Journal of Systems Architecture 131 (2022) 102722

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

# IDERES: Intrusion detection and response system using machine learning and attack graphs

Joseph R. Rose [a], Matthew Swann [a], Konstantinos P. Grammatikakis [b], Ioannis Koufos [b], Gueltoum Bendiab [c,*], Stavros Shiaeles [a], Nicholas Kolokotronis [b]

[a] Cyber Security Research Group, University of Portsmouth, PO1 2UP, Portsmouth, UK
[b] Department of Informatics and Telecommunications, University of the Peloponnese, 22131 Tripolis, Greece
[c] Department of Electronics, University of Frères Mentouri, Constantine 25000, Algeria

ARTICLE INFO

*Keywords:*
Machine learning
Intrusion detection
Intrusion response
Security
Internet of Things

ABSTRACT

The rapid increase in the use of IoT devices brings many benefits to the digital society, ranging from improved efficiency to higher productivity. However, the limited resources and the open nature of these devices make them vulnerable to various cyber threats. This paper explores the potential of using network profiling, machine learning, and game theory, to secure IoT against cyber-attacks. The proposed anomaly-based intrusion detection solution dynamically and actively profiles and monitors all networked devices for the detection of IoT device tampering attempts as well as suspicious network transactions. Any deviation from the defined profile is considered to be an attack and is subject to further analysis. Raw traffic is also passed on to the machine learning classifier for identification of potential attacks. To complement this solution, an intrusion response system is used to act upon the generated alerts and compute the mitigation actions at real-time. Performance assessment of the proposed methodology is conducted on the Cyber-Trust testbed using normal and malicious network traffic. The experimental results show that the proposed anomaly detection system delivers promising results with an overall accuracy of 98.35% and 0.98% of false-positive alarms, resulting in the mitigation of the majority of the executed attacks.

## 1. Introduction

IoT devices are virtually everywhere today and have been increasingly deployed to critical infrastructure sectors such as power grid, healthcare, and industrial control systems. Statistical studies show that the last few years have seen an expected increase in IoT-enabled devices usage with more than 10 billion active IoT devices worldwide in 2021. This number will surpass 25.4 billion in 2030, whereas by 2025 there will be 152,200 IoT devices connecting to the internet per minute [1]. Since IoT-enabled devices can connect to a wider network, new entry points for the network are created introducing new security risks due to having very limited security capabilities [2]. A single compromised device can provide a foothold to internal networks and expose infrastructure to major security breaches, including theft of sensitive information, such as financial records and access credentials. More destructive malware, such as ransomware, can even cause the failure of medical and military equipment, endangering life or allowing breaches of national security, which is a big challenge for the whole public sector and research community [2,3]. These security threats should be identified before causing any type of loss or damage to the organisations. In this context, Intrusion Detection Systems (IDS) are commonly used by organisations to monitor their networks and identify possible malicious activity. However, attackers are continuously developing new attack strategies to obfuscate their attacks and avoid detection. This has been proved by the increasing number of IoT-based attacks reported annually [4,5]. Therefore, developing an effective, efficient and adaptive IDS is an active research area that researchers have been working on for decades. A large number of studies have attempted to design new IDS tailored specifically for the IoT ecosystem needs. However, most of them still need improvements in terms of scalability, detection accuracy, false alarms rate and energy consumption [6]. Moreover, most of the existing solutions are ineffective in detecting unknown and new versions of attacks for which do not exist signatures or predefined patterns, also known as zero-day exploits [6,7].

In addition to an IDS solution, *intrusion prevention systems* (IPS) and *intrusion response systems* (IRS) have been used to enable appropriate

* Corresponding author.
*E-mail addresses:* joseph.rose@port.ac.uk (J.R. Rose), matthew.swann@port.ac.uk (M. Swann), kpgram@uop.gr (K.P. Grammatikakis), ikoufos@uop.gr (I. Koufos), bendiab.kelthoum@umc.edu.dz (G. Bendiab), sshiaeles@ieee.org (S. Shiaeles), nkolok@uop.gr (N. Kolokotronis).

https://doi.org/10.1016/j.sysarc.2022.102722
Received 15 March 2022; Received in revised form 26 July 2022; Accepted 26 August 2022
Available online 2 September 2022

handling of security incidents [8]. Although both IPS and IRS solutions have a common aim, they differ significantly with respect to the timing that a mitigation action is taken, with the former acting immediately upon the detection of an attack and the latter acting with some time delay due to also considering the effectiveness of mitigation actions [9]; this allows the IRS solutions to achieve in principle better results. Various designs have been proposed for IRS solutions in the literature [8,9]. An important aspect of such a design is the threat modelling methodology adopted, which greatly impacts the efficiency of the mitigation actions generated by an IRS solution: the more complete and precise the threat models are, the better an IRS responds to a (modelled) attack. The representation of threats, network or host related, requires a manageable data structure that is compact enough and further allows efficient mitigation action generation algorithms. *Attack trees* (AT) were an early attempt to such threat modelling [10], where a tree structure is used for organising the possible attack paths towards some target system (i.e. an attacker's goal) that constitutes the tree's root node. Since ATs cannot accurately represent the inherent complexity of realistic IoT networks and their large number of threats, the use of *attack graphs* (AG) was suggested in the literature that are better suited for large-scale networks [11]. Despite the improved modelling of threats through AGs, the performance of an IRS is greatly impacted by the size of the graph, the sharing of meaningful and accurate information by other network monitoring systems, and the design of efficient mitigation action generation algorithms capable of leveraging such information. These are key challenges for IoT networks that have to be addressed by IRS solutions.

To overcome the above issues, this paper presents a novel *intrusion detection and response system* (IDERES) based on network traffic profiling, machine learning and attack graphs. Building upon our prior work [12], we adopt a tight coupling of the IDS and the IRS designs so that the overall solution allows not only to accurately detect known and unknown attacks, but also to efficiently respond to sophisticated multi-stage attacks. The new proposed system involves three primary components, i.e. the profiling service, the IDS and the IRS, for the detection and prevention of potential known and unknown attacks at the network level as well as on a per device basis.

*(a) Profiling service.* This component is used for performing network profiling and vulnerability assessment / identification of systems that are located in the *local area network* (LAN). It has two main functionalities. First, it automatically scans all the connected devices in the LAN for potential common vulnerabilities and running services. Second, it calculates out of bound network profile behaviours by continuously monitoring the network traffic flows from each device. Every deviation from the defined profile is considered a potential attack and is subject to further analysis.

*(b) Intrusion detection system.* The IDS uses efficient binary visualisation techniques employed by the ML-based malware detection alongside Suricata's network signature-based detection. The binary visualisation method uses the clustering algorithm space-filling curve to ensure that close data are grouped together, which helps to create much more appropriate RGB images for the classification process. Alerts generated by Suricata are sent alongside predictive alerts of malicious network patterns which are predicted by our ML module. The ML module is an extension of our approach proposed in [13] by using more data for training the model. In addition, the testing of the ML module is done by real-time (i.e. not simulated) attacks.

*(c) Intrusion response system.* This component monitors the security status of a network through the lens of a probabilistic AG, which is a special case of a *graph-based network security model* (GNSM), by leveraging information shared by the IDS. Its tight coupling with the IDS allows exploiting alerts being generated by either intrusion detection engine (rule-based and ML-based) and therefore being capable of also responding to unknown attack patterns, like zero-day attacks, with excellent performance results in terms of successful mitigation actions. The IRS decision-making algorithm generating the mitigation actions is well-suited for IoT networks since it monitors and assesses an attacker's behaviour to intelligently respond (using a game-theoretic approach) at real time in an optimal manner by balancing between security and network availability.

The performance evaluation of the proposed techniques is carried out using our training and testing datasets. The testing data is created using the Cyber-Trust [14] testing network, whereas, the training dataset is created from both normal and malicious traffic, from pre-existing public datasets. The datasets are publicly available on the IEEE DataPort website [15]. These datasets contain both the packet captures of the network attack scenarios carried out to test IDS and profiling system as well as the images used to train the machine learning module itself. The experimental results show that the proposed IDS framework can achieve high accuracy (98.35%) with low false alarm rates (0.98%). The conducted experiments also proved the capability of our system in detecting malicious traffic generated from zero-day attacks.

The paper is organised as follows: Section 2 discusses recent works in the domain. Section 3 presents the proposed IDS methodology and the overall components of the iIRS system. This section also describes the created network traffic dataset on the Cyber-Trust testbed. Experimental design and results are discussed in Section 4. Finally, Section 5 concludes the paper along with some future directions.

## 2. Related work about ML-based intrusion detection and response systems

Research in the intrusion detection in the IoT ecosystem highlights the need and the importance of effective IDSs. To enhance their accuracy and reduce false positives, most IDSs rely on ML/DL techniques, as they are suitable for detecting even unknown attacks like zero-day exploitations. Different kinds of graph-based models are introduced as way to describe and analyse complicated multi-vector cyber attacks. Related work highlights their usage with game theoretic intrusion response systems capable of utilising IDS alerts. This section will take a schematic review of the recent developments of intrusion detection and response solutions in IoT.

