attack patterns. It could also be used to understand benign abnormal access scenarios and how these might influence the false-positive/negative results issued by classic DM/ML techniques.

We also consider that a possible solution to address the worsening problem of cyber attacks is to make sure that the security controls employed by organizations are consistently effective. It is another research area that can be addressed by the ECAD framework through components like the Testing agents. Their purpose would be to proactively identify configuration issues and what assets could become interesting targets for attacks and could be embedded as sub-modules of automated security-auditing techniques. The solution we have presented is not a finalised product, but rather an evolving concept. We aimed to address the ever-evolving requirements of the IT industry and the various cyber threats and

related security defence countermeasures. It also gives us the opportunity to contemplate on Bruce's Schneier "security-as-a-process" concept and on how poor and misguided application of AI/ML techniques could be used to derive new security threats. One of our goals is a better understanding of security-related data. Public research had been invested in how DM/ML solutions could improve security, but there is little or no information on how AI could actually be used as an offensive tool.

Author Contributions: All authors have contributed equally to this manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: We are very grateful to Alexandru Coman from Yonder's Cybersecurity and Security Awareness division https://tss-yonder.com/solutions/cyber-threat-aware/ (accessed on 13 July 2021). for providing us with the third environment in Yonder's private cloud to test the platform that implements the ECAD Framework and for his helpful advice. The authors would also like to acknowledge Gheorghiță Butnaru's contribution in the infrastructure deployment. Gheorghiță is a system and network engineer at the "Gheorghe Asachi" Technical University of Iasi, Romania. Consistent research efforts supporting this paper have been performed by members of the Open Infrastructure Research Center (OIRC) https://oirc.tuiasi.ro (accessed on 13 July 2021): Cătălin Mironeanu, Alexandru Archip, Cristian-Mihai Amarandei.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ECAD	Experimental Cyber Attack Detection
MISP	Malware information-sharing Platform
UCST	Universal Cyber Security Toolkit
APT	Advanced Persistent Threat
IDS	Intrusion Detection System
IDPS	Intrusion Detection-Prevention Systems
HIDS	Host-based Intrusion Detection System
NIDS	Network-based Intrusion Detection System
LIDS	Log-based Intrusion Detection System
SIEM	Security Information and Event Management
DoS	Denial of Service
DDoS	Distributed Denial of Service
R2L	Remote to Local
U2R	User to Root
DM	data-mining
ML	Machine Learning
AI	Artificial Intelligence
IoC	Indicator of Compromise
IMDB	In-memory databases
CI	Continuous Integration
CD	Continuous Delivery
CDP	Continuous Deployment
CNCF	Cloud Native Computing Foundation
SDN	Software-Defined Network
IaC	Infrastructure as Code
BYOD	Bring Your Own Device

