Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Federated Learning and Data Mining based Botnet Attack Detection Framework for Internet of Things

Kushan Sudheera Kalupahana Liyanage

kushan@eie.ruh.ac.lk

University of Ruhuna

Madhuwantha Priyashan Lokuge Lehele Gedara

University of Ruhuna

Sanduni Pavithra Oruthota Arachchige

University of Ruhuna

Tharindu Aththanayake Malwaththe Widanalage

University of Ruhuna

Chatum Aloj Sankalpa Wijethunga Gamage

University of Ruhuna

Nadeesha Sandamali Gammana Guruge

University of Ruhuna

Research Article

Keywords: Botnet Attack, Cyber-Security, Data Mining, Federated Learning, IoT, Machine Learning

Posted Date: November 14th, 2024

DOI: https://doi.org/10.21203/rs.3.rs-5365489/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations:
No competing interests reported.

Author Contributions: Conceptualization: [Kalupahana Liyanage Kushan Sudheera], [Wijethunga Gamage Chatum Aloj Sankalpa], [Gammana Guruge Nadeesha Sandamali];

Methodology: [Kalupahana Liyanage Kushan Sudheera], [Lokuge Lehele Gedara Madhuwantha Priyashan], [Oruthota Arachchige Sanduni Pavithra], [Malwaththe Widanalage Tharindu Aththanayake];

Formal analysis and investigation: [Lokuge Lehele Gedara Madhuwantha Priyashan], [Oruthota Arachchige Sanduni Pavithra], [Malwaththe Widanalage Tharindu Aththanayake];

Writing - original draft preparation: [Kalupahana Liyanage Kushan Sudheera], [Lokuge Lehele Gedara Madhuwantha Priyashan], [Oruthota Arachchige Sanduni Pavithra];

Writing - review and editing: [Kalupahana Liyanage Kushan Sudheera], [Wijethunga Gamage Chatum Aloj Sankalpa], [Gammana Guruge Nadeesha Sandamali];

Resources: [Kalupahana Liyanage Kushan Sudheera], [Wijethunga Gamage Chatum Aloj Sankalpa];

Supervision: [Kalupahana Liyanage Kushan Sudheera], [Wijethunga Gamage Chatum Aloj Sankalpa], [Gammana Guruge Nadeesha Sandamali]

Federated Learning and Data Mining based Botnet Attack Detection Framework for Internet of Things

Kalupahana Liyanage Kushan Sudheera[1] · Lokuge Lehele Gedara Madhuwantha Priyashan[1] · Oruthota Arachchige Sanduni Pavithra[1] · Malwaththe Widanalage Tharindu Aththanayake[1] · Wijethunga Gamage Chatum Aloj Sankalpa[1] · Gammana Guruge Nadeesha Sandamali[1]

Received: date / Accepted: date

Abstract Advancements in the Internet of Things (IoT) have resulted in numerous IoT devices, both standard and non-standard, being connected to the Internet. These devices have inherent vulnerabilities and are not patched or maintained well after deployment. This has created a wider threat surface due to the sheer number of devices present. Recently, botnet attacks exploited such weaknesses in IoT and carried out massive DDoS attacks on high profile targets. In this work, we propose a data mining and federated learning based solution to detect the individual attack stages in botnet attacks. The proposed solution extracts advanced features by mining anomalous patterns in network traffic using frequent itemset mining. Then, we use these features to train a distributed federated learning solution at gateways of networks while extracting the global distribution of attack patterns at a centralized security manager without violating user privacy. We carry out a set of experiments using OpenStack and a real IoT testbed to evaluate the performance of the proposed solution. The results demonstrate that the performance of the proposed privacy-preserving system is comparable to existing solutions that compromise user privacy.

Keywords Botnet Attack · Cyber-Security · Data Mining · Federated Learning · IoT · Machine Learning

Kalupahana Liyanage Kushan Sudheera
E-mail: kushan@eie.ruh.ac.lk ·
[1] Department of Electrical and Information Engineering
Faculty of Engineering
University of Ruhuna
Sri Lanka

1 Introduction

Internet of Things (IoT) has gained immense popularity and has increasingly been deployed worldwide. As per [1], the IoT market is projected to be close to 2500 billion USD by 2029. IoT devices are becoming so popular with the benefits such as remote access to real-time efficient communication over a network of interconnecting devices. One such example is Closed-Circuit Television (CCTV) cameras, which allow the users to monitor the camera feed remotely. With the advancement of IoT technology, people tend to use more and more devices and a vast matrix of non-standard devices and endpoints are getting connected to the Internet [2]. Many of these IoT devices are open to cyber-attacks due to inherent vulnerabilities such as open ports and default credentials. Given the sheer numbers and heterogeneity, IoT devices present an attractive attack surface as evident from the recent botnet attacks such as Mirai, Hajime [3]. The recent Mirai botnet attack was able to compromise over 600K vulnerable IoT devices and take down many high profile targets including Krebs on Security (over 600Gbps DDoS attack) and Dyn, a popular DNS provider for many high-traffic sites such as Amazon, Github, Netflix, PayPal, Reddit, and Twitter [3].

A botnet attack generally consists of multiple attack stages and occur over multiple networks over time, thus spatially and temporally dispersed. A typical botnet attack starts with reconnaissance, where it scans for vulnerabilities in IoT devices. If vulnerabilities are detected, a chain of attack stages—intrusion, malware injection, Command & Control (C&C) communication, and attacks targeting external entities such as SYN flooding, DDoS, data exfiltration, etc.—follow. Often these attacks leave traces in the network traffic, which we can utilize to develop a network level solution. How-

ever, the aforementioned attack stages span over a period of time and are difficult to be spotted by manually analyzing network traffic in a limited time span.

To address these advancing threats, we should have a proper detection approach to capture these different attack stages. Such an approach can be useful in many ways, providing early warning of malware spread, extraction of signature, timely prediction of attack stages, and quick mitigation [4]. An end-point solution may not be ideal as the corresponding solution may not be able to see and learn from the different variants of the attacks. Moreover, unlike operating systems based computers and mobiles, IoT devices are not frequently updated and therefore, in the risk of being exploited [5]. Alternatively, a centralized network level solution will have the visibility to learn from different network traffic and can act as a complementary solution to a end-point solution or even work independently. However, exposing user generated data violates the privacy of the users and risks being exploited by unauthorized external parties. Recent studies such as [5] suggest that even the encrypted traffic can give out behavioral information about the users.

Therefore, in this work, we propose FDA, a Federated learning and Data mining based Attack detection framework, which learns from the global distribution of network traffic while preserving privacy of the users. FDA is a network level solution which is deployed at both gateways and a centralized network manager. At the gateway level, first the network traffic is filtered using an existing anomaly detection solution to extract anomalous network traffic. Then, anomalous traffic is mined with a data mining algorithm—Frequent Itemset Mining (FIM)—to extract patterns. The extracted patterns are further mined to obtain association rules with network features in the antecedent and attack stage in the consequent. Then, these association rules are used to train deep neural networks locally at gateways. Finally, a global model is trained at the network manager using federated learning by exchanging the weight parameters from the local models without exposing the user data. In order to evaluate the performance, we carry out a set of experiments using OpenStack. As per the results, FDA is able to compete with several existing solutions, more importantly, without violating user privacy. We further experiment with a real IoT testbed to validate the performance of FDA in practical environment.