In recent years, several studies have attempted to design new intrusion detection systems tailored specifically for the IoT ecosystem [6,27–29]. In [6], authors proposed a literature review of existing intrusion detection solutions in IoT environments (over 40 studies). The study proposed a classification of the existing solutions based on different features like the intrusion detection techniques, location of the IDS in the network, evaluation techniques and type of attacks. The study concluded that existing intrusion detection solutions for IoT networks still need improvements in terms of scalability, detection accuracy, true positive rate and energy consumption. In face of the limitations of signature-based techniques, ML/DL techniques have recently received considerable attention for their ability to accurately detect intrusions and therefore reduce the false positive by detecting unknown or modified attacks. In [30], authors surveyed intelligent NIDS systems based on ML/DL models (supervised, unsupervised, and semi-supervised learning). The authors concluded that proposing an efficient NIDS framework using DL algorithms is a potential future scope of research in IoT. In this context, authors in [16] proposed an intelligent NIDS model using a non-symmetric deep Autoencoder (AE) and Random Forest (RF) classifier. The complexity of the model is reduced by using non-symmetric deep AE for efficient feature selection. However, the model is evaluated using old datasets: KDD99 and NSL-KDD. In the same context, authors in [7] investigated the effectiveness of different ML algorithms in securing IoT devices against DoS/DDoS attacks. This study tried to suggest appropriate methods for developing IDS systems using ensemble learning for IoT applications. The assessed classifiers are Random forest (RF), AdaBoost (AB), Extreme gradient boosting (XGB), Gradient boosted machine (GBM), and Extremely randomised trees (ETC).

**Table 1**
Overview of recent ML/DL-based intrusion detection systems for IoT.

| Authors | Year | Dataset | ML/DL algorithms | Accuracy (%) |
|---|---|---|---|---|
| M. H. Ali et al [16] | 2018 | KDD99, NSL-KDD | Deep AE, RF | 99.60% |
| D. Preuveneers et al [17] | 2018 | NSL-KDD dataset | Federated Learning | 97.00% |
| M. Ge et al [18] | 2019 | BoT-IoT dataset | FNN | 99.90% |
| H. A. Moreton et al [19] | 2019 | Personal Dataset | LSTM, GRU | 96.08% |
| Ch. Luo et al [7] | 2020 | CSIC 2010 | LSTM (RALM), RNN | 98.77% |
| M. A. Ferrag et al [20] | 2020 | Bot-IoT dataset | DNN, RNN, CNN | 98.37% |
| I. Idrissi et al [21] | 2021 | Bot-IoT dataset | Deep CNN | 99.94% |
| M. Zhong et al [22] | 2021 | ADFA-LD, KDD99 | GRU, Text-CNN | 98.00% |
| I. Ullah et al [23] | 2021 | BoT-IoT, IoT Network Intrusion, MQTT-IoT-IDS2020, IoT-23 | 1D, 2D and 3D CNN | 99.79% |
| Y. Otoum et al [24] | 2022 | NSL-KDD | SDPN, SMO | 99.02% |
| Y. Saheed et al [25] | 2022 | UNSW-NB15 | XGBoost, KNN, SVM, QDA,NB | 99.90% |
| T. laSaba et al [26] | 2022 | BoT-IoT, Network intrusion detection (NID) | CNN | 99.85% |

In another work [3], authors proposed a technique based on the Online Sequential Extreme Learning Machine (OS-ELM) for intrusion detection. This work used the alpha and beta profiling methods to reduce the time complexity when irrelevant features are discarded from the training dataset. The evaluation of this solution is performed with the NSL-KDD 2009 (Network Security Laboratory-Knowledge Discovery and Data Mining) dataset. This solution achieved an accuracy rate of 98.66%, with a false positive rate of 1.74% and detection time of 2.43 s. In order to solve the issues related to the data being widely spread across large networks of connected devices, authors in [17] proposed a decentralised IDS by using a Federated Learning (FL) based scheme. In this scheme, the training is performed locally. Also, devices can benefit from their peers' knowledge by communicating only their updates with a remote server that aggregates the latter and shares an improved detection model with participating devices. The evaluation of this desentalised IDS is done by the NSL-KDD dataset and achieved an accuracy of 97%

In more recent work [18], authors developed a DL-based intrusion detection approach including both binary and multi-class classifications for IoT networks. The proposed framework achieved a high accuracy (99.90%) with the feed-forward neural networks (FNN) model. Authors in [31] developed a network intrusion forensics system based on a transductive scheme that can detect and analyse efficiently computer crime in networked environments and extract digital evidence automatically. TCM-KNN (Transdative Confidence Machines for K-Nearest Neighbours) is a novel algorithm combing TCM and KNN algorithm. Experiment results showed that this approach can provide comprehensible aid for a forensic expert. In [21], the Convolutional Neural Network (CNN) has been implemented and tested against some well-known Botnet attacks using a specific Bot-IoT dataset. The proposed IDS obtained promising results with 99.94% in validation accuracy, 0.58% in validation loss, and a prediction execution time less than 0.34 ms. CNN has also been used in [23] to develop a novel anomaly-based intrusion detection model for IoT networks. The proposed CNN model is validated using the BoT-IoT, IoT Network Intrusion, MQTT-IoT-IDS2020, and IoT-23 intrusion detection datasets. The minimum detection rate for binary classification was 99.79% for the Theft class. Similarly, CNN has been implemented in [26] and tested using the BoT-IoT and the Network intrusion detection (NID) datasets. In this work, the proposed DL-based IDS achieved the best accuracy (99.85%) with the BoT-IoT dataset. Authors in [24], combined the spider monkey optimization (SMO) algorithm and the stacked-deep polynomial network (SDPN) to create a new deep learning–based IDS (DL-IDS) to detect security threats in IoT environments. The proposed DL-IDS has been evaluated using the NSL-KDD data set, and it achieved an accuracy rate of 99.02%, precision (99.38%), recall (98.91%), and F1-score (99.14%). In [25], a ML-based IDS for IoT is proposed to improve the security and privacy challenges of IoT. The proposed IDS has been tested with the on the UNSW-NB15 dataset and different ML algorithms including Cat boost, KNN, SVM, NB and XGBoost. The high accuracy (99.9%) is achieved with the XGBoost ML algorithm.

Work in [32] focused on the application of specific framework models to integrate machine learning components for flow-based traffic monitoring. This work placed the machine learning SDN controller at the centre of each network control transaction and used decision-making logic to spot inconsistencies or abnormalities to the traffic flows that pass through the SDN. This anomaly-based model is placed on top of the SDN and works in real-time to detect abnormalities. Notably missing from this model is further identification or informed action, just the detection of anomalies. This feeds into the detection rates, depending on the model used, of up to 95.16% with a false positive rate over 2.49%. Work in [33] explored contemporary research and literature to compare the speed, accuracy and false-positive rates of machine learning components integrated into traditional and fully machine-learning-based IDSs. The main objective of this comparison is to check the effectiveness of different training sets for anomaly detection. The authors noted that the speed and accuracy of most current solutions are dependants on the pre-training data, as is the case with most supervised machine learning systems. However, the study noted that the data set that has been used for the majority of these systems is imbalanced and introduces limiting factors into IDS systems based on it. This is since it does not determine how accurate background traffic is to the malicious traffic for the data set. Moreover, KDD Cup 99' dataset is considered old and obsolete, thus another purpose of this paper expect of proposing a new system is to provide the community with some recent datasets of malicious traffic which can be downloaded from [15]. Table 1 presents examples of some recent studies that applied ML/DL techniques to NIDS.

In addition to intrusion detection, intrusion response systems have been the subject of intense research and many constructions have been proposed to allow mitigating known and unknown threats in a both proactive and reactive manner. Work in [34] states that the protection of critical networks is a particularly difficult task and requires the proper understanding of not only the systems themselves but also their vulnerabilities along with their interdependencies. It is also explained that apart from the detection process that causes an event, the event itself must be analysed for optimal intrusion response to be provided. In the past, graph-based models were shown to provide interesting results by analysing an attack in multiple steps while correlating vulnerability exploitations with an attacker's ultimate goal. In particular, AGs, were initially presented in [35], where the authors created the first automated and exhaustive graph-based model of its kind that was generated by taking as an input a set of network states and their corresponding transition relations. *Attack paths* in Sheyner's AGs are called atomic attacks and can be interpreted as series of exploits. *Logical attack graphs* (LAG) further captured the interest or researchers, illustrating logical dependencies among the attack goals of the attacker by utilising reasoning methods to produce a directed graph-based model [11]. Being more descriptive than regular AGs, LAGs implement multiple type of nodes, so that an exploitation is described by its causes and its corresponding outcome in the network topology. To enhance the scalability of AGs, *Bayesian attack graphs* (BAG) were shown to be a

convenient solution to enumerate all potential attack paths by building more compact models (in comparison to AGs) and analyse exploitations which is achieved by encoding security conditions and vulnerabilities into Bayesian belief networks [36,37]. BAGs further provide a template for applying risk management methods and specifically risk and impact analysis algorithms [38]. While the generation of large scale *Attack Graphs* have been addressed by recent works [37,39], the handling of such models is not considered a trivial task [40]. Furthermore, the authors in [41] present a tool that utilises *Attack Graphs* as a basis for risk mitigation, prioritising detected vulnerabilities and analysing their impact on large-scale network topologies, exhaustively considering all possible sub-attack paths.

