# EDIMA: Early Detection of IoT Malware Network Activity Using Machine Learning Techniques

Ayush Kumar and Teng Joon Lim
Department of Electrical and Computer Engineering
*National University of Singapore*
Singapore
ayush.kumar@u.nus.edu, eleltj@nus.edu.sg

*Abstract*—The widespread adoption of Internet of Things has led to many security issues. Post the Mirai-based DDoS attack in 2016 which compromised IoT devices, a host of new malware using Mirai's leaked source code and targeting IoT devices have cropped up, e.g. Satori, Reaper, Amnesia, Masuta etc. These malware exploit software vulnerabilities to infect IoT devices instead of open TELNET ports (like Mirai) making them more difficult to block using existing solutions such as firewalls. In this research, we present EDIMA, a distributed modular solution which can be used towards the detection of IoT malware network activity in large-scale networks (e.g. ISP, enterprise networks) during the scanning/infecting phase rather than during an attack. EDIMA employs machine learning algorithms for edge devices' traffic classification, a packet traffic feature vector database, a policy module and an optional packet sub-sampling module. We evaluate the classification performance of EDIMA through testbed experiments and present the results obtained.

*Index Terms*—Internet of Things, IoT, Malware, Mirai, Reaper, Satori, Botnet, Bot Detection, Machine Learning, Anomaly Detection

## I. INTRODUCTION

The Internet of Things (IoT) [1] is a network of sensing devices with limited resources and capable of wired/wireless communications with cloud services. IoT devices are being increasingly targeted by attackers using malware as they are easier to infect than conventional computers. This is due to several reasons [2] such as presence of legacy devices with no security updates, low priority given to security within the development cycle, weak login credentials, etc.

In a widely publicized attack, the IoT malware *Mirai* was used to propagate the biggest DDoS (Distributed Denial-of-Service) attack on record on October 21, 2016. The attack targeted the Dyn DNS (Domain Name Service) servers [3] and generated an attack throughput of the order of 1.2 Tbps. It disabled major internet services such as Amazon, Twitter and Netflix. The attackers had infected IoT devices such as IP cameras and DVR recorders with Mirai, thereby creating an army of bots (botnet) to take part in the DDoS attack.

The source code for Mirai was leaked in 2017 and since then there has been a proliferation of IoT malware. Script "kiddies" as well as professional blackhat/greyhat hackers have used the leaked source code to build their own IoT malware. These malware are usually variants of Mirai using a similar brute force technique of scanning random IP addresses for

open TELNET ports and attempting to login using a built-in dictionary of commonly used credentials (Remaiten, Hajime), or more sophisticated malware that exploit software vulnerabilities to execute remote command injections on vulnerable devices (Reaper, Satori, Masuta, Linux.Darlloz, Amnesia etc.). Even though TELNET port scanning can be countered by deploying firewalls (at the user access gateway) which block incoming/outgoing TELNET traffic, malware exploiting software vulnerabilities involving application protocols such as HTTP, SOAP, PHP etc. are more difficult to block using firewalls because those application protocols form a part of legitimate traffic as well.

Bots compromised by Mirai or similar IoT malware can be used for DDoS attacks, phishing and spamming [4]. These attacks can cause network downtime for long periods which may lead to financial loss to network companies, and leak users' confidential data. Bitdefender mentioned in its blog in September 2017 [5] that researchers had estimated at least 100,000 devices infected by Mirai or similar malware revealed daily through TELNET scanning telemetry data. In an October 2017 article [6], Arbor researchers estimated that the actual size of the Reaper botnet fluctuated between 10,000-20,000 bots but warned that this number could change at any time with an additional 2 million devices having been identified by botnet scanners as potential Reaper bots. A Kaspersky lab report [7] released in September 2018 says that 121,588 IoT malware samples were identified in the first half of 2018 which was three times the number of IoT malware samples in the whole of 2017.

Further, many of the infected devices are expected to remain infected for a long time. Therefore, there is a substantial motivation for detecting these IoT bots and taking appropriate action against them so that they are unable to cause any further damage. As pointed out in [8], attempting to ensure that all IoT devices are secure-by-construction is futile and it is practically unfeasible to deploy traditional host-based detection and prevention mechanisms such as antivirus, firewalls for IoT devices. Therefore, it becomes imperative that the security mechanisms for the IoT ecosystem are designed to be network-based rather than host-based.