The remainder of the paper is structured as follows. Section 2 and 3 reviews the existing literature and background, respectively. In section 4, we present the framework to detect botnet attack stages. Section 5 presents and discusses the results and finally, section 6 concludes the paper.

2 Literature Review

2.1 Detecting Attacks on IoT Networks

A three layer supervised Machine Learning (ML) based Intrusion Detection System (IDS) has been introduced in [6] to detect cyber attacks on IoT networks. In the first layer, an ML model fingerprints the devices by generating their normal profiles. Subsequently, in the second layer, it detects anomalies from the network traffic and finally, the third layer identifies the attacks from the anomalies. Different ML algorithms are tried out in the experiments, and J48 decision tree achieved the best performance across different functions. Note that all the layers are executed at the network access level and thus, they do not learn from the different characteristics across different networks. Further, the proposed system works on packet level information, and thus, both privacy and scalability are open issues.

In [7], an unsupervised anomaly detection system is proposed targeting botnet attacks in IoT networks. It further focuses on decreasing the computational requirement via reducing dimensionality of the feature vector by identifying features with higher entropy, higher variance, and Hopkins statistics. The proposed model achieves a considerable performance with a minimal feature set and is capable of detecting unknown attacks as well. However, it is possible to have high false positives as these approaches consider slight deviations from normal behaviors as suspicious activities. This restricts their applicability in real environment as users and service providers prefer not to get their normal operations disrupted.

Samy et al. [8] have proposed to bring the attack detection down to the fog layer to speed up the detection. Here, the packets are captured at the edge layer, i.e., at local gateways, and sent to a centralized cloud layer to train a Long Short-Term Memory (LSTM) based Deep Learning (DL) model. Then, the trained LSTM model is installed at fog layer to detect attacks. A few notable limitations of the framework are, i) labeling of the data collected in the edge layer, ii) computational overhead generated by training on data coming from many IoT devices, iii) violation of user privacy by sending the data collected on the edge to the cloud.

2.2 Detecting Individual Stages

There are studies that focuses on detecting specific attacks on networks. In [9], a Deep Neural Network (DNN) is proposed to detect early stages of botnet attacks. Similarly, [10] and [11] have proposed methods to detect intrusion attempts in the networks. The former

extracted login patterns to identify unauthorized intrusions, whereas the latter utilized NetFlow to detect successful SSH intrusions. In [12], the authors have proposed a method, BotDet, to detect C&C traffic, a critical stage, in botnet attacks. BotDet recognizes the use of blacklisted IP addresses, SSL certificates, domain servers, and TOR servers and generates alerts. Later, these independent alerts are correlated to minimize false positives. Among the studies that focus on the detection of DDoS attacks, [13] uses the inherent characteristics of IoT devices, such as the limited number of endpoints that an IoT device connects to and the regular time intervals between packets, in training an ML model to detect DDoS traffic. Another study in [14] combined Software Defined Networking (SDN) and ML to detect volumetric attacks in IoT networks, where normal behavior is profiled through Manufacture Usage Description (MUD). These methods solely focus on individual attacks and do not have the capability to connect dots between different stages.

2.3 Detecting Multi Stage Attacks

Different techniques are used to detect multi stage attacks on networks. Data mining is used in [15] and [16] to correlate the alerts generated by IDS in detecting multi stage attacks. The former used a probabilistic approach to estimate the similarity between alerts coming from multiple networks. The later used sequential pattern mining to correlate alerts and detect multi stage attacks. Hidden Markov Model (HMM) is used in [17] to model alerts as observable states and the corresponding actual attack stages as hidden stages. Then using the sequence and transitions of states, the likely attack type and stage is estimated. These work heavily rely on the transition of states and in modern botnet attacks, the transitions of attack stages could vary depending on many factors such as botnet variants, malware mutation, etc. In [18], the authors have used clustering to detect multi stage attacks on the basis that network flows from the same attack stage tend to fall into the same group. However, the authors have not investigated how to attribute these clusters to corresponding attack stages.

In our previous work, we proposed ADEPT [19], that addresses some of concerns highlighted above. ADEPT is a distributed framework that detects and identifies the individual attack stages in coordinated botnet attacks. It detects deviations from normal IoT profiles as anomalies at gateway level and sends alerts to a centralized security manager. At the manager, the alerts are mined to extract spatial and temporal patterns and finally, train a ML model using extracted information to detect attack stages. Even though ADEPT does not expose user data directly, with the advancement in ML, the packet meta-data contained in alerts can be used to infer behavioral information of users [20].

This work can be seen as an extension of ADEPT, where the focus is on preserving privacy of user data, while detecting stages of botnet attacks. Thus, we extract patterns at local gateways (not at the security manager, unlike ADEPT), train deep neural networks locally and use federated learning to learn from the distribution of global data.

3 Background

3.1 Threat Model

With the rapid increase in the number of IoT devices deploying globally, the number of botnets and cyber-attacks are increasing at a similar scale. A bot can be anything on the internet that compromises vulnerable devices, in our case, IoT devices. Malicious botnets engage in illicit activities like service disruption, information theft over the internet. A botnet consists of several bots, a C&C server which is the botnet's central control that sends commands to the botnet, and a botmaster who controls the entire botnet.

In the IoT domain, one of the fundamental vulnerabilities of IoT devices is that most third-party devices come with critical ports, such as SSH and Telnet, open and with default credentials, which makes them attractive targets for intruders [21]. Once IoT devices are infected with malware, these compromised devices try to spread their malware with hundreds or thousands of other vulnerable devices. The botnet controls infected systems through one or more C&C channels.

There are different variants of botnet attacks, from which Mirai and Hajime can be listed as more recent ones. The former made headlines in 2016, when it generated a more than 1 terabits per second DDoS attack taking down the Dyn DNS provider [3]. Mirai primarily targeted IP cameras, Digital Video Recorders (DVRs), and routers. Hajime botnet uses an infection method similar to the Mirai botnet. However, Hajime is based on a fully distributed communication architecture and makes use of the BitTorrent Distributed HashTag (DHT) protocol for peer discovery and the uTorrent Transport Protocol for data exchange [22].

As mentioned above, a botnet attack consists of multiple stages that may span over a period of time. While the attack stages and corresponding characteristics may slightly differ according to the botnet, we can generalize it to the following attack stages.