Research interest has also been shown in the integration of IDSs with more complex *intrusion response systems* (IRS) to facilitate the calculation of optimal responses to ongoing cyberattacks. A comparative study on the design of IRSs was performed by [9] in which the tight coupling between the two was highlighted, and three approaches to alert-response mapping were discussed: static, dynamic, and cost-sensitive—the last derived from a process balancing intrusion damage to the side effects of the response. A survey of game theoretic approaches in the area of intrusion detection and response was conducted by [42], highlighting that stochastic games are more realistic than static games, under the assumption that the current state of the network is known with certainty; that players are often assumed to exhibit perfect rationality, i.e. having complete knowledge of their actions' costs and benefits; and that real-world evaluation of game parameters is often out of the scope of most works. An IRS based on attack-response trees and a sequential attacker-defender Stackelberg stochastic game was presented by [43]; a case study on the protection of a *supervisory control and data acquisition* (SCADA) network was presented, modelling three possible defender actions: to reset & restart a controller, to switch off the network's human-machine interface, and no operation. A cost-sensitive approach was also taken by [44], presenting an IRS with the ability to choose between 24 actions, including firewall rules and service/host actions, whose cost is evaluated by 10 criteria. Their implementation used Snort IDS alerts matched to an attack graph using CVE IDs, and was tested on a large network consisting of a DMZ and 5 subnetworks. To model the problem of network security defense, a *partially observable Markov decision process* (POMDP) was presented in [45]; in it a condition dependency graph was used to represent the network state, with an attacker aiming to reach a desirable state by exploiting vulnerabilities and a defender trying to stop this process in the most optimal way possible by blocking exploit nodes. To combat complexity issues regarding the state space of the Markov process, a problem also noted by [46], a *partially observable Monte-Carlo planning* (POMCP) algorithm was used to calculate the most suitable defense policy without exhausting all possibilities. A similar approach was also taken by [47], in which a small-scale graphical representation of a network with three nodes (representing three network elements) and eight exploits was used to test the effectiveness of a POMDP. Their approach to the state space problem involved a modified recursive depth-first search algorithm to discover feasible paths while also pruning unattainable states. Two behavioural models of attackers were used: one incapable of exploiting blocked exploit nodes (inexperienced), and one capable, but with a certain probability of success (experienced). It was noted that the actions of experienced attackers caused the mean cost of defensive actions to be much higher, compared to inexperienced ones; this was attributed to the need to isolate entire parts of the network to thwart them.

## 3. System architecture

In this section, we present a high-level architecture of the proposed framework for intrusion detection and mitigation in IoT-based network. The proposed framework is composed of three main components: the network profiling service, the intrusion detection system and intrusion response system.

As illustrated in the right upper half of Fig. 1, the network profiling component is located in the IoT gateway for data collection and communication. Given the additional computational requirements, this component may be relocated on a separate hardware device but closely connected to the gateway (bridged with the gateway). This allows the component to access and profile the LAN that the gateway resides on, as well as inter-device connections. The first detection sub-component of the second component resides in the getaway and is running a light version of the Suricata IDS (without machine learning) and IoT specific signatures (modified version of snort rules) to monitor packets on the local network and detect known malicious threats. Whereas, the ML IDS sub-component is deployed at the Internet Service Provider (ISP) Cloud as it requires more resources due to the ML operations. The main objective of the IDS is the detection of potential known and unknown cyber-attacks on the networks level as well as at the device levels. Detected attacks from both signature-based IDS and ML/IDS are further forwarded to the iIRS in the form of event alerts.

A high-level view of the *intelligent intrusion detection system* (iIRS), showcasing both of its major sub-components and their interaction with the proposed IDS, is presented in the left half of Fig. 1. The first sub-component, the *iIRS attack graph generator* (iRG), is primarily responsible for the generation of the *graph-based network security model* (GNSM) according to topology information received from the network profiling module and for performing dynamic risk analysis. The second major sub-component of the iIRS, is the *iIRS mitigation engine* (iRE) which collects the topology model from the iRG and optimally selects mitigation actions upon reception of event alerts from both Suricata and the ML/IDS. All possible mitigation actions are generated in the iRG component and are then sent to the iRE to form its mitigation repertoire. Similar to the IDS, the iIRS can also be relocated on separate hardware outside of the IoT network, as long as it remains connected to the other gateway components, due to the inability of most network equipment to meet its computational requirements.

### 3.1. Network profiling component

The behaviour of the network profiling component is divided into two distinct parts, firstly is the functionality to automatically scan connected devices on the locally available network for potential common vulnerabilities and currently running services. The vulnerability information is primarily sourced from a routinely updated listing from the publicly available CVE Mitre database [48]. These common vulnerabilities are mapped to the available network services which in turn are discovered through network port scanning techniques such as those provided by Nmap. Once the list of potential vulnerabilities of each device is collated, each device is profiled utilising information relating to each device. This includes routing information, the reported hostname, network flow and topology; this information is then provided to external components to digest this information through a REST API framework. The second main characteristic of the network profiling services is the ability to calculate out of bound network profile behaviour, this is calculated by the continual monitoring of the network traffic flow from each device across the network. It utilises rate informed heuristic profiling to create an expected throughput pattern for each device on the LAN that it is connected to. This profile is then compared against three different predefined profiles:

(1) *Hourly Profile (H)* - A profile of the network that is informed by a packet capture that is refreshed hourly.
(2) *Daily Profile (D)* - A profile of the network that is informed by a packet capture that is refreshed daily
(3) *Weekly Profile (W)* - A profile of the network that is informed by a packet capture that is refreshed weekly

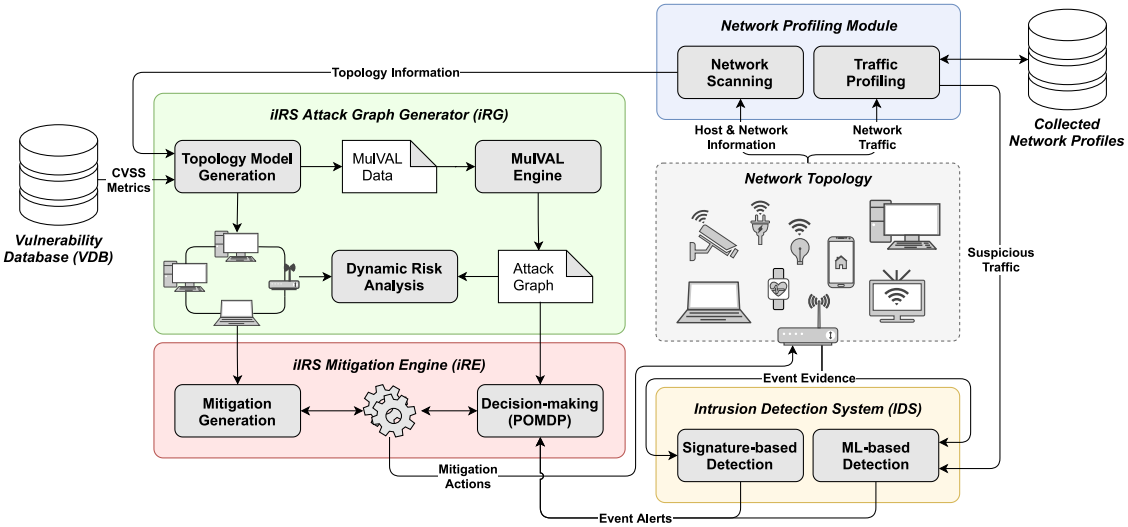



**Fig. 1.** High-level architecture of the proposed system.

The objective of utilising different profiles separated and refreshed by period is to provide a more accurate map of the network conditions that a device would experience over time, this increases profile accuracy and makes the system more adaptable to variable network conditions and varied device usage. The rate metric for these captures is calculated via

$$RM = \frac{n}{t} \tag{1}$$

where $n$ is the total number of bytes transmitted and $t$ is the time of capture. The component is then able to take periodic network captures of the LAN traffic from the gateway, this new capture is then run through the same profiling system as the timed profile captures, and a new rate metric is calculated. A percentage difference is calculated, comparing the rate profile of the new capture to each timed profile:

$$\Delta = \frac{CRM}{PRM} \times 100 \tag{2}$$

where $\Delta$ is the percentage difference between CRM, the calculated rate metric and PRM, the profile rate metric. If the delta value passes over a threshold value that can be configured per implementation depending on network volatility, the network activity of the device is flagged as out of profile and a re-scan of the network is initiated to re-scan for any possible actively exploited attack surface on the network. This process is fast, but minimal in terms of network impact and will not degrade network performance, even on a small network as the scale of the scan will increase or decrease in intensity automatically depending on scan timings and throughput. This threshold can be raised or lowered depending on the network's volatility. If scanning is to frequent, the threshold can be increased on a busy, variable load network for example. The traffic capture, stored in PCAP format, that the network profiling component uses to calculate and profile each device can then be transferred to the machine learning component to check the traffic for patterns that could indicate malicious traffic, including active attacks or ongoing exploitation. This can then be used to inform mitigation actions across the affected network.

### 3.2. IDS system

The intrusion detection system uses both signature-based and anomaly-based detection techniques for the detection of potential known and unknown cyber security threats (see Fig. 1). The anomaly-based detection technique is based on the Malware-Squid approach proposed in [13]. In this approach, the machine learning module makes use of the Hilbert space-filling curve as its main clustering algorithm,

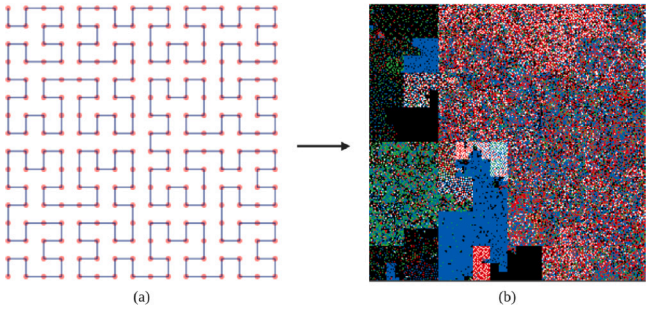



**Fig. 2.** (a) The Hilbert space-filling curve mapping and (b) the output 2D image. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

this is achieved by assigning specific colours to each byte as it is converted into a 2D image. This conversion is performed on each byte depending on its ASCII character reference (see Fig. 2).