In this research, we propose a solution towards detecting the network activity of IoT malware in large-scale networks such as enterprise and ISP (Internet Service Provider) networks.

Our proposed solution consists of machine learning (ML) algorithms running at the user access gateway which detect malware activity based on their scanning traffic patterns, a database that stores the malware scanning traffic patterns and can be used to retrieve or update those patterns, and a policy module which decides the further course of action after gateway traffic has been classified as malicious. It also includes an optional packet sub-sampling module which can be deployed for example, in case of enterprises where a number of IoT devices ($\approx$ 10-100) are connected to a single access gateway. The bot detection solution can be deployed both on physical access gateways supplied by the ISP companies or as NFV (Network Function Virtualization) functions at the customer premises/enterprise in a SDN-NFV based network architecture, where SDN stands for Software-Defined Networking.

Bots scanning for and infecting vulnerable devices are targeted in particular by our solution. This is because the scanning and propagation phase of the botnet life-cycle stretches over many months and we can detect and isolate the bots before they can participate in an actual attack such as DDoS. If the DDoS attack has already occurred (due to a botnet), detecting the attack itself is not that difficult and there are already existing methods both in literature and industry to defend against such attacks. Once the IoT bots are detected, the network operators can take suitable countermeasures such as blocking the traffic originating from IoT bots and notifying the local network administrators. The major contributions of this paper are listed below:

1) We have categorized most of the current IoT malware into a few categories to help identify similar malware and simplify the task of designing detection methods for them.
2) We have analyzed the traffic patterns for IoT malware from each category through testbed experiments and packet capture utilities.
3) We have proposed a modular solution towards detection of IoT malware activity by using ML techniques with the above traffic patterns.

## II. RELATED WORK

There are several works in the literature on detecting PC-based botnets using their CnC (Command-and-control) server communication features. Bothunter [9] builds a *bot infection dialog model* based on which three bot-specific sensors are constructed and correlation is performed between inbound intrusion/scan alarms and the infection dialog model to generate a consolidated report. Spatio-temporal similarities between bots in a botnet in terms of bot-CnC coordinated activities are captured from network traffic and leveraged towards botnet detection in a local area network in Botsniffer [10]. In BotMiner [11], the authors have proposed a botnet detection system which clusters similar CnC communication traffic and similar malicious activity traffic, and uses cross cluster correlation to detect bots in a monitored network.

There has also been some research on intrusion detection and anomaly detection systems for IoT. A whitelist-based intrusion detection system for IoT devices (Heimdall) has been presented in [12]. The authors in [13] propose an intrusion detection model for IoT backbone networks leveraging two-layer dimension reduction and two-tier classification techniques to detect U2R (User-to-Root) and R2L (Remote-to-Local) attacks.

Of late, there has been an interest in IoT botnet and attack detection in the research community resulting in a number of papers addressing these problems. In [14], deep-autoencoders based anomaly detection has been used to detect attacks launched from IoT botnets. A few works have focused on building normal communication profiles for IoT devices which are not expected to deviate much over a long period of time. DEFT [15] has used ML algorithms at SDN controllers and access gateways to build normal device traffic fingerprints while [16] proposes a tool to automatically generate MUD (Manufacturer Usage Description) profiles for a number of consumer IoT devices. In DIoT [17], the authors have proposed a method to classify typically used IoT devices into various device types and build their normal traffic profiles so that a deviation from those profiles is flagged as anomalous traffic.

Our work addresses a few important gaps in the literature when it comes to distinguishing between legitimate and botnet IoT traffic. First, the works on detecting botnets using their CnC communication features [9]–[11], [18] are designed for PC-based botnets rather than IoT botnets which are the focus of our work. Second, we do not aim to detect botnets (networks of bots) but instead, network activity generated by individual bots. IoT botnets tend to consist of hundreds of thousands to millions of devices spread over vast geographies, hence, it is impractical to detect a whole network of IoT bots. Therefore, we do not require computationally expensive clustering algorithms as used in [10], [11].