- Reconnaissance: Here, bots scan for vulnerable IoT devices in the Internet and explore how to exploit them. In botnet attacks, bots mainly scan for entry points, i.e., they scan for open ports where ports such as SSH and Telnet can be critical.
- Login: Bots then try to login to vulnerable IoT devices using brute force dictionary attack. Here, they use a known dictionary of usernames and passwords.
- Malware Infection: Once a bot has access to the kernel, it downloads a malware from an external server, which will be automatically executed each time the victim is rebooted.
- C&C Communication: When the malware is executed, the compromised IoT devices will establish a connection with a C&C server. Most commonly, the Internet Relay Chat (IRC) protocol implements the communication channel. The compromised IoT device joins the C&C channel specified in the malware after establishing an IRC session with the server. C&C server sends commands and receive outputs of bots and compromised devices of the botnet. Heartbeat messages are used to keep the connection between compromised devices and C&C server alive.
- Targeted Attacks: The botmaster sends commands to the C&C server to initiate an attack towards a particular target, and the C&C server spreads the command throughout the compromised devices. While there are different attacks, the most common attack used by botnets is DDoS.

  A DDoS attack attempts to make an internet service or website inaccessible by overwhelming it with large amounts of traffic from multiple sources, here from compromised IoT devices. A large-scale DDoS attack can generate traffic in the scale of tens or hundreds of gigabits per second, making it difficult for a typical network to handle. There are different versions of DDoS attack such as volumetric attacks (UDP flood), protocol based attacks (SYN flood), and application layer attacks (HTTP flood) [23].

3.2 Profiling IoT Devices

Typically, an IoT device has a specific set of protocols serving as the standard, secured data communications channels with the third parties. Although the IoT and human interaction happen via a mobile or web application, the data communications consume the underlying protocols such as Hypertext Transfer Protocol (HTTP) and Hypertext Transfer Protocol Secure (HTTPS). Thus, these set of protocols can use as a unique identifier or as normal behavior for a given IoT device.

The profile of a specific IoT device can be obtained during the first usage of the device assuming the IoT device communicates only with its defined protocols. Afterwards, these communications can contain unpermitted protocols due to cyber-attacks, communications failures, and many other reasons. The profile creation and sorting occur when an IoT device connects to the system or in the case of a software update. The generated benign (normal) profiles need to be stored in the system and during the normal operation, these profiles are read continuously to detect anomalies, i.e, any deviation from the benign profile.

In order to generate the benign profiles of IoT devices, we use Cuckoo hash table following the work in [19]. The cuckoo hashing guarantees a constant look up time for any key value. But, it goes through a few steps consuming a considerable amount of time during the insertion operation compared to other hash tables. However, in our case, data insertions happen at a low rate, while the search and read happen in real-time continuously at a high rate. So, the cuckoo hash table is ideal to store and read the IoT profile tables.

3.3 Network Anomalies

A network anomaly is a sudden deviation of the network's operations from its expected normal state [24]. Those anomalies can occur due to an intruder's actions who has illegal expectations or purely due to a failure in network functions. The anomalies that occur due to system failures can be minimized through regular maintenance. Nevertheless, anomalies generated due to intrusions should be detected and their adverse effects should be mitigated.

Two types of anomalies can be identified in an IoT network, namely, connection anomalies and behavioral anomalies.

3.3.1 Connection Anomaly

Each IoT device in a network has defined network routes and end points it connects to. If any IoT device is deviating from its defined network routes and end points, it can be considered as a connection anomaly. The C&C communications during a botnet attack is such an example. The C&C server's network path is not a defined path for any IoT device in the network.

3.3.2 Behavioral Anomaly

The behavioral anomaly occurs when the connection is acceptable, but the traffic deviates from the regular statistics. The compromised devices can communicate

with a commercial server during a DDoS attack, defined as accepted in IoT routes but with high network bandwidth. Such an event is considered as a behavioral anomaly.

3.4 Frequent Itemset Mining (FIM)

In this work, we need to extract patterns from anomalies to engineer advanced features in the classification of attack stages. FIM is a popular data mining technique, which extracts patterns from data.

An itemset (also called as a pattern) that meets the minimum support, θ, is called as a frequent itemset and the number of items in the itemset represents its size. For instance, `<<X, Y, Z>>` is an itemset with three items, and if the support (frequency) of the above itemset is greater than a specified minimum support ($0 \leq \theta \leq 1$), it will be mined as a frequent itemset.

These frequent itemsets mined from the dataset are used to generate the association rules. The association rules are if-else statements, which show the confidence of relationships between frequent itemsets. `<<X>>`$\rightarrow$`<<Y>>` represents an association rule, where `X` and `Y` are disjoint itemsets. Support, confidence and lift are the most commonly used measures in association rule mining for evaluation and setting selection thresholds [25].

FIM is associated with different algorithms that mine different frequent itemset. Apriori algorithm is used in this work as we need to measure the association between patterns [26].

Apriori Algorithm: Apriori takes an iterative approach to discover the most frequent itemsets, and the two steps of the algorithm called "join" and "prune" reduces the search space of the algorithm. "Join" generates (k+1) itemsets from k-itemsets by joining each item with itself. "Pruning" scans the count of each item and removes items that do not meet the minimum support requirement. Apriori is based on the fact that if an itemset is not frequent, all of its superset (i.e., higher order itemset containing that itemset) will not be frequent.

First, the algorithm counts the occurrences (support) of each item in the dataset to determine the candidate itemset of size one. Then, only the ones that meet the minimum support is taken as the first order frequent itemset. To find the second order itemset, the first order frequent itemset are combined and check for the minimum support requirement. This process continues until k-order frequent itemset are found (if k is specified) or until no more frequent itemset are available.

3.5 Federated Learning

The conventional neural network models require gathering of data to a central entity before training a model, which risk the data being exposed. However, governments are now coming up with transactional violation laws to protect the data owners' privacy [27]. With these latest regulations and legislation on data usage, conventional neural network based models face difficulties.

Federated learning addresses this problem by training the models at distributed locations without exposing the data to outside [28]. Once the models are trained at distributed locations, the weight parameters corresponding to the models (neural networks) are sent to a centralized entity, where they are aggregated using different algorithms. By doing that, the central model aims to learn the characteristics of the distributed models. Note that, the original data that is used to train the distributed models are not exposed, and thus, preserves privacy of the data. Yet, the model aggregation may not necessarily yield the same performance as of training a model using all the data combined, which is one of the limitations of federated learning.

4 FDA - Proposed Framework

4.1 Problem Statement

With the advancement of IoT technology, numerous types of IoT devices come to the market everyday. However, most of these devices are third-party non-standard devices, and they come with critical ports open and with default usernames and passwords. Users do not pay much attention to maintenance of these devices, especially to change default credentials. This creates an attractive attack surface for intruders and attackers. As mentioned under the threat model in section 3.1, botnets exploit these weaknesses and carry out multi stage attacks targeting IoT devices. In this work, we aim to detect different stages of botnet attacks while preserving data privacy. We limit the explanation to the stages mentioned under section 3.1. However, note that the proposed framework can be applied to detect other attacks subjected to the availability of data.