- **blue** for printable characters
- **green** for control characters
- **red** for extended characters
- **black** for the null character, or 0 × 00
- **white** for the non-breaking space, or 0xFF

These generated byte arrays are then processed using the Hilbert algorithm, transforming them into images that retain optimal locality for pattern recognition (see Fig. 2), allowing them to be processed by machine learning image classification models [5], where we used samples of malicious network traffic captures to train a trained neural network classifier. For performance reasons, multiple packets chunks are created and forward to the visual representation tool to convert them into a 2D image. Then, the trained neural network classifier is used to analysis and classify the output images as legitimate or malware. In this context, deep learning neural networks, especially, the convolutional neural network (CNN) are ideal for processing 2D images and achieved promising results in image recognition in many areas such as healthcare, and security. Currently, there are many architectures of CNN that have been developed by research groups from well known companies such as google, Microsoft, and Facebook. All these architectures have demonstrated that CNN is one of the best learning algorithms for understanding and analysing image content that has shown high

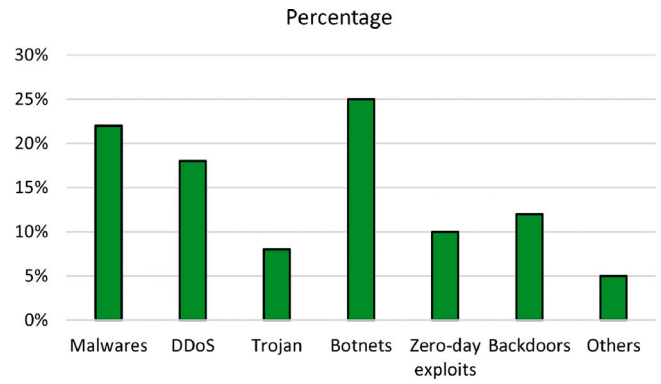



**Fig. 3.** Malicious traffic sample percentage according to type of malware.

performance in image segmentation, classification, detection, and retrieval related tasks. Specifically, we have implemented our ML-based IDS component with the Deep CNN MobileNetV3. MobileNets are small, low-latency, low-power models parameterised to meet the resource constraints of a variety of use cases. MobileNetV3 is the third version of the architecture. The main contribution of MobileNetV3 is the use of AutoML to find the best possible neural network architecture for a given problem. MobileNetV3 first searches for a coarse architecture using MnasNet, which uses reinforcement learning to select the optimal configuration from a discrete set of choices. After that, the model fine-tunes the architecture using NetAdapt, a complementary technique that trims under-utilised activation channels in small decrements. It extends MobileNetV2 by incorporating squeeze-and-excitation blocks as part of the search space which ended up yielding more robust architectures.

### 3.3. Data collection and processing

Before the tests are performed, the ML module was initially conditioned by a data collection of normal and malicious PCAP data. However, the overall process of the machine learning algorithm training is carried out incrementally, each time new malware traffic is found, that is, without ignoring the information identified in earlier training phases. This would greatly improve the detection accuracy of the machine learning module. The initial dataset created for the initial training of the learning module consists of 913 samples of both normal and malicious, these are in binary visualisation format (2D images of size 1024 × 256). The dataset is publicly available on the open-access IEEE DataPort website [15]. Normal samples contain real network traffic that was collected from various clean IoT devices using tools such as NMAP and Wireshark. Malicious samples were collected from different public sources of malware PCAP files and contain real malicious traffic that was generated by different types of attacks such as malware, DDoS, Key loggers, backdoors, OS scans, etc. The dataset has an approximate size of 100 MB. Fig. 3 shows the percentage of malicious traffic samples in the training dataset.

To test our framework, we have created our collection of PCAP files provided by real malware traffic in the Cyber-Trust testbed. More precisely, malicious PCAP files were created from different real-world attack scenarios, including the Mirai Botnet [49], Blackenergy Botnet [50], Zeus Botnet [51], as well as attack replay scenario which consisted of several attack types such as an Java-RMI Backdoor [52], distcc_exec backdoor [53], Web Tomcat Exploit [54] and Hydra Brute-force attacks [55] to name a few. Full list is available in Table 2. The most important attack scenario consisted of a simulated zero-day attack whereby network signatures relating to the VSFTDP [56] attack scenario were removed from the Suricata IDS and thus measured by alerts solely generated from the machine learning module. The PCAP files were generated by running live demos of each attack scenario and recording inter-device network communication using tcpdump [57]. While normal network traffic file captures have been obtained from uninfected device traffic by conducting regular network-intensive activities, such as copying both text and media files across the network, as well as running SSH sessions, streaming media content, and API contact, all of which may be assumed to be contained in a normal smart home network. Once these PCAP files have been stored, they can be replayed by the respective devices using TCPreplay [58] to re-transmit the captured packets across the network in real-time. This helps us to track and document the responses of the IDS systems to the actual attack data and to measure the efficacy of the detection operation for each attack scenario.

### 3.4. iIRS attack graph generator (iRG)

Regarding the testing of this work, the proposed IDS was further utilised along with an experimental *intrusion response system* (IRS). The first half of the IRS is the *iIRS attack graph generator* (iRG) which operates similarly to the tool presented in [37]. A GNSM is created based on the *Multi-host, Multi-stage Vulnerability Analysis language* (MulVAL) tool [59] using vulnerability scanner reports (e.g., OpenVAS & Nessus) and network-related information about the LAN as input to generate the LAG. To accommodate these needs, the network profiling module provides information about available network services accompanied with their potential common vulnerabilities. Furthermore, the topology is provided as well, including all the profiled information which is gathered upon network discovery, as described in Section 3.1. Specifically, the gathered information is as follows:

- The network IP ranges for the network segments covered by the iIRS in *classless inter-domain routing* (CIDR) notation.
- Network hosts: interface names, IP addresses and their corresponding association to a LAN/WAN.
- Network structure: subnetworks, IP addresses, network mask and address of the gateway.
- Allowed interactions between hosts: source and destination IP & ports along with the related transport protocol.
- Whitelisted network interactions: hostname of the involved host, IP address, interface name, destination network IP ad-dress and the network mask.

The aforementioned input is processed by the data extraction subsystem from [37], allowing the topology generated information to be loaded in the XSB Reasoning Engine [60] as Datalog tuples (a subset of Prolog) to generate the attack graph. The LAG is then used to model complex networks, analysing interdependencies between vulnerabilities and representing the actions of an attacker pivoting between the available target hosts until a specific goal state is reached. MulVAL utilises interaction rules that enclose exploits with preconditions that must be met and post-conditions that imply the result of that exploitation. Specifically, acquired information from the network profiling module is used to model all the fundamental aspects of the exploitation prerequisites (network service info, vulnerabilities, VLANs, etc.) which are represented as LEAF nodes in the directed LAG. Moreover, the exploitation that occurs from the combination of different LEAF nodes, is represented as an AND node and leads to a number of post-conditions. Fig. 4 shows a LAG generated for the SOHO environment described in Section 4.1, the three node types are represented with a different node shape: AND nodes with ellipses, LEAF nodes with rectangles, and OR nodes with rhombuses. Additionally, all the generated post-conditions (excluding goal conditions) are further used as preconditions for, previously inaccessible by the attacker, vulnerability exploitations. The directed edges in the graph portray the attacker's transitions from preconditions to post-conditions.

The directed LAG decomposition in LEAF, AND & OR nodes allows the implementation of probabilistic risk analysis methods on the GNSM; this is achieved by handling the LAG as a BAG [36]. The foundation

of the risk analysis methodology that is utilised is based on [38]. Vulnerabilities are identified as potential threat sources in the BAG and are assigned a probability value by taking into account their related *common vulnerability scoring system* (CVSS) metrics which are retrieved from the *vulnerability database* (VDB). All available vulnerabilities are gathered from the *National Vulnerability Database*[1] and for compatibility reasons the iRG uses CVSS version 3.1 metrics whenever they are available and CVSS version 2 metrics otherwise. For the probabilistic risk analysis to take place, all edges that have a LEAF node as a source are assigned with a probability value and the risk assessment algorithm operates by calculating and assigning unconditional probabilities in a bottom-up manner. When multiple exploitations or security preconditions are involved, the algorithm computes local condition probabilities by using the product rule of probability or the noisy-OR operator (as defined in [61]) respectively.

### 3.5. iIRS mitigation engine (iRE)

The second half of the experimental IRS, the *iIRS mitigation Engine* (iRE), is responsible to reactively respond to sophisticated multi-stage network attacks, while balancing the effectiveness of the chosen actions with host and service availability. To make the optimal choice, and possibly anticipate the attacker's future actions, a belief about the network state is formed and updated constantly based on newly received information.

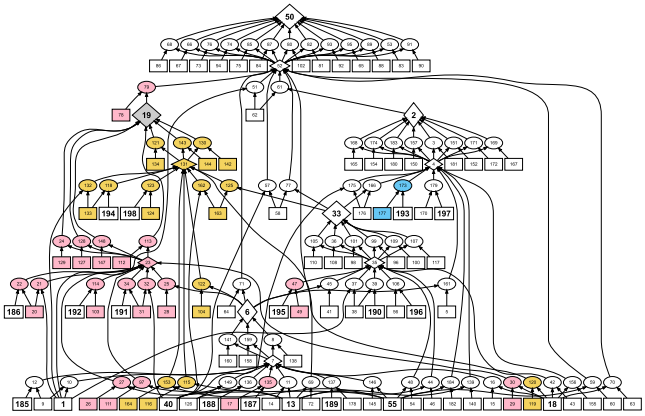
#### 3.5.1. Game-theoretic model

The main algorithm of the iRE is the discrete-time *partially observable Markov decision process* (POMDP) presented in [45], a game theoretic approach defining two players: an *attacker* trying to reach a goal condition by exploiting system vulnerabilities to move through the network; and a *defender* trying to prevent the attacker's progression by employing mitigation actions. The basis of the game is the LAG generated by the iRG, which is viewed as having two sets of nodes: one representing possible states of network devices, that is, all OR & LEAF nodes of the LAG; and one representing conjunctive relations between them—termed as *security conditions* and *exploits* respectively. Directed edges connect precondition states to post-condition states through a related exploit, thus representing the relation of an action's requirements to its results. Security conditions are considered to be either possessed by an attacker (enabled) or not, while two assumptions about the attacker's behaviour are made: exploits may be attempted only if the attacker has something to gain from them (i.e. if their post-conditions are disabled); and capabilities possessed by an attacker cannot be taken away—the monotonicity assumption [62]. In addition, some security conditions are deemed to be highly desirable by the attacker, these are termed *goal conditions* and are considered as possible final goals of an attack—the iRE only considers nodes representing the ability to execute arbitrary code on a device with root privileges.