References

1. Galal, H.S.; Mahdy, Y.B.; Atiea, M.A. Behavior-based features model for malware detection. *J. Comput. Virol. Hacking Tech.* **2016**, *12*, 59–67. [CrossRef]
2. Schneier, B. The Process of Security. 2000. Available online: https://www.schneier.com/essays/archives/2000/04/the_process_of_secur.html (accessed on 27 April 2021).
3. Hutchins, E.; Cloppert, M.; Amin, R. Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. In Proceedings of the 6th International Conference on i-Warfare and Security, Washington, DC, USA, 17–18 March 2011.
4. Strom, B.E.; Applebaum, A.; Miller, D.P.; Nickels, K.C.; Pennington, A.G.; Thomas, C.B. *MITRE ATT&CK®: Design and Philosophy*; Technical Report; The MITRE Corporation: McLean, VA, USA, 2020.
5. Straub, J. Modeling Attack, Defense and Threat Trees and the Cyber Kill Chain, ATT&CK and STRIDE Frameworks as Blackboard Architecture Networks. In Proceedings of the 2020 IEEE International Conference on Smart Cloud (SmartCloud), Washington, DC, USA, 6–8 November 2020; pp. 148–153. [CrossRef]
6. Taipale, K.A. Data-mining and domestic security: Connecting the dots to make sense of data. *Colum. Sci. Tech. L. Rev.* **2003**, *5*, 1–83.
7. Liu, B. *Web Data-Mining: Exploring Hyperlinks, Contents, and Usage Data*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2011. [CrossRef]
8. Lee, W.; Stolfo, S.J. Data-Mining Approaches for Intrusion Detection. In *7th USENIX Security Symposium, SSYM'98*; USENIX Association: Berkeley, CA, USA, 1998; Volume 7, pp. 6–21.
9. Lee, W.; Stolfo, S.J.; Mok, K.W. Algorithms for Mining System Audit Data. In *Data-Mining, Rough Sets and Granular Computing*; Physica-Verlag GmbH: Heidelberg, Germany, 2002; pp. 166–189.
10. Lee, W. Applying data-mining to Intrusion Detection: The Quest for Automation, Efficiency, and Credibility. *SIGKDD Explor.* **2002**, *4*, 35–42. [CrossRef]
11. Jin, Z.; Cui, Y.; Yan, Z. Survey of Intrusion Detection Methods Based on data-mining Algorithms. In Proceedings of the 2019 International Conference on Big Data Engineering, BDE 2019, Hong Kong, China, 11–13 June 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 98–106. [CrossRef]
12. Bartos, V.; Zadnik, M.; Habib, S.M.; Vasilomanolakis, E. Network entity characterization and attack prediction. *Future Gener. Comput. Syst.* **2019**, *97*, 674–686. [CrossRef]
13. Ahsan, M.; Gomes, R.; Chowdhury, M.M.; Nygard, K.E. Enhancing Machine Learning Prediction in Cybersecurity Using Dynamic Feature Selector. *J. Cybersecur. Priv.* **2021**, *1*, 199–218. [CrossRef]
14. Kotenko, I.; Saenko, I.; Branitskiy, A. Detection of Distributed Cyber Attacks Based on Weighted Ensembles of Classifiers and Big Data Processing Architecture. In Proceedings of the IEEE INFOCOM 2019—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, 29 April–2 May 2019; pp. 1–6.
15. Hofer–Schmitz, K.; Kleb, U.; Stojanović, B. The Influences of Feature Sets on the Detection of Advanced Persistent Threats. *Electronics* **2021**, *10*, 704. [CrossRef]
16. Thuraisingham, B.; Kantarcioglu, M.; Hamlen, K.; Khan, L.; Finin, T.; Joshi, A.; Oates, T.; Bertino, E. A data-driven Approach for the Science of Cyber Security: Challenges and Directions. In Proceedings of the 2016 IEEE 17th International Conference on Information Reuse and Integration (IRI), Pittsburgh, PA, USA, 8–30 July 2016; pp. 1–10. [CrossRef]
17. Masud, M.M.; Chen, Q.; Khan, L.; Aggarwal, C.C.; Gao, J.; Han, J.; Srivastava, A.; Oza, N.C. Classification and Adaptive Novel Class Detection of Feature-Evolving Data Streams. *IEEE Trans. Knowl. Data Eng.* **2013**, *25*, 1484–1497. [CrossRef]
18. Ju, A.; Guo, Y.; Ye, Z.; Li, T.; Ma, J. HeteMSD: A Big Data Analytics Framework for Targeted Cyber-Attacks Detection Using Heterogeneous Multisource Data. *Secur. Commun. Netw.* **2019**, 1–9. [CrossRef]
19. Manso, M. SPHINX Architecture v2. 2020. Available online: https://doi.org/10.5281/zenodo.3727777 (accessed on 13 July 2021).
20. Moustakidis, S.; Karlsson, P. A novel feature extraction methodology using Siamese convolutional neural networks for intrusion detection. *Cybersecurity* **2020**, *3*, 1–13. [CrossRef]
21. Babić, I.; Miljković, A.; Čabarkapa, M.; Nikolić, V.; Đorđević, A.; Ranđelović, M.; Ranđelović, D. Triple Modular Redundancy Optimization for Threshold Determination in Intrusion Detection Systems. *Symmetry* **2021**, *13*, 557. [CrossRef]
22. Wagner, C.; Dulaunoy, A.; Wagener, G.; Iklody, A. MISP: The Design and Implementation of a Collaborative Threat Intelligence Sharing Platform. In Proceedings of the 2016 ACM on Workshop on information-sharing and Collaborative Security, New York, NY, USA, 24 October 2016; pp. 49–56. [CrossRef]
23. Tounsi, W.; Rais, H. A survey on technical threat intelligence in the age of sophisticated cyber attacks. *Comput. Secur.* **2018**, *72*, 212–233. [CrossRef]
24. Morris, K. *Infrastructure as Code: Dynamic Systems for the Cloud Age*, 2nd ed.; O'Reilly Media: Newton, MA, USA, 2021.
25. Bullington-McGuire, R.; Dennis, A.K.; Schwartz, M. *Docker for Developers*; Packt Publishing: Birmingham, UK, 2020.
26. Mironeanu, C.; Craus, M.; Butincu, C.N. Intrusion detection using alert prioritization and multiple minimum supports. In Proceedings of the 2015 14th RoEduNet International Conference—Networking in Education and Research (RoEduNet NER), Craiova, Romania, 24–26 September 2015; pp. 109–114. [CrossRef]

27. Mironeanu, C.; Aflori, C. An efficient method in pre-processing phase of mining suspicious web crawlers. In Proceedings of the 2017 21st International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 19–21 October 2017; pp. 272–277. [CrossRef]
28. Farcic, V. *The DevOps 2.1 Toolkit: Docker Swarm*; Packt Publishing: Birmingham, UK, 2017.
29. McGrew, W. *An Attacker Looks at Docker: Approaching Multi-Container Applications*; Technical Report; HORNE Cyber: Memphis, TN, USA, 2018.
30. Douligeris, C.; Raghimi, O.; Lourenço, M.B.; Marinos, L.; Sfakianakis, A.; Doerr, C.; Armin, J.; Riccardi, M.; Wim, M.; Thaker, N.; et al. *ENISA Threat Landscape 2020—Distributed Denial of Service*; Technical Report; European Union Agency for Cybersecurity: Heraklion, Greece, 2020.