4.2 Overview of FDA

FDA is a federated learning and data mining based framework, where advanced features are engineered using data mining and a privacy preserving global de-

Table 1 An emulated profile of an IoT device

Key					Value	
int_IP	**ext_IP**	**dst_Port**	**protocol**	**dir**	**count**	**size**
172.24.4.159	10.1.0.104	8080	TCP	OUT	11	76
172.24.4.159	10.1.0.104	44634	TCP	IN	1	56
172.24.4.159	10.1.0.104	44635	TCP	IN	1	56
172.24.4.159	10.1.0.104	44636	TCP	IN	1	56

tection model is trained by aggregating the distributed models through federated learning.

Here, we consider an architecture of multiple distributed IoT networks, similar to smart homes, connected to a central security manager, which could be a service provider. Each local network has a gateway to which all the IoT devices are connected. We train the distributed deep learning models at gateways, as the gateway has access to all the network traffic flowing in the network. We see the functions of gateways to be integrated with smart home routers deployed by Internet Service Providers (ISPs), which enables them to run anomaly detection and mitigation solutions at home gateways. We assume gateways have reasonable computational and storage resources as of a standard mini-PC of the day; this is also a trend we observe in industry [29].

The security manager aggregates the distributed models and builds a global detection model. It can be deployed in the ISP datacenter or cloud, and can be integrated with Security Information and Event Management (SIEM) engine [30], or with a SDN controller. Finally, the global model is sent out to all the distributed gateways to detect botnet attack stages at the local entry points without the involvement of the centralized manager. In next subsections, we explain each step of the proposed framework in detail.

4.3 Anomaly Detection

As mentioned in section 3.3, we focus on the connection anomalies and behavioral anomalies in this work. FDA passively captures network traffic flowing through the gateways and cross check them with the benign profiles of IoT devices, which were generated in the initial stages using Cuckoo Hash tables as discussed in section 3.2. The process is indicated by step 1 in Fig. 1.

In the generation of the IoT profiles, the connections of each device is used as the 'key' of the hash table, whereas the packet statistics, i.e., mean values of packet size and count, are used as the 'values'. A connection or a key is defined with respect to 5-tuples of `<<internal IP, external IP, destination port, protocol, direction>>`. A sample profile is shown in Table 1.

In order to detect connection anomalies, we select the unique connections in each traffic capture session. If these connections do not exist in the profile hash table of that IoT device as a key, we flag such events as *connection anomalies*. If a key is found, meaning that a connection exist, then we look for the behavioral anomalies. Here, we analyze whether the statistics of the connection significantly deviates from the values in the profile table, and if so, such events are flagged as *behavioral anomalies*.

Note that a perfect anomaly detection is not required at this stage, as we will be further inspecting the anomalies to detect attack stages, and in the process, false anomalies will be further filtered out.

4.4 Pattern Extraction

The detected anomalies are then fed into the pattern extraction module as shown by step 2 in Fig. 1. A few sample anomalies are listed in Table 2. Apriori algorithm in FIM is used to extract all the frequent itemset (patterns) in the anomaly dataset. Note that FIM works with categorical features and thus, we need to transform anomalies into a categorical feature vector.

FIM Features

- IP Address: Internal IP address (i.e., IP address from the IoT network) is not included as a feature as it does not provide any useful information. The external IP address is categorized as local or public based on its range.
- Port: Only the destination port is considered here as the source port randomly changes with each new connection. There are various dynamic ports in network traffic data. Therefore, the ports other than the typically used ports of botnet attacks are categorized as 'any' (one category).
- Protocol: The network protocol used in the underlying communication. Primarily TCP and UDP, however, the exact protocols such as FTP, SSH, DNS, are also used, if information is available.
- Direction: The direction of the network traffic as in "inwards" or "outwards". The former reflects the traffic coming into the IoT device and latter represents the traffic going out from the device.
- Packet Length: Categorized into three levels. Typically, control packets such as scans are small, whereas volumetric attack packets are large. In addition, packets used in C&C communication are usually of medium size. The range of each category was selected based on a statistical analysis on the packet length.
- Attack Stage: The corresponding attack stage the network traffic belongs to. In this work, we consider

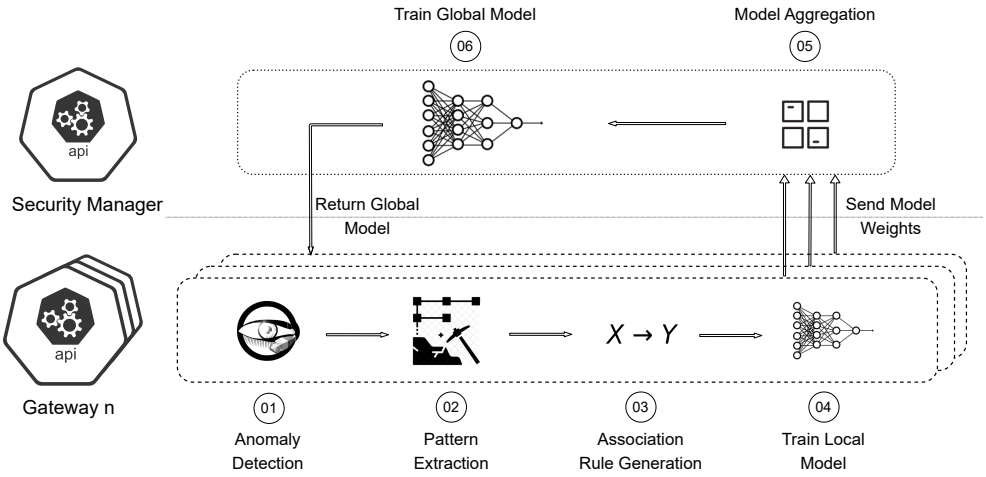


Fig. 1 Architecture and Operation of FDA.

Table 2 Sample input to FIM

#	External IP	Destination Port	Protocol	Dir	Packet Length	Attack Stage
1	Private	22	SSH	IN	Small	Scan
2	Private	22	SSH	IN	Small	Scan
3	Private	23	Telnet	IN	Small	Scan
4	Private	22	SSH	IN	Medium	Login
5	Private	22	SSH	IN	Small	Login
6	Public	21	FTP	OUT	Large	Malware
7	Public	21	FTP	OUT	Large	Malware
8	Private	21	TCP	IN	Small	Malware
9	Private	7000	TCP	OUT	Medium	C&C
10	Private	7000	TCP	OUT	Medium	C&C
11	Private	7000	TCP	IN	Small	C&C
12	Public	80	HTTP	OUT	Large	DDoS
13	Public	80	HTTP	OUT	Large	DDoS
14	Public	53	DNS	OUT	Small	DDoS
15	Public	53	DNS	OUT	Small	DDoS
16	Private	Any	TCP	OUT	Medium	Noise
17	Private	443	TCP	IN	Small	Noise
18	Private	28	UDP	IN	Large	Noise