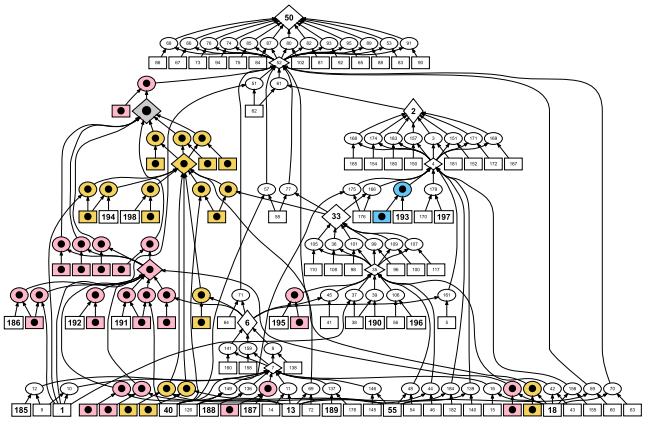
#### 3.5.2. Belief calculation

Both players are uncertain about the true state of the network: the attacker is uncertain whether exploits have already been blocked by the defender, while the defender is not aware of the attacker's current capabilities (i.e. which security conditions are enabled), true strategy, and final goal. Although, by definition, it is not possible for the defender to be certain of the attacker's true intentions and goals, three sources of information can be used to estimate the current network state and the attacker's strategy:

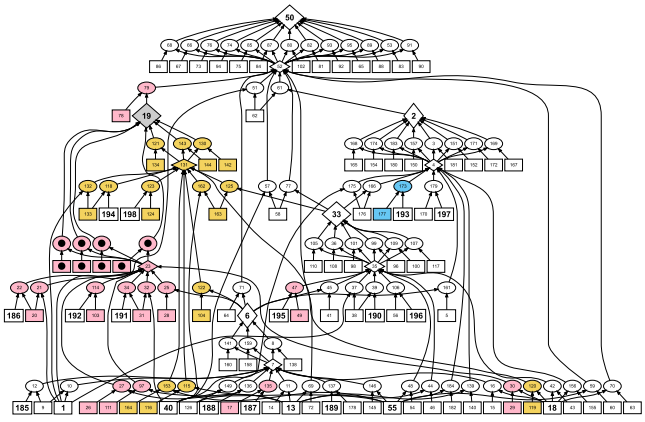
- The LAG describes all network hosts, their vulnerabilities, and their connectivity.



(a) Vertices associated with the host.



(b) Global firewall rule blocking all services.



(c) Firewall rule targeting port 22 (SSH).

**Fig. 4.** Views of the attack graph that relate to a vulnerable Linux host of the SOHO. The three types of nodes are represented as: ellipses for AND nodes, rhombuses for OR nodes, and rectangles for LEAF nodes.

- Network event alerts identify whether any and which host or service is under attack, and indicate, with some degree of uncertainty, the current actions of the attacker.
- Current and previous mitigation actions provide information on the changes made to the network topology, which in conjunction with the observed response can provide further insight on the attacker's strategy.

From these sources, the defender calculates and maintains a belief about the state of the network, expressed as the set of probabilities that

[1] `nvd.nist.gov`.

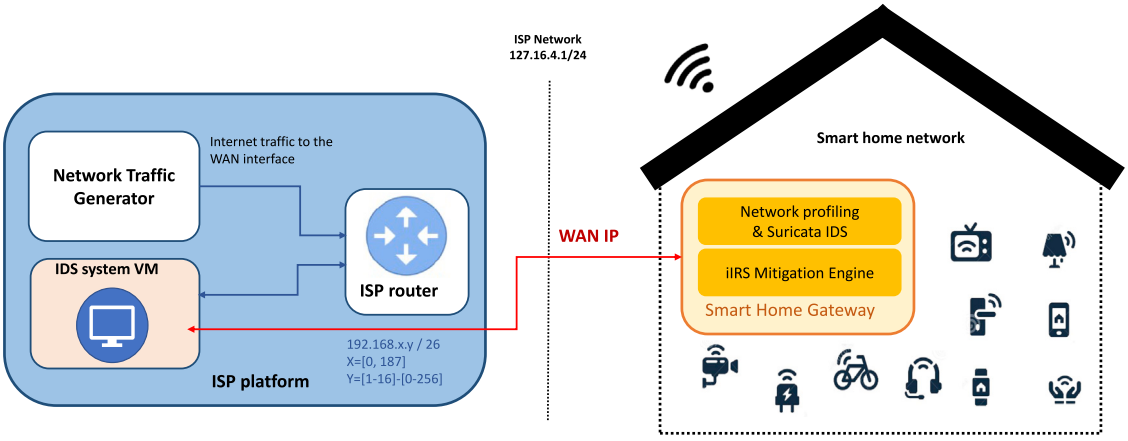



**Fig. 5.** Implemented testbed.

each security condition has been enabled, and a belief of which pre-defined behavioural profile (defining parameters for the probabilistic terms of the model) matches the attacker's observed actions.

### 3.5.3. IDS alert matching

Most attack graph nodes (security conditions and exploits) are associated with their respective hosts to allow network alert matching. Three matching levels are applied in cascading order, from most to least specific, based on the information extracted from the received alerts:

- *Partial info*: matching a specific IP address, port number, and transport protocol.
- *Limited info*: matching everything from the previous level, but for all transport protocols.
- *IP only*: matching only a specific IP address.

For example, Fig. 4(a) shows the nodes associated with one of the vulnerable Linux hosts of the SOHO. The following three sets of nodes match alerts at the partial info level: for the `OpenSSH`[2] service shaded in pink; for the Python `BaseHTTPServer`[3] shaded in yellow; and for the `Tornado`[4] web server shaded in sky blue. All three sets combined with the sole grey node form the set of nodes matched at the IP only level. This grey node represents the goal condition associated with the host, a post-condition of the two exploitable services which cannot be matched at any more specific level.

### 3.5.4. Defensive actions

The only action available to the defender, in the current iRE version, is the deployment of `iptables` firewall rules to effect changes to host interconnectivity, and thus block a number of exploit nodes of the LAG, by either blocking communication between two hosts over a specific port and transport protocol (specific rule) or by completely isolating a host from the network (global rule). These rules are generated by the iRG using the process detailed in [63]. For each exploit node to be disabled, a tree structure is created by this process containing two types of nodes: those describing inter-host access relations and those representing the logical operators of conjunction and disjunction, from OR & AND nodes respectively. The resulting solutions for the LAG are then extracted by collapsing redundant operations, resulting in a tree representing multiple sets of firewall rules effectively blocking the exploit node in a *disjunctive normal form* (DNF):

$$(R_1 \wedge \cdots \wedge R_k) \vee (R_1 \wedge \cdots \wedge R_n) \vee \cdots \vee (R_1 \wedge \cdots \wedge R_m)$$

where each $R_i$ term represents an `iptables` firewall rule set. Each solution is also associated with a set of nodes affected by its application, calculated by removing the targeted exploit node and any unconnected parent nodes from the LAG. Continuing with the previous example: Fig. 4(b) shows the nodes affected by the application of the following global rule:

```
iptables -A [INPUT, OUTPUT] -s 192.168.0.21 -j DROP
```

which targets the host's goal condition and forbids an attacker from exploiting either vulnerable service; while Fig. 4(c) shows the nodes affected by the application of seven rule pairs, one per network host, blocking access to the vulnerable `OpenSSH` service from the entire network—one such rule is:

```
iptables -I [INPUT, OUTPUT] -s 192.168.0.9/32
         -d 192.168.0.21/32 -p TCP --dport 22:22 -j DROP
```

As can be seen from Fig. 4(c), only four exploit parent nodes of the goal condition, along with their security conditions, are affected. These refer to CVE-2018-20685, CVE-2019-6109, CVE-2019-6110, and CVE-2019-6111.

## 4. System implementation & testing

### 4.1. Experimental setup

The experiments were performed in a virtualised smart home (SH) environment in the Cyber-Trust testbed [14]. The SH environment has been implemented by using several virtualised devices, divided into small groups where a separate Ubuntu VM is acting as the gateway for the SM, as depicted in Fig. 5. This VM incorporates the network profiling service and a light version of the Suricata IDS (i.e., without the ML module). Conceptually, this component may reside on the SH gateway for data collection and communication or given the additional computational requirements, it may be relocated on a separate hardware device but closely connected to the smart gateway. This allows the component to access and profile the LAN network that the gateway resides on, as well as the inter-device traffic as referenced in Section 3. The network profiling component is dockerised to allow for ease of deployment and logging. This comes with the added benefits in the form of container resource optimisation and improved execution speed, on a load dependant basis. Whereas, the IDS system is deployed as a single VM at the ISP level, running the Suricata IDS and the ML module based on Debian GNU/Linux 10.2.

The two components of the experimental IRS are deployed on two separate Ubuntu 18.04 LTS VMs inside the SOHO network, ignored by the network profiling module and excluded by the generated LAG.