Third, unlike [14], [17], we aim to detect IoT malware activity much before the actual attack, during the scanning/infection phase. Finally, instead of fingerprinting the normal traffic of IoT devices [15], [17] and using those fingerprints towards anomaly detection, we detect the malware-induced scanning packet traffic generated by infected IoT devices. This is because the former approach suffers from limitations such as possibility of misclassification of an infected device as a legitimate device type, testing against only simple malware e.g. Mirai which may result in failure to detect other, more sophisticated malware, etc. The latter approach is not free from limitations as well, since it is not resilient against new undiscovered malware whose scanning traffic features have not been updated in the database. We advocate for a combined approach consisting of both IoT device fingerprinting/anomaly detection and IoT malware scanning traffic detection.

## III. EDIMA ARCHITECTURE

Our proposed solution towards detecting the scanning packet traffic generated by IoT malware through the use of ML

algorithms is called EDIMA (Early Detection of IoT Malware Network Activity) and is shown in Fig.1. It is designed to have a modular architecture with of five different modules:

1) **ML Classifier**: The ML classifier runs on the access gateway connected to IoT devices at customer premises or enterprise. It collects the incoming traffic samples, extracts the feature vectors for those samples and classifies them based on the ML model trained by *ML model constructor*. More details about the ML classifier are given in Section IV-B.

2) **ML Model Constructor**: The ML model for classifying access gateway traffic is trained by ML model constructor using the feature vectors and class labels retrieved from *Packet traffic feature database* as inputs to a supervised classification algorithm such as Naive Bayes (NB), Decision Trees (DT), Support Vector Machines (SVM) etc. The model is then sent to the *ML classifier*. Whenever a new malware is discovered, the ML model has to be re-trained and compared with the existing ML model for classification performance. If there is no significant improvement in performance, the existing ML model continues to be used, otherwise the re-trained ML model is updated to the *ML classifier* module.

3) **Packet Traffic Feature Database**: The database stores a list of feature vectors extracted from traffic samples collected from access gateways connected to IoT devices infected with known IoT malware as well as gateways connected to uninfected devices. The database is updated frequently for newly discovered malware. The feature vectors and corresponding class labels are retrieved by the *ML model constructor* for training ML classifier for the first time and also for re-training the classifier whenever a new malware is discovered. We envisage a community of security researchers, industry personnel and users who will collect traffic data for IoT malware through honeypots, consumer access gateways etc. The feature vectors extracted from the raw traffic data samples and the class labels assigned to those samples will be updated to the online feature database.

4) **Policy Module**: The policy module consists of a list of policies defined by network administrator which decide the course of actions to be taken once the traffic from an access gateway has been classified as malicious by the ML classifier module. For instance, the network administrator can block the entire traffic originating from bots and bring them back online only after it is confirmed that the malware has been removed from those IoT devices.

5) **Sub-sampling Module** (optional): For premises having thousands of IoT devices such as enterprises, industries etc. we also propose an optional sub-sampling module as introduced in [19]. This module samples the packet traffic from IoT devices both along time as well across the devices and presents them as input to the ML classifier module. The sub-sampling module would help reduce the computational overhead for ML classifier module by

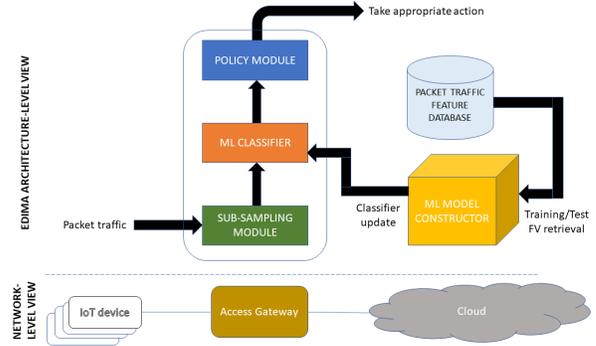forwarding only a fraction of the incoming IoT packet traffic.