Table 3 A few extracted patterns from data in Table 2

#	External IP	Destination Port	Protocol	Dir	Packet Length	Attack Stage	Support
1	Private	*	SSH	IN	Small	Scan	2/18
2	Private	22	SSH	IN	Small	Scan	2/18
3	Private	22	SSH	IN	*	Login	2/18
4	Public	21	FTP	OUT	Large	Malware	2/18
5	*	21	*	*	*	Malware	3/18
6	Private	7000	TCP	OUT	Medium	C&C	2/18
7	Private	7000	TCP	*	*	C&C	3/18
8	Public	80	HTTP	OUT	Large	DDoS	2/18
9	Public	53	DNS	OUT	Small	DDoS	2/18
10	Public	*	*	OUT	*	DDoS	4/18
11	Private	*	TCP	*	*	Noise	2/18
12	Private	*	*	IN	*	Noise	2/18
13	*	*	TCP	*	*	*	6/18
14	*	*	*	*	*	Scan	3/18
15	Public	*	*	OUT	Large	*	4/18

the stages mentioned under the threat model in section 3.1. Any network traffic that does not belong to any of the attack stages is labeled as "noise".

Note that FIM is an unsupervised approach and it extracts frequent patterns which has a support (i.e., occurrence) higher than a specified minimum support, $\theta(0 \leq \theta \leq 1)$. The input data to FIM is called *transactions* and Table 2 shows a few examples. Table 3 depicts a few frequent itemset extracted from Table 2 with a minimum support of 2/18[1]. We exclude patterns such as 13 and 15 as the attack stage is not included in them.

4.5 Association Rule Generation

Next, the extracted frequent itemsets are used to find the association rules. Table 4 illustrates association rules generated from the patterns in Table 3. Here, we only consider the frequent itemset with two or more items (Patterns such as 13 and 14 in Table 3 are excluded). As mentioned in section 3.4, an association rule is in the format of `<<X>>` $\rightarrow$ `<<Y>>`, where `X` is the antecedent and `Y` is the consequent. Here, we only consider the association rules which has an attack stage in the consequent and other features in the antecedent.

Simply, an association rule depicts how much `X` is associated with `Y` and in our case, it shows which feature values are associated with the corresponding attack stages. The confidence measures how frequently the features in antecedent are associated with the attack stage in consequent. Note that botnet attack stages have a higher confidence value compared to "Noise" patterns. Similar to mining of frequent itemset, we only mine association rules, if they satisfy a minimum confidence level. For instance, if a minimum confidence level of 0.5 is set, the noise rules (11 and 12 in Table 4) will be filtered out.

[1] Only a few patterns are included due to space limitation and patterns with a lower support than the minimum support are not mined.

Table 4 A few association rules generated from patterns in Table 3

#	Antecedent	Consequent	Confidence
1	<<Private, *, SSH, IN, Small>>	Scan	0.66
2	<<Private, 22, SSH, IN, Small>>	Scan	0.66
3	<<Private, 22, SSH, IN, *>>	Login	0.5
4	<<Public, 21, FTP, OUT, Large>>	Malware	1.0
5	<<*, 21, *, *, *>>	Malware	1.0
6	<<Private, 7000, TCP, OUT, Medium>>	C&C	1.0
7	<<Private, 7000, TCP, *, *>>	C&C	1.0
8	<<Public, 80, HTTP, OUT, Large>>	DDoS	1.0
9	<<Public, 53, DNS, OUT, Small>>	DDoS	1.0
10	<<Public, *, *, OUT, *>>	DDoS	0.66
11	<<Private, *, TCP, *, *>>	Noise	0.33
12	<<Private, *, *, IN, *>>	Noise	0.22

4.6 Model Training

Association rules itself can be used as an expert system to detect botnet attack stages. However, the local IoT networks do not have a wider view and do not see different network traffic pattern that is needed to build an efficient classification system. Each and every IoT network do not see network traffic related to all the attack stages. Even the same attack stage may has different characteristics when seen across different networks [31]. Therefore, it is important to train a classification model using diverse network traffic.

In order to do that, one solution is to collect network traffic from different networks to a single point, say network manager, and train a global classification model. However, as mentioned earlier, this approach is privacy intrusive as it may expose private information of users of the devices [32].

Alternatively, we use federated learning here to learn from the distributions of different local models without exposing underlying data that is used in training the models. Before developing a local neural network, we need to transform the categorical features in the association rules to numerical values and one-hot encoding is used here.

One-hot encoding yielded 28 distinctive features, and thus, the neural network input layer has 28 nodes. The output layer has 6 nodes representing the 5 attack stages and another for any rules related to noise that may exist. A few dropout layers are used to reduce overfitting. A summary of the neural network is shown in Fig. 2.

Each local network trains their own model with the same setup as shown in Fig. 2. Once the models are trained, the model weight parameters are sent to a centralized security manager via REST API.