[2] `cpe:/a:openbsd:openssh:7.9p1:`
[3] `cpe:/a:python:python:3.7.3:`
[4] `cpe:/a:tornadoweb:tornado:6.0.4.`

**Table 2**
Results for the network behaviour profile tests and scenarios' mitigation outcomes.

| Tested scenarios of cyber-attack | Out of profile | Detected from profile | Δ (%) | Attack mitigated | Firewall rule type |
|---|---|---|---|---|---|
| Zero-day exploit | Yes | D | 82.37% | Yes ( 3/16) | Specific |
| DDoS attack with Mirai Botnet | Yes | H, D, W | 98.53% | Yes (12/16) | Global |
| DDoS attack with Black Energy | Yes | H, D, W | 128.42% | Yes (12/16) | Global |
| Zeus malware | Yes | W | 96.70% | Yes (12/16) | Global |
| Java-RMI backdoor | Yes | D, W | 96.88% | Yes (10/16) | Global |
| distcc_exec backdoor | Yes | D, W | 98.64% | Yes (12/16) | Global |
| UnrealIRCD backdoor | Yes | D, W | 97.69% | Yes (10/16) | Global |
| Web Tomcat exploit | Yes | W | 395.52% | Yes (10/16) | Global |
| Ruby DRb code execution | Yes | D, W | 682.16% | Yes (11/16) | Global |
| Hydra FTP bruteforce | Yes | D, W | 95.15% | Yes (12/16) | Global |
| Hydra SSH bruteforce | Yes | D, W | 99.14% | Yes (12/16) | Global |
| SMTP User Enumeration | Yes | D, W | 93.50% | Yes ( 7/16) | Global |
| NetBIOS-SSN | Yes | D, W | 307.39% | Yes (12/16) | Global |

Each VM is equipped as follows: the iRG VM uses two CPU cores, 4 GiB of RAM, and 32 GiB of storage; while the iRE VM uses four CPU cores, 16 GiB of RAM, and 32 GiB of storage. For the SOHO virtualised devices, the following OSes were used in VM or dockerised form, Ubuntu 14/16/18.04, Windows XP-SP3, Windows 7 and Android. The smart home network configuration is done via the gateway VM which is assigned two Interface Cards, from here we control the network assignments for both WAN and LAN traffic. The interface card eth0 is referenced as NIC1 (172.16.4.1/24) and has Internet connectivity (WAN). While the second interface eth1 is referenced as NIC2 (192.168.1.1/26) and acts as a gateway IP for the smart home isolated network (LAN).

### 4.2. Performance evaluation metrics

The performance evaluation parameters used to investigate the results of our system are; accuracy (AC), false alarm or False Positive Rate (FPR), Precision (P), Recall (R) and F1-score (F1). The accuracy measures the proportion of the total number of correct classifications as in the following formula:

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

where normal traffic represents positive instances while malicious represents negative instances. True Positive (TP) is the number of instances that have been correctly classified as legitimate. False Positive (FP) is the number of malicious instances that have been incorrectly classified as normal. True Negative (TN) is the number of samples of malicious traffic that have been correctly classified as anomalous. False Negative (FN) is the number of normal PCAP files that have been incorrectly classified as anomalous instances.

$$FPR = \frac{FP}{FP + TN} \tag{4}$$

Precision (P) provides the percentage of positively classified samples that are truly positive.

$$P = \frac{TP}{TP + FP} \tag{5}$$

The recall represents the number of normal samples that were correctly classified.

$$R = \frac{TP}{TP + FN} \tag{6}$$

F-score is a weighted average between precision and recall.

$$F1 = 2 \times \frac{P \times R}{P + R} \tag{7}$$

### 4.3. Results and discussion

#### 4.3.1. Network profiling results

Several tests were performed to evaluate the success of the proposed IPS framework and determine the accuracy of the ML module. During the testing, the threshold parameter of the network profiling service is set to 80% calculated rate difference in the assigned 60 s of capture time. Such a large difference from normal transmission rate, in any capture was determined to be a good baseline for our use cases. However, it is important to note that this threshold is able to be configured by the end-user to match their network use cases if their network activity throughput is markedly more volatile, or stable than the SOHO networks we tested the configuration on. Table 2 present the overall results for the network profiling component. As illustrated in Table 2, all the attack scenarios conducted using the network traffic capture replays were detected as out of profile for the devices that have been affected and were correctly identified as such, this yielded a 100% detection success rate for the attacks tested when monitoring out-of-bound network traffic behaviour based on the three network profile assignments against their average profile difference (Δ %).

Network-level attack methods and results are likely to result in false positives (or false alarms) at the expense of accuracy. To test how susceptible our solution is to false positives due to normal user interactions, we conducted testing with benign network traffic scenarios to see what operations, if any would trigger an out of profile response from the profiling system. These tests included extended SSH sessions, downloading and uploading data over FTP, downloading and uploading data over HTTP; with only one scenario: Copying an extremely large PCAP of 7 GB over the network using SCP. This was detected as abnormal by the hourly, daily and weekly profiles with a percentage difference of 281.58%. Out of the 12 scenarios conducted, only one triggered a false positive, meaning the achieved FPR was 8.3%.

#### 4.3.2. IDS system results

As mentioned before, the ML module is implemented by using MobileNetV3. The training of the neural network is done for 100 epochs with a batch size of 32 and a time of 0.16 min for each epochs. The learning rate parameter is fixed to 0.05 after running the learning rate finder function (LRFinder) [64]. The results for the ML component were achieved through the replaying of captured network attack scenarios that were performed on the Cyber-Trust platform. With the PCAP replays to the ML component, we were able to assess the metrics of malicious network traffic with quantifiable data; the results of this testing resulted in the following overall statistics. Table 3 presents the overall results of the conducted attack scenarios, which reached an overall detection accuracy of 98.35%, which is a high rate and met the required accuracy rate in practical use. The accuracy rate has been greatly improved compared to the rates achieved in the design phase (from 95% in [13] to 98%). This is mainly due to the contentious training of the ML module. When observing the results for normal traffic, the average traffic accuracy was in the ranges of 94%–98%, this results in false Positive and False Negative rates of between 1%–4% over full result testing. Comparatively, the results for malicious traffic that was tested results in average traffic accuracy of between 95%–98%, informing a false positive and a false negative rate of between 1%–4% over result testing. These averages do not reflect the best results

**Table 3**
Testing results of the ML module.

| Metrics | Results (normal and malicious traffic) | Best results |
|---|---|---|
| Accuracy (%) | ≈ 94% – 98% | 98.35% |
| FPR (%) | ≈ 1% – 4% | 0.98% |
| Precision (%) | ≈ 95% – 98% | 99.31% |
| Recall (%) | ≈ 94% – 98% | 99.01% |
| F1-score (%) | ≈ 94% – 98% | 99.16% |

that were achieved over 100 individual runs of each attack scenario processed by the ML component.

While running these proposed testing scenarios of network attack traffic over 100 runs, the best accuracy (A) result achieved was 98.35%, false Positive rate was 0.98% and false Negative rate 0.71%. The precision (P) result was also very high with a rate of 99.3%, which shows strong overall confidence in the pattern recognition process. It is worth noting that in these tests, precision is very important because getting False Negatives (FN), when malware traffic is considered as normal, cost more than False Positives (FP), when normal traffic is considered as malicious traffic. The recall percentage (R) had a result of 99.01%. The F1 value (F1) achieved was 99.16%. In conclusion, the average performance of both components was considerable. Combining the results of both the network profiling service and the machine learning component, the accuracy rate of both solutions is consistent across our range of testing. The combination of both systems results in an average accuracy rate per iteration of 99.17%. These results were acquired from testing against device exploitation from known common vulnerabilities and high impact botnets that have seen extensive infection in the real world, this speaks to the high efficacy of the solution.

To properly analyse the efficiency of our solution, we need to do a comparative analysis against more recent methods that are very close to our work [65–69]. Authors in [65] employed gray-scale images converted from malware binaries and a lightweight CNN for classifying the families of DDoS malware in IoT environments. Similarly, authors in [68] used the Binvis algorithm to generate RGB images that are used as input to the objective of malware/benign filetype classification. In [66], different ML algorithms have been used to detect PDF files and identify which ones might be malicious RGB images. In [67], authors implemented a CNN that learns visual features from executable files to classify Windows malware into families. Finally, authors in [69] used a DL image recognition method for network intrusion detection. The network characteristics are converted into four-channel images (Alpha, Red, Green, Blue). Next, the ARGB images are used to train and evaluate the pre-trained DL model ResNet50. UNSW-NB15 and BOUN DDoS datasets are used to test the proposed method. Noting that this method achieved a low accuracy rate (84.5%) with the UNSW-NB15 network intrusion dataset, which has different categories of malware traffic.

As shown in Table 4, our approach outperformed other approaches in terms of accuracy. It is worth noting that the method in [69] achieved a low accuracy rate (84.5%) with the UNSW-NB15 network intrusion dataset, which has different categories of malware traffic. The 99.7% value of accuracy has been achieved with one category of malware traffic (DDoS attacks). However, our approach, which achieved an overall accuracy of 98.35%, has been tested on a dataset that consists of different categories of cyber-attacks. In summary, the adoption of deep learning methods to recognise malware and network intrusions from features converted to images has proven to be more efficient than traditional approaches. For that a wide variety of neural network models and architectures are being explored, modified, and adopted.

#### 4.3.3. iIRS system results

The aforementioned attack scenarios were executed under sixteen different configurations of the iIRS to evaluate how four behavioural and algorithmic choices affected its attack mitigation capabilities; these were:

(1) The use of either *preset* or *CVSS-based* vulnerability metrics by the iRG to calculate the initial risk values assigned to each network host, as well as, the probabilities of exploit attempt (OR → AND nodes) and exploit success (AND → OR nodes). This choice controls the initial probabilities of the LAG and, in turn, the belief calculation of the iRE.
(2) The iRE alert management policy which controls how the POMDP model handles the set of LAG nodes identified to be enabled from the received IDS alerts. *Strict policy* forces the POMDP model to consider all IDS-matched nodes to be available to the attacker regardless of the formed belief. While, *typical policy* allows the POMDP model to only consider IDS-matched nodes whose precondition nodes are considered to be already enabled by the attacker—and are thus believed to be exploitable. In other words, the former places weight on the current state of the network (as seen through IDS alerts), whilst the latter takes advantage of the predictive abilities of the POMDP model to filter false positives.
(3) The *compromise belief threshold* ($\in [0, 1]$) at which the POMDP model considers security conditions to be enabled by the attacker; effectively controlling the sensitivity of the typical alert management policy and belief update process. During testing this threshold was set at 0.5 and 1.
(4) The choice of employing *both global and specific* firewall rules or to restrict the selection to *global rules only*. The former offers a broader choice of mitigation actions to the iRE, allowing it to take broader action to completely isolate a host from the network (global rules) or choose a more fine-grained approach by blocking a specific connection (specific rules). The latter favours the effectiveness of mitigation actions against service availability.