**Fig. 1:** EDIMA Architecture

## IV. EXTRACTION OF IoT MALWARE TRAFFIC FEATURES

### A. Malware Categorization

We have categorized known IoT malware into three categories based on type of vulnerability that they target: TELNET, HTTP POST and HTTP GET. TELNET is an application-layer protocol used for bidirectional byte-oriented communication. Typically, a user with a terminal and running a TELNET client program, accesses a remote host running a TELNET server by requesting a connection to the remote host and logging in by providing its credentials. HTTP GET and POST are methods based on HTTP (HyperText Transfer Protocol) application-layer protocol which are used to request data from and send data to server resources respectively. For example, HTTP GET is commonly used for requesting web pages from remote web servers through a browser. We have presented the malware categories, various malware belonging to those categories and brief descriptions of their operation in Table I.

### B. ML Classification

The classification is performed on IoT access gateway-level traffic rather than device-level traffic as working on aggregate traffic is faster and reduces the memory space required. We define two classes of gateway-level traffic : *benign* and *malicious*. Benign traffic refers to the gateway traffic with no malware-induced scanning packets while malicious traffic refers to gateway traffic that includes malware-induced scanning packets from one of the three malware categories. For classification of gateway traffic, we have to first generate training data samples consisting of packet captures belonging to those classes. Benign traffic is not difficult to generate since it involves the normal operation of uninfected devices. However, malicious traffic would contain both benign traffic as well as scanning/infection packets generated by malware. To keep things simple, we chose to collect the gateway traffic statically in fixed session intervals. Further, we apply the classification algorithm on these traffic sessions rather than individual packets because per-packet classification is computationally much more costly and doesn't yield any significant benefits. The

| Category | Malware | Description |
|---|---|---|
| TELNET | Mirai | Sends SYN packets to probe open TELNET ports at random IP addresses. If successful, it tries to login using list of default credentials [20]. |
| | Hajime | Same propagation mechanism as Mirai, but no CnC server. Instead, it is built on a P2P network. Purpose seems to be to improve security of IoT devices [21]. |
| | Remaiten | Same propagation mechanism as Mirai. Downloads binary specific to targeted platform. Uses IRC protocol for CnC server communication [22]. |
| | Linux.Wifatch | Same propagation mechanism as Mirai. Apparently, it tries to secure IoT devices from other malware [23]. |
| | Brickerbot | Rewrites the device firmware, rendering the device permanently inoperable [24]. |
| HTTP POST | Satori | Sends NewInternalClient request through miniigd SOAP service (REALTEK SDK) or sends malicious packets to port 37215 (Huwaei home gateway) [25]. |
| | Masuta | Forms SOAP request which bypasses authentication and causes arbitrary code execution [26]. |
| | Linux.Darlloz | Sends HTTP POST requests by using PHP 'php-cgi' Information Disclosure Vulnerability to download the worm from a malicious server on an unpatched device [27]. |
| | Reaper | Scans first on a list of TCP Ports to fingerprint devices, then second wave of scans on TCP ports running web services such as 80, 8080..., sends HTTP POST request for command injection [28]. |
| HTTP GET | Reaper | Scanning behavior similar as above, sends HTTP request for remote command execution, usually through CGI or PHP. |
| | Amnesia | Makes simple HTTP requests, searches for a special string "Cross Web Server" in the HTTP response from target. If successful, sends four more HTTP requests which contain exploit payloads of four different shell commands [29]. |

**TABLE I:** IoT Malware Categories

The steps for gateway-level traffic classification are given below:

1) Filter each traffic session to include only TCP packets with SYN flag activated and destination port numbers belonging to a target list.
2) Extract the feature vectors for each traffic session.
3) Retrieve the trained classifier from ML model constructor and apply it on the extracted feature vectors to classify the corresponding sessions.

The target list of destination port numbers is made on the basis of information obtained from public malware exploits. For example, in 'TELNET' category, target destination port numbers are 23 and 2323. In 'HTTP POST' category, target destination port numbers are 37215, 80, 20736, 36895 etc. In 'HTTP GET' category, target destination port number is always 80.