At the security manager, the weight parameters of all the local models are averaged (step 5 in Fig. 1) to develop a new global neural network (step 6 in Fig. 1) with the same set of layers as shown in Fig. 2. This global model is expected to have learned from the distributions of different variants of botnet attack stages.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 20)                580
_________________________________________________________________
dense_1 (Dense)              (None, 200)               4200
_________________________________________________________________
leaky_re_lu (LeakyReLU)      (None, 200)               0
_________________________________________________________________
dropout (Dropout)            (None, 200)               0
_________________________________________________________________
flatten (Flatten)            (None, 200)               0
_________________________________________________________________
dense_2 (Dense)              (None, 400)               80400
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 400)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 400)               0
_________________________________________________________________
flatten_1 (Flatten)          (None, 400)               0
_________________________________________________________________
dense_3 (Dense)              (None, 200)               80200
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 200)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 200)               0
_________________________________________________________________
dense_4 (Dense)              (None, 6)                 1206
=================================================================
Total params: 166,586
Trainable params: 166,586
Non-trainable params: 0
_________________________________________________________________
```

Fig. 2 Summary of the neural network model.

However, note that underlying data, that is only used to train the local deep learning models, are never exposed to outside. Finally, the global model is sent back to the local gateways to detect attack stages locally.

In reality, gateways encounter new data from time to time. For an instance, new IoT devices can be introduced to the IoT network, which will generate new network traffic and new patterns, leading to new association rules. Therefore, once the global model is received by the gateways, they set the weight parameters of the global model as the initial weights and continue to train the neural network with newly found association rules. Then again, the local model weight parameters are shared with the security manager, which it uses to develop a further improved global model. Similarly, this process continues iteratively with the time.

Note that the attack stage detection is performed at the gateway level using the deep learning model it possesses at that moment. Once the real-time traffic comes in, the network traffic is first cross checked for anomalies similar to training. Then, we extract patterns with more than one feature in the itemset (i.e., second order and above) similar to step 2 in Fig. 1. These patterns are then fed through the neural network at gateways to detect the potential attack stage.

5 Performance Evaluation

We carried out a set of experiments to evaluate the performance of FDA. We mainly used OpenStack cloud platform to evaluate the performance comparatively, and another experiment is carried out using a practical testbed as a proof of concept.

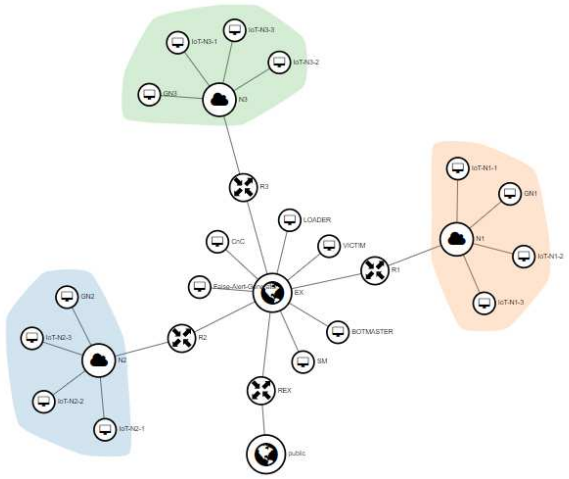


Fig. 3 Emulated IoT network in OpenStack.

5.1 Experiment Setup

We emulated an IoT network in OpenStack as shown in Fig. 3 and programmed a botnet attack resembling Mirai. The emulated network consists of 9 IoT devices, 3 gateways, a security manager, a bot master, a C&C server, a malware server (loader), a victim, a false alert generator and a security manager. All the instances are hosted as virtual machines using Ubuntu 18.04 and custom Linux images.

We implemented all the attack stages mentioned in section 3.1. Botmaster initiates the botnet attack with reconnaissance. Nmap is used for port scanning [33] as part of reconnaissance. Port scanners send a request to connect sequentially to each port and determine if the port is open, closed, or filtered, depending on the response. The devices found with opened access ports such as SSH are considered as vulnerable devices and a dictionary attack is launched by the botmaster targeting these vulnerable devices. If a login attempt is successful, the botmaster fetches the malware from the loader via FTP and executes it[2]. This establishes a C&C communication channel with the C&C server. As the last stage, the botmaster initiates a SlowLoris DDoS attack through the C&C server using the compromised IoT devices targeting the victim machine. During the span of the attack, the false alert generator generates noise traffic (normal operational traffic) resembling a realistic environment to challenge the attack stage detection algorithms.

Four experiments are performed with the above experimental setup and the results are discussed in the next sections.

[2] Note that the malware is written to init services so that each time the device restarts, the C&C channel will be established.

5.2 Experiment 01 - Evaluation of Anomaly Detection

The primary aim of this experiment is to evaluate the results of the anomaly detection module. We first generates the Cuckoo hash table based benign profiles of IoT devices at gateways by capturing the network traffic during the initial few minutes of the experiment. After that, we start both botnet attack and false alert generator. The former generates anomalous traffic, while the latter generates random non-malicious network traffic. The anomaly detection modules explained in section 4.3 continues to detect deviations from benign profiles as anomalies. The confusion matrices of obtained results at three gateways are shown in Fig. 4, Fig. 5, and Fig. 6.

According to the confusion matrices, the anomaly detection solution has not classified a single anomalous connection as a non-anomalous connection. Also, nearly 99% of total instances are correctly classified with respect to their true classes. Only a small number of non-anomalies, which are generated by the false alert generator, are classified as anomalies. However, these false positives can be further filtered during association rule generation.

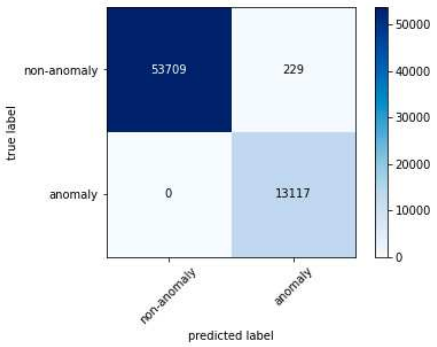


Fig. 4 Confusion matrix of anomaly detection at gateway one

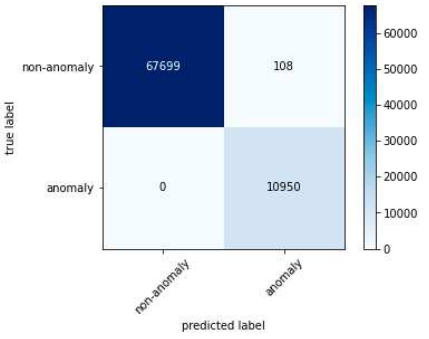


Fig. 5 Confusion matrix of anomaly detection at gateway two

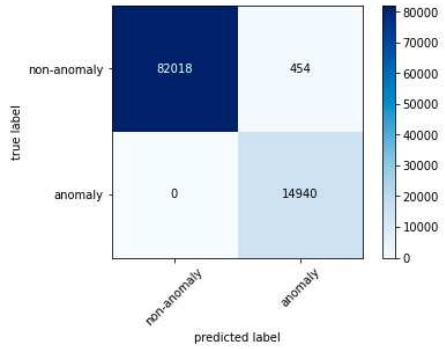


Fig. 6 Confusion matrix of anomaly detection at gateway three

5.3 Experiment 02 - Evaluation of Association Rule Generation

The objective of this experiment is to evaluate the extracted association rules and find suitable minimum confidence levels to apply in the association rule mining algorithm.