The results of the sixteen executions (all combinations of these choices) for each attack scenario are presented in the two rightmost columns of Table 2. The first of the two columns indicates whether the specific scenario was mitigated successfully and the number of configurations that were successful, while the second one indicates the type of the chosen firewall rule—either global or specific (as described in Section 3.5.4). From these executions, four configurations always resulted in failure to mitigate the ongoing attack across all thirteen scenarios due to the combination of a high compromise belief threshold and the choice of the typical alert matching policy. Regarding the remaining executions, the IDS correctly alerted the iIRS, but due to limitations imposed by some scenarios the current iRE implementation incorrectly matched the alerts to LAG subgraphs describing unrelated services. Problematic scenarios involved either lack of knowledge about a network host or vulnerability (as noted in the zero-day scenario) or communication over ports opened after the generation of the LAG.

## 5. Conclusions and future work

In this paper, we have introduced a novel IDS framework to identify malicious network traffic in IoT networks by using network profiling and machine learning and tested its effectiveness against network attacks by coupling it with an intrusion response system. This work is done in the context of the Cyber-Trust project [14] and tested on the Cyber-Trust testbed, which hosts a significant number of simulated IoT based home networks (SOHOs). In conclusion, the overall performance of the proposed solution was considerable, this was especially true when considering the results of the machine learning component; which recorded a peak accuracy of 98.35% over 100 tests with only a 0.98% FPR and a 99.31% precision rating. Whereas, the overall accuracy rate of the proposed solution is 99.175%. These results were acquired from testing against device exploitation from known common vulnerabilities and high impact botnets that have seen extensive infection in the real world, this speaks to the high efficacy of the

**Table 4**
Comparison of the ML-based IDS with other methods.

| Work | Colour scheme | Dataset (size) | ML/DL algorithms | Best accuracy |
|---|---|---|---|---|
| J. Su et al. [65] | Gray-scale images (64 × 64) | 365 samples | CNN | 94.00% |
| C-Y. Liu et al. [66] | RGB images (200 × 150) | 9000 samples | VGG19, CNN | 97.30% |
| Daniel Gibert et al. [67] | Gray-scale images (128 × 128) | 10 868 samples | CNN | 97.50% |
| S. O'Shaughnessy et al. [68] | RGB images (512 × 512) | 12,249 samples | kNN, RF, DT, SVM, NB | 97.70% |
| J. Toldinas et al. [69] | ARGB images (42 × 1) | 9,335,605 samples (BOUN DDoS) | ResNet50 | 99.70% |
| Our work (IDERES) | RGB images (1024 × 215) | 913 samples | MobileNetV3 | 98.35% |

solution. The accuracy rate of the machine learning component still stands to improve its accuracy rate with further training. It is worth noting that when it comes to describing future work, tests could be performed to assess whether this model can increase its accuracy with more extensive or alternative forms of binary visualisation training and techniques. The effectiveness of the IDS framework was also reflected in the increased effectiveness of the experimental IRS implementation which managed to mitigate every attack scenario that was executed.

In future work, we intend to improve the results achieved by the ML module, by using more samples for training and testing the classification module, multi-label classification for the malicious traffic behaviour (i.e keylogger, spyware, etc.) and exploring the possibility of running the ML on low power devices such as the gateway. The profiling service can also be improved in future work, allowing for adaptable variance per profile for each device by raising or lowering the out of profile threshold for a device dynamically over time. As it also has been identified in preliminary testing in [70], the graph-based network modelling capabilities of the experimental IRS can be further enhanced by implementing a dynamic graph generation process that is capable of adapting to network topology and vulnerability changes in an automatic manner, altering the graph structure without reconstructing the whole GNSM in real time. The decision-making ability of the experimental IRS also requires improvements to be made in order to better handle attacks for which limited or no information is known, while significant performance upgrades must be made before proceeding with more complex, and effective, decision-making algorithms.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 786698. The work reflects only the authors' view and the Agency is not responsible for any use that may be made of the information it contains.

## References

[1] B. Jovanović, Internet of things statistics for 2021 – taking things apart, 2021, DataProt.net. URL https://dataprot.net/statistics/iot-statistics/.
[2] P. Jayalaxmi, R. Saha, G. Kumar, N. Kumar, T.-H. Kim, A taxonomy of security issues in industrial internet-of-things: scoping review for existing solutions, future implications, and research challenges, IEEE Access 9 (2021) 25344–25359.
[3] R. Singh, H. Kumar, R. Singla, An intrusion detection system using network traffic profiling and online sequential extreme learning machine, Expert Syst. Appl. 42 (22) (2015) 8609–8624.
[4] M.B. Barcena, C. Wueest, Insecurity in the internet of things, in: Security Response, Symantec, 2015.
[5] G. Bendiab, S. Shiaeles, A. Alruban, N. Kolokotronis, IoT malware network traffic classification using visual representation and deep learning, in: 2020 6th IEEE Conference on Network Softwarization, NetSoft, IEEE, 2020, pp. 444–449.
[6] S. Hajiheidari, K. Wakil, M. Badri, N.J. Navimipour, Intrusion detection systems in the internet of things: A comprehensive investigation, Comput. Netw. 160 (2019) 165–191.
[7] A. Verma, V. Ranga, Machine learning based intrusion detection systems for IoT applications, Wirel. Pers. Commun. 111 (4) (2020) 2287–2310.
[8] A. Shameli-Sendi, N. Ezzati-Jivan, M. Jabbarifar, M. Dagenais, Intrusion response systems: Survey and taxonomy, Int. J. Comput. Sci. Netw. Secur. (IJCSNS) 12 (2012).
[9] Z. Inayat, A. Gani, N.B. Anuar, M.K. Khan, S. Anwar, Intrusion response systems: Foundations, design, and challenges, J. Netw. Comput. Appl. 62 (2016) 53–74.
[10] B. Schneier, Attack trees, 1999, https://www.schneier.com/academic/archives/1999/12/attack_trees.html. (Accessed 27 Oct. 2021).
[11] X. Ou, W.F. Boyer, M.A. McQueen, A scalable approach to attack graph generation, in: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS '06, ACM, New York, NY, USA, 2006, pp. 336–345.
[12] J. Rose, M. Swann, G. Bendiab, S. Shiaeles, N. Kolokotronis, Intrusion detection using network traffic profiling and machine learning for IoT, in: 2021 IEEE 7th International Conference on Network Softwarization, NetSoft, IEEE, 2021, pp. 409–415.
[13] R. Shire, S. Shiaeles, K. Bendiab, B. Ghita, N. Kolokotronis, Malware squid: a novel IoT malware traffic analysis framework using convolutional neural network and binary visualisation, in: Internet of Things, Smart Spaces, and Next Generation Networks and Systems, Springer, 2019, pp. 65–76.
[14] Cyber-Trust, 2021, URL https://cyber-trust.eu/. (Accessed 04 March 2021).
[15] J. Rose, M. Swann, G. Bendiab, S. Shiaeles, N. Kolokotronis, 913 Malicious network traffic PCAPs and binary visualisation images dataset, in: IEEE Dataport, 2021, http://dx.doi.org/10.21227/pda3-zy88.
[16] M.H. Ali, B.A.D. Al Mohammed, A. Ismail, M.F. Zolkipli, A new intrusion detection system based on fast learning network and particle swarm optimization, IEEE Access 6 (2018) 20255–20261.
[17] D. Preuveneers, V. Rimmer, I. Tsingenopoulos, J. Spooren, W. Joosen, E. Ilie-Zudor, Chained anomaly detection models for federated learning: An intrusion detection case study, Appl. Sci. 8 (12) (2018) 2663.
[18] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, A. Robles-Kelly, Deep learning-based intrusion detection for IoT networks, in: 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing, PRDC, IEEE, 2019, pp. 256–25609.
[19] H. Alaiz-Moreton, J. Aveleira-Mata, J. Ondicol-Garcia, A.L. Muñoz-Castañeda, I. García, C. Benavides, Multiclass classification procedure for detecting attacks on MQTT-IoT protocol, Complexity 2019 (2019).
[20] M.A. Ferrag, L. Maglaras, S. Moschoyiannis, H. Janicke, Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study, J. Inform. Secur. Appl. 50 (2020) 102419.
[21] I. Idrissi, M. Boukabous, M. Azizi, O. Moussaoui, H. El Fadili, Toward a deep learning-based intrusion detection system for IoT against botnet attacks, IAES Int. J. Artif. Intell. 10 (1) (2021) 110.
[22] M. Zhong, Y. Zhou, G. Chen, Sequential model based intrusion detection system for IoT servers using deep learning methods, Sensors 21 (4) (2021) 1113.
[23] I. Ullah, Q.H. Mahmoud, Design and development of a deep learning-based model for anomaly detection in IoT networks, IEEE Access 9 (2021) 103906–103926.
[24] Y. Otoum, D. Liu, A. Nayak, DL-IDS: a deep learning–based intrusion detection framework for securing IoT, Trans. Emerg. Telecommun. Technol. 33 (3) (2022) e3803.
[25] Y.K. Saheed, A.I. Abiodun, S. Misra, M.K. Holone, R. Colomo-Palacios, A machine learning-based intrusion detection for detecting internet of things network attacks, Alex. Eng. J. 61 (12) (2022) 9395–9409.
[26] T. Saba, A. Rehman, T. Sadad, H. Kolivand, S.A. Bahaj, Anomaly-based intrusion detection system for IoT networks through deep learning model, Comput. Electr. Eng. 99 (2022) 107810.
[27] S.K. Wagh, V.K. Pachghare, S.R. Kolhe, Survey on intrusion detection system using machine learning techniques, Int. J. Comput. Appl. 78 (16) (2013).
[28] Z. Tian, W. Jiang, Y. Li, L. Dong, A digital evidence fusion method in network forensics systems with Dempster-shafer theory, China Commun. 11 (5) (2014) 91–97.