In this work, we use a total of 4 features for ML model training and traffic classification:

1) Number of unique destination IP addresses
2) Number of packets per destination IP address (maximum, minimum, mean)

The motivation behind selecting the first feature is that the malware generate random IP addresses and send malicious requests to them. Hence, the number of unique destination IP addresses in case of malware-induced scannning traffic will be far more than benign traffic. The second feature set seeks to exploit the fact that malware typically do not send multiple malicious packets to the same IP address (only a single packet is sent in most cases), possibly to cover as many devices as possible during the scanning/propagation phase.

One may argue that the malware author/attacker can adopt a less aggressive scanning strategy to avoid detection. The attacker will incur a cost though, in terms of the malware performance, resulting in fewer infected devices in a fixed time period. We plan to investigate this malware performance-scanning behavior trade off by formulating an optimization problem in the future. For now, the duration of traffic sessions collected for training/classification can be increased to counter any decrease in scanning rates by the attacker.

## V. PERFORMANCE EVALUATION

### A. Testbed Description

We built a testbed with IoT devices, a laptop PC, Android smartphone and a wireless access gateway to collect ingress/egress traffic at the gateway which would form a part of the training data used to train the ML algorithms to be deployed in the *ML Classifier* module. The IoT devices were: Philips Hue bridge, D-Link DCS-930L Wi-Fi IP camera and TP-Link HS110 Smart Wi-Fi Plug. The laptop PC has an Intel Core i3-5020U 2.2 GHz processor with 4GB RAM and runs Windows 10 OS. Network applications such as web browser (accessing web pages, video streaming sites e.g. YouTube), email client, WiFi camera online platform etc. were run on the the laptop PC by a user. The Android smartphone has Cortex-A53 Octa-core 1.6 GHz processor with 3GB RAM and runs Android 8.0 OS. Again, the same user ran applications such as web browser, social media (Facebook/Twitter/LinkedIn), chat (WhatsApp), Wi-Fi plug app, Hue app etc. on the smartphone which also ran a few other network applications in the background. The wireless access gateway was a D-Link DIR-600 router with an Atheros AR7240 350 MHz network processor, Atheros AR9285 network adapter, 32MB RAM, 4MB flash supporting 1EEE 802.11b/g/n Wi-Fi standards. The testbed is shown in Fig. 2. We used a TP-Link TL-SG108E Gigabit Ethernet switch with port-mirroring feature to mirror the traffic from all of the above devices (IoT, laptop, smartphone) to a Raspberry Pi 3B+ Ethernet port and monitor the cumulative traffic.
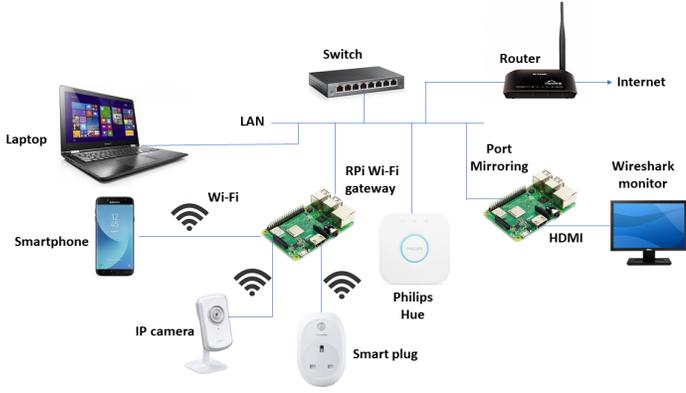
**Fig. 2:** Testbed used to collect packet traffic for ML training



**(a)** *Benign traffic* training data

**(b)** *Malicious TELNET traffic* training data

**(c)** *Benign traffic* training data

**(d)** *Malicious POST traffic* training data

**(e)** *Benign traffic* training data

**(f)** *Malicious GET traffic* training data

**Fig. 3:** Distribution of feature vector values

## B. Evaluation Methodology

As we can't use real malware due to legal and ethical considerations, we wrote scripts to simulate the generation of malicious packets based on publicly available exploits [30] for the vulnerabilities exploited by those malware. The script generates random IP addresses and sends malicious requests to them in order to execute remote command injection attacks. The injected commands were non-malicious (for ex. *ls -l*, *uname -a*), thus causing no actual harm to any device in the network even if it was vulnerable. The scanning/infection rates in our scripts were designed keeping in the mind the scanning/infection behavior reported online and the Mirai source code which is the basis for most of the current IoT malware. We selected one malware per category for our performance evaluation since the malware in each category have similar scanning/infection behavior.