The number of association rules is usually very high. Note that we only consider the rules with attack stage in the consequent. Yet, there could be many uninteresting rules, i.e., rules that do not provide useful information. Lift is a measure that computes the interest of an association rule and the definition is in (1).

$$\text{Lift}(X \rightarrow Y) = \frac{\sup(X,Y)}{\sup(X)\sup(Y)} \tag{1}$$

where X and Y are two itemset, and "sup" indicates the support of the itemset.

An association rule can be considered as *interesting* when the lift is greater than or equal to one. A rule is a positive rule, if the rule is interesting; otherwise, the rule is known as a negative rule.

We evaluated both positive and negative rules for 200 different minimum confidence values. Then the percentages of positive rules and negative rules were computed and plotted against minimum confidence values. The results at three gateways are in Fig. 7, Fig. 8, and Fig. 9.

According to the graphs, the positive rule count increases and the negative rule count decreases with increasing minimum confidence levels. This is an expected pattern as rules with high confidence values (i.e., the features in the antecedent and attack stage in the consequent appears more frequently) tend to be interesting. Both positive and negative rule counts saturate at higher and lower ceilings, respectively, after certain minimum confidence levels. To be more specific, the rules saturate at 0.0061, 0.0091, 0.007, respectively, for three gateways. Note that when the minimum confidence level is increasing, even though the positive rule

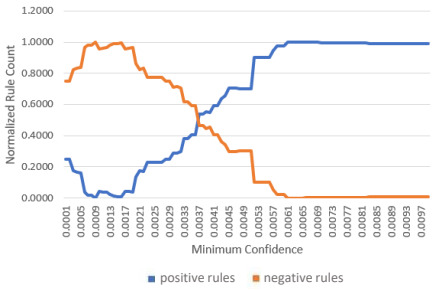


Fig. 7 Positive rule count and negative rule count vs minimum confidence - gateway one



Fig. 8 Positive rule count and negative rule count vs minimum confidence - gateway two

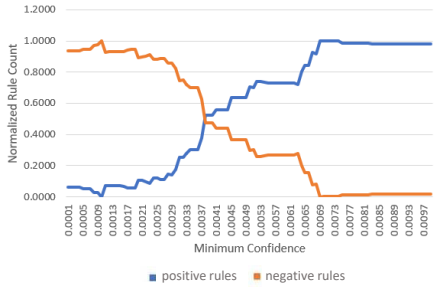


Fig. 9 Positive rule count and negative rule count vs minimum confidence - gateway three

percentage increases, the number of extracted rules inversely decreases. Therefore, we pick the above mentioned levels, where the positive and negative rules saturates, as minimum confidence levels for three gateways in next experiments.

5.4 Experiment 03 - Evaluation of Decentralized and Centralized Approaches

This experiment aims to compare the performance parameters in centralized training and decentralized training approaches. The experiment is performed for five test cases. At the beginning of the experiment, the individual dataset at gateways are randomly divided into five portions to use in five test cases. Each portion is split into training and testing with a 70:30 ratio. Then a new centralized training dataset and a testing dataset

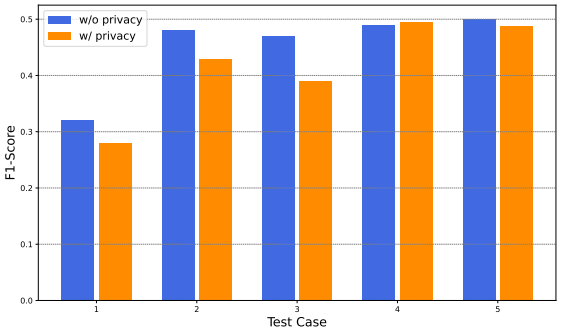


Fig. 10 Performance comparison between centralized and decentralized approaches

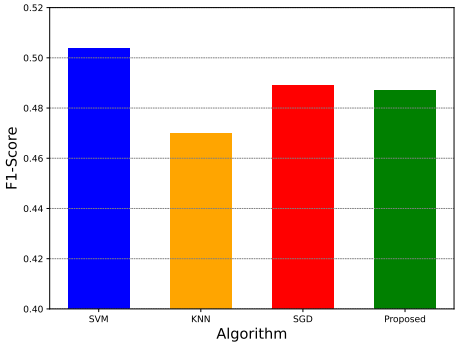


Fig. 11 F1-score with different algorithms

are created by combining each corresponding training data portion and testing data portion in each gateway. This dataset is used in training a centralized model with the same neural network layout in Fig. 2. The proposed model takes a decentralized approach as mentioned earlier, and only the local neural network models' weight parameters are shared with a centralized security manager. The F1-score[3] is used to measure the performance of each approach and the corresponding results are shown in Fig. 10.

Note that the centralized approach is privacy intrusive as the user data is sent out from the local gateways to a third party entity and thus, we label it as "w/o privacy" in Fig. 10. The decentralized approach is considered as "w/ privacy" as the user data is not exposed. We observed that the F1-score of the "w/o privacy" model is greater than the F1-score of the "w/ privacy" model in four test cases (1, 2, 3, and 5), and in the fourth test case, the F1-score of "w/o privacy" is slightly less than the "w/ privacy". Overall, the performance of the proposed approach, FDA, is competitive with its centralized counterpart (average difference is around 0.025). At the same time, FDA handles the user data at local gateways without exposing the data to a third party, unlike the centralize approach, and thus, preserves the privacy of the data.

5.5 Experiment 04 - Comparison with Existing Methods

In this experiment, we compare the performance of the proposed federated learning based method with respect to existing classification algorithms of Support Vector Machine (SVM), K-Nearest Neighbourhood (KNN), and Stochastic Gradient Descent (SGD). Note that the above comparative algorithms are applied as in a centralized approach, i.e., the complete dataset (after splitting into training and testing) is used in training and validation of the models. However, the proposed method is decentralized, i.e., the local models are generated at gateways and the global model is developed by aggregating the weight parameters of local models. Similar to the previous experiment, F1-score is used as the performance metric and the results are plotted in Fig. 11.

As per the results, SVM yields the best performance and KNN the lowest. FDA and SGD achieves a F1-score between 48% and 49%. However, the difference between the F1-score of the best performing model (i.e., SVM) and FDA is even less than 0.2%. This indicates that the performance of the proposed approach is competitive with existing supervised classification methods. On top of that, unlike the evaluated methods, FDA does not use user data in training a centralized model directly, rather use the weight parameters of the locally trained models preserving user privacy.

5.6 Proof of Concept using a Real IoT Testbed

We setup an IoT network using real IoT devices and programmable boards and carried out a Mirai-like botnet attack as a proof of concept experiment. The aim of the experiment was to demonstrate and validate the proposed botnet attack stage detection framework in a realistic environment.

The IoT network consisted of 8 IoT devices, two gateways, a botmaster, a C&C server, a loader, a victim, a router, a switch and a security manager. Fig. 12 shows the actual IoT testbed used in the experiment. The botmaster, victim and security manager are implemented using three personal computers. The two gateways, C&C server, and loader are implemented using raspberry PI boards. In addition to a few real IoT devices such as smart camera, smart bulbs, smart plugs, 5 raspberry PI boards were also emulated as IoT devices. The router works as the head node and assigns IP addresses to the devices in the network. The inter-

[3] F1-score is the harmonic mean of precision and recall.



Fig. 12 IoT Testbed.

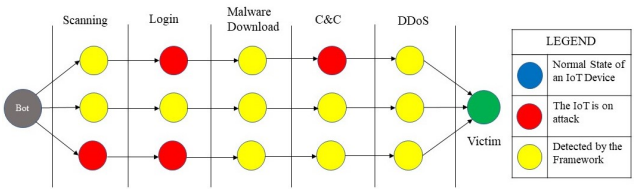


Fig. 13 GUI - Detection of attack stages.

net connection of the router is disabled to isolate the prototype network from the outside.