[29] C. Luo, Z. Tan, G. Min, J. Gan, W. Shi, Z. Tian, A novel web attack detection system for internet of things via ensemble classification, IEEE Trans. Ind. Inf. 17 (8) (2020) 5810–5818.

[30] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, F. Ahmad, Network intrusion detection system: A systematic study of machine learning and deep learning approaches, Trans. Emerg. Telecommun. Technol. 32 (1) (2021) e4150.

[31] Z. Tian, W. Jiang, Y. Li, A transductive scheme based inference techniques for network forensic analysis, China Commun. 12 (2) (2015) 167–176.

[32] N. Satheesh, M. Rathnamma, G. Rajeshkumar, P.V. Sagar, P. Dadheech, S. Dogiwal, P. Velayutham, S. Sengan, Flow-based anomaly intrusion detection using machine learning model with software defined networking for OpenFlow network, Microprocess. Microsyst. 79 (2020) 103285.

[33] I.S. Al-Mandhari, L. Guan, E.A. Edirisinghe, Investigating the effective use of machine learning algorithms in network intruder detection systems, in: K. Arai, S. Kapoor, R. Bhatia (Eds.), Advances in Information and Communication Networks, Springer International Publishing, Cham, 2019, pp. 145–161.

[34] S. Jajodia, S. Noel, B. O'Berry, Topological analysis of network attack vulnerability, in: V. Kumar, J. Srivastava, A. Lazarevic (Eds.), Managing Cyber Threats: Issues, Approaches, and Challenges, Springer US, Boston, MA, 2005, pp. 247–266.

[35] O. Sheyner, J.W. Haines, S. Jha, R. Lippmann, J.M. Wing, Automated generation and analysis of attack graphs, in: Proceedings 2002 IEEE Symposium on Security and Privacy, 2002, pp. 273–284.

[36] Y. Liu, H. Man, Network vulnerability assessment using Bayesian networks, in: B.V. Dasarathy (Ed.), Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2005, Vol. 5812, SPIE, International Society for Optics and Photonics, 2005, pp. 61–71.

[37] F.-X. Aguessy, O. Bettan, G. Blanc, V. Conan, H. Debar, Bayesian attack model for dynamic risk assessment, 2016, arXiv preprint arXiv:1606.09042.

[38] N. Poolsappasit, R. Dewri, I. Ray, Dynamic security risk management using Bayesian attack graphs, IEEE Trans. Dependable Secure Comput. 9 (2012) 61–74.

[39] A. Ibrahim, S. Bozhinoski, A. Pretschner, Attack graph generation for microservice architecture, in: SAC '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1235–1242, http://dx.doi.org/10.1145/3297280.3297401.

[40] T. Musa, K.C. Yeo, S. Azam, B. Shanmugam, A. Karim, F.D. Boer, F.N. Nur, F. Faisal, Analysis of complex networks for security issues using attack graph, in: 2019 International Conference on Computer Communication and Informatics, ICCCI, 2019, pp. 1–6.

[41] G. Stergiopoulos, P. Dedousis, D. Gritzalis, Automatic analysis of attack graphs for risk mitigation and prioritization on large-scale and complex networks in industry 4.0, Int. J. Inf. Secur. 21 (2022) 37–59.

[42] C. Kiennert, Z. Ismail, H. Debar, J. Leneutre, A survey on game-theoretic approaches for intrusion detection and response optimization, ACM Comput. Surv. 51 (5) (2018) 1–31.

[43] S.A. Zonouz, H. Khurana, W.H. Sanders, T.M. Yardley, RRE: A game-theoretic intrusion response and recovery engine, IEEE Trans. Parallel Distrib. Syst. 25 (2) (2014) 395–406.

[44] A. Shameli-Sendi, M. Dagenais, ORCEF: Online response cost evaluation framework for intrusion response system, J. Netw. Comput. Appl. 55 (2015) 89–107.

[45] E. Miehling, M. Rasouli, D. Teneketzis, A POMDP approach to the dynamic defense of large-scale cyber networks, IEEE Trans. Inf. Forensics Secur. 13 (10) (2018) 2490–2505.

[46] K. Zenitani, A multi-objective cost-benefit optimization algorithm for network hardening, Int. J. Inf. Secur. (2022) 1–20.

[47] K. Pisal, S. Roychowdhury, Cyber-defense mechanism considering incomplete information using POMDP, in: D. Giri, J.K. Mandal, K. Sakurai, D. De (Eds.), Proceedings of International Conference on Network Security and Blockchain Technology, Springer Nature Singapore, Singapore, 2022, pp. 3–17.

[48] CVE, CVE website, 2021, URL https://cve.mitre.org/. (Accessed 04 March 2021).

[49] Usenix Association, Proceedings of the Second Workshop on Real Large Distributed Systems : December 13, 2005, San Francisco, CA, USA, Usenix Association, 2005, URL https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis.

[50] Nazario Jose and Net, BlackEnergy DDoS bot analysis, 2007, URL https://shorturl.at/axLV6.

[51] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, L. Wang, On the analysis of the zeus botnet crimeware toolkit, in: 2010 Eighth International Conference on Privacy, Security and Trust, 2010, URL https://ieeexplore.ieee.org/abstract/document/5593240.

[52] CVE-2018-10611 : Java remote method invocation (RMI) input port in GE MDS PulseNET and MDS PulseNET enterprise version 3.2.1 and prior ma, 2018, Cvedetails.com. URL https://www.cvedetails.com/cve/CVE-2018-10611/.

[53] NVD - CVE-2004-2687, 2021, Nist.gov. URL https://nvd.nist.gov/vuln/detail/CVE-2004-2687.

[54] CVE - CVE-2020-9484, 2020, Mitre.org. URL https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-9484.

[55] Vanhauser-thc, Vanhauser-thc/thc-hydra, 2021, GitHub, URL https://github.com/vanhauser-thc/thc-hydra.

[56] CVE - CVE-2011-2523, 2011, Mitre.org. URL https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2523.

[57] TCPDUMP/LIBPCAP public repository, 2020, Tcpdump.org. URL https://www.tcpdump.org/.

[58] F. Klassen, AppNeta, Tcpreplay - Pcap editing and replaying utilities, tcpreplay.appneta.com. URL https://tcpreplay.appneta.com/.

[59] X. Ou, S. Govindavajhala, A.W. Appel, MulVAL: A logic-based network security analyzer, in: USENIX Security Symposium. Vol. 8, Baltimore, MD, 2005, pp. 113–128.

[60] T. Swift, D.S. Warren, XSB: Extending Prolog with tabled logic programming, Theory Pract. Log. Program. 12 (1–2) (2012) 157–187.

[61] I. Koufos, N. Kolokotronis, K. Limniotis, Dynamic risk management, in: N. Kolokotronis, S. Shiaeles (Eds.), Cyber-Security Threats, Actors, and Dynamic Mitigation, CRC Press, 2021, pp. 247–280.

[62] P. Ammann, D. Wijesekera, S. Kaushik, Scalable, graph-based network vulnerability analysis, in: Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS '02, ACM, New York, NY, USA, 2002, pp. 217–224.

[63] K.-P. Grammatikakis, N. Kolokotronis, Attack graph generation, in: N. Kolokotronis, S. Shiaeles (Eds.), Cyber-Security Threats, Actors, and Dynamic Mitigation, CRC Press, 2021, pp. 281–334.

[64] L.N. Smith, Cyclical learning rates for training neural networks, in: 2017 IEEE Winter Conference on Applications of Computer Vision, WACV, IEEE, 2017, pp. 464–472.

[65] J. Su, D.V. Vasconcellos, S. Prasad, D. Sgandurra, Y. Feng, K. Sakurai, Lightweight classification of IoT malware based on image recognition, in: 2018 IEEE 42Nd Annual Computer Software and Applications Conference, Vol. 2, COMPSAC, IEEE, 2018, pp. 664–669.

[66] C.-Y. Liu, M.-Y. Chiu, Q.-X. Huang, H.-M. Sun, PDF malware detection using visualization and machine learning, in: IFIP Annual Conference on Data and Applications Security and Privacy, Springer, 2021, pp. 209–220.

[67] D. Gibert, C. Mateu, J. Planes, R. Vicens, Using convolutional neural networks for classification of malware represented as images, J. Comput. Virol. Hacking Techn. 15 (1) (2019) 15–28.

[68] S. O'Shaughnessy, S. Sheridan, Image-based malware classification hybrid framework based on space-filling curves, Comput. Secur. 116 (2022) 102660.

[69] J. Toldinas, A. Venčkauskas, R. Damaševičius, Š. Grigaliūnas, N. Morkevičius, E. Baranauskas, A novel approach for network intrusion detection using multistage deep learning image recognition, Electronics 10 (15) (2021) 1854.

[70] K.-P. Grammatikakis, I. Koufos, N. Kolokotronis, C. Vassilakis, S. Shiaeles, Understanding and mitigating banking trojans: From zeus to emotet, in: 2021 IEEE International Conference on Cyber Security and Resilience, CSR, IEEE, 2021, pp. 121–128.