All the attack stages listed in section 3.1 are programmed and executed in the experiment. The same process depicted in section 4 is executed here and the detected attack stages are graphically presented in a GUI as shown in Fig. 13. As it can be seen from the results, FDA was able to detect most of the emulated attack stages during the experiment, which is a testimony for the practicality of the proposed framework.

6 Conclusion

In this work, we proposed FDA, a federated learning and data mining based framework to detect stages of botnet attacks in IoT networks. FDA is a privacy preserving, decentralized network level solution involving local gateways and a centralized security manager. At gateway level, it first filters anomalies using benign profiles of IoT devices. Then, the patterns are mined from the anomalies using FIM. These extracted patterns are used to generate association rules with attack stages in the consequent and network features in antecedent. To learn from the diverse distributions across different IoT networks, federated learning is used. Here, we trained deep neural networks using the generated association rules locally at gateways. The weight parameters of the local models are then shared with a security manager, where the weight parameters from different networks are aggregated to develop a global neural network. Finally, the weight parameters of the global model are returned to the local networks to perform attack stage detection locally. According to the experiment results, FDA displays a competitive performance with its centralized counterpart and existing supervised models, still without directly utilizing the user data.

Declarations

The authors did not receive financial support to assist with the preparation of this manuscript. In addition, the authors have no conflicts of interest to declare that are relevant to the content of this article.

Acknowledgements The authors extend their sincere gratitude to the Department of Electrical and Information Engineering, Faculty of Engineering, University of Ruhuna, Sri Lanka for their support and resources, which were essential to the successful completion of the project and the development of this paper.

References

1. Internet of Things (IOT) Market (Sep. 2021). Https://www.fortunebusinessinsights.com/industry-reports/internet-of-things-iot-market-100307; accessed: Feb. 2023
2. D. Kumar, K. Shen, B. Case, D. Garg, G. Alperovich, D. Kuznetsov, D. Kuznetsov, R. Gupta, Z. Durumeric, in *Proceedings of the 28th USENIX Conference on Security Symposium* (USENIX Association, USA, 2019), SEC'19, p. 1169–1185
3. M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J.A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, Y. Zhou, in *Proceedings of the 26th USENIX Conference on Security Symposium* (USENIX Association, USA, 2017), SEC'17, p. 1093–1110
4. D.M. Divakaran, R.P. Singh, K.L.K. Sudheera, M. Gurusamy, V. Sachidananda, in *Proc. NDSS-DISS* (2020)
5. N. Apthorpe, D. Reisman, N. Feamster. A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic (2017). DOI 10.48550/ARXIV.1705.06805. URL `https://arxiv.org/abs/1705.06805`
6. E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos, P. Burnap, IEEE Internet of Things Journal **6**(5), 9042 (2019). DOI 10.1109/JIOT.2019.2926365
7. S. Nõmm, H. Bahşi, in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)* (2018), pp. 1048–1053. DOI 10.1109/ICMLA.2018.00171
8. A. Samy, H. Yu, H. Zhang, IEEE Access **8**, 74571 (2020). DOI 10.1109/ACCESS.2020.2988854
9. M.M. Alani, in *Advances in Nature-Inspired Cyber Security and Resilience* (Springer, 2021), pp. 9–27
10. H. Siadati, N. Memon, in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Association for Computing Machinery, New York, NY, USA, 2017), CCS '17, p. 1273–1284. DOI 10.1145/3133956.3134003. URL `https://doi.org/10.1145/3133956.3134003`

11. R. Hofstede, L. Hendriks, A. Sperotto, A. Pras, SIGCOMM Comput. Commun. Rev. **44**(5), 20–26 (2014). DOI 10.1145/2677046.2677050. URL `https://doi.org/10.1145/2677046.2677050`
12. I. Ghafir, V. Prenosil, M. Hammoudeh, T. Baker, S. Jabbar, S. Khalid, S. Jaf, IEEE Access **6**, 38947 (2018). DOI 10.1109/ACCESS.2018.2846740
13. R. Doshi, N. Apthorpe, N. Feamster, in *Proc. IEEE Security and Privacy Workshops (SPW)* (2018), pp. 29–35
14. A. Hamza, H.H. Gharakheili, T.A. Benson, V. Sivaraman, in *Proc. ACM SOSR* (2019), pp. 36–48
15. A. Valdes, K. Skinner, in *Proc. 4th International Symposium on Recent Advances in Intrusion Detection* (2001), RAID '00, p. 54–68
16. H. Brahmi, S. Ben Yahia, in *Proc. Database and Expert Systems Applications* (2013), pp. 450–457
17. D. Ourston, S. Matzner, W. Stump, B. Hopkins, in *Proc. 36th Annual Hawaii International Conference on System Sciences,* (2003), pp. 10 pp.–
18. J. Shin, S.H. Choi, P. Liu, Y.H. Choi, Future Generation Computer Systems **100**, 811 (2019). DOI https://doi.org/10.1016/j.future.2019.05.032
19. K.L.K. Sudheera, D.M. Divakaran, R.P. Singh, M. Gurusamy, IEEE Internet of Things Journal **8**(8), 6591 (2021). DOI 10.1109/JIOT.2021.3055937
20. R. Trimananda, J. Varmarken, A. Markopoulou, B. Demsky, in *Proc. NDSS* (2020)
21. S. Boddy, J. Shattuck, D. Walkowski, D. Warburton. The Hunt for IoT: Multi-Purpose Attack Thingbots Threaten Internet Stability and Human Life (2018). Tech report, F5 Labs, https://www.f5.com/labs/articles/threat-intelligence/the-hunt-for-iot--multi-purpose-attack-thingbots-threaten-intern; accessed: Mar. 2023
22. C. Kolias, G. Kambourakis, A. Stavrou, J. Voas, Computer **50**(7), 80 (2017)
23. Ethical Hacking - DDOS Attacks. "https://www.tutorialspoint.com/ethical_hacking/ethical_hacking_ddos_attacks.htm.; accessed: Aug. 2020"
24. T. Ahmed, B. Oreshkin, M. Coates, in *Proceedings of the 2nd USENIX workshop on Tackling computer systems problems with machine learning techniques* (USENIX Association, 2007), pp. 1–6
25. S. Raschka, The Journal of Open Source Software **3**(24) (2018). DOI 10.21105/joss.00638. URL `http://joss.theoj.org/papers/10.21105/joss.00638`
26. R. Agrawal, R. Srikant, et al., in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215 (Santiago, Chile, 1994), vol. 1215, pp. 487–499
27. T. Klosowski. The State of Consumer Data Privacy Laws in the US (And Why It Matters) (Sep. 2021). "https://www.nytimes.com/wirecutter/blog/state-of-privacy-laws-in-us/; accessed: Mar. 2023"
28. J. Konečnỳ, H.B. McMahan, D. Ramage, P. Richtárik, arXiv preprint arXiv:1610.02527 (2016)
29. ShieldIOT: Gateway Security. Https://shieldiot.io/device-level-security; accessed: Aug. 2020
30. S. Bhatt, P.K. Manadhata, L. Zomlot, IEEE Sec. Privacy **12**(5), 35 (2014)
31. M. Bailey, E. Cooke, F. Jahanian, Y. Xu, M. Karir, in *2009 Cybersecurity Applications & Technology Conference for Homeland Security* (2009), pp. 299–304. DOI 10.1109/CATCH.2009.40
32. A. Acar, H. Fereidooni, T. Abera, A.K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.R. Sadeghi, S. Uluagac, in *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (2020), pp. 207–218
33. Port Scanners Infosec Resources. Https://resources.infosecinstitute.com/topic/port-scanners/; accessed: Nov. 2022