A total of 60 traffic sessions of 15 minutes duration each were collected for both benign and malicious classes through our testbed. The traffic sessions collected for each case were divided into two sets: *training* and *test* data using a 70:30 split. For the training data, the class labels were assigned to each feature vector extracted from the traffic sessions included in the training data.

## C. Results

The distributions of the feature values for *benign* and *malicious* training data where the malware belongs to TELNET, HTTP POST and HTTP GET categories are shown in Fig. 3 using box plots. The distribution plots for feature *F1* under benign and malicious conditions where the malware belongs to TELNET category, are quite visibly distinct, though for the other features *Feature2, Feature3, Feature4*, the plots are not that easily distinguishable. Similarly, the distributions of *HTTP POST* and *GET* packet traffic features under benign and malicious conditions are not completely distinguishable. If there is a significant difference in the distribution of a feature under benign and malicious conditions, that difference can be leveraged by the trained ML model to distinguish between benign and malicious traffic with reasonable detection accuracy. However, if the feature distributions under the two

| Classifier | Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|
| Random Forest | 88.8% | 0.86 | 1 | 0.92 |
| k-NN | 94.44% | 0.92 | 1 | 0.96 |
| Gaussian Naive Bayes | 77.78% | 0.75 | 1 | 0.86 |

**TABLE II:** Accuracy, Precision, Recall and F1 scores for various classifiers

conditions are not easily distinguishable, it may impair the detection performance.

The *scikit-learn* ML algorithms library [31] was used for training and classification purposes. We trained Gaussian Naive Bayes, k-NN (k-Nearest Neighbor) and Random Forest algorithms with our training data and evaluated the trained ML models with test data for all three malware categories. The classification accuracy, precision, recall and F-1 scores obtained for the above three classification algorithms are shown in Table II.

The *classification accuracy* refers to the fraction of the total number of input samples whose labels are correctly predicted by a classifier. The *precision* is the ratio $\frac{TP}{TP+FP}$, where $TP$ is the number of true positives and $FP$ is the number of false positives. It represents the ability of a classifier to avoid labeling samples that are negative as positive. The *recall* is the ratio $\frac{TP}{TP+FN}$, where $TP$ is the number of true positives and $FN$ is the number of false negatives. It represents the ability of a classifier to avoid labeling samples that are positive as negative. The *F1 score* is the harmonic mean of precision and recall, expressed as $2 \times \frac{precision \times recall}{precision + recall}$. It represents balance between precision and recall offered by a classifier. The scores in Table II show that the k-NN classifier performs the best followed by Random Forest classifier and Gaussian Naive Bayes classifier.

## VI. CONCLUSION

In this paper, we proposed EDIMA, a modular solution for early detection of network activity originating from IoT malware using ML classification techniques. Existing IoT malware were distributed among multiple categories based on their targeted software vulnerabilities. Later, steps for the ML classifier operation and the features used for classification were listed. A testbed consisting of PC, smartphone and IoT devices connected to an access gateway was used to evaluate the classification performance of EDIMA. Using packet traffic captures at access gateway-level, feature vectors were extracted with class labels (*benign* or *malicious*) assigned to them. Subsequently, we depicted the distribution of benign and malicious traffic feature vectors for different malware categories. A proportion of the feature vectors extracted were used as training data to train few standard ML algorithms and the ML models thus obtained were applied to test data with their classification scores reported. As part of our future work, we are working on the software-based implementation of EDIMA and its performance evaluation. We are also planning to adapt some state-of-the-art botnet detection techniques using bot-CnC communication features and ML algorithms for malware activity detection and compare their performance with EDIMA.

## REFERENCES

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, Fourthquarter 2015.

[2] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A Survey on Security and Privacy Issues in Internet-of-Things," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1250–1258, Oct 2017.

[3] B. Krebs, "Hacked Cameras, DVRs Powered Today's Massive Internet Outage," https://krebsonsecurity.com/2016/10/hacked-cameras-dvrs-powered-todays-massive-internet-outage/, Oct 2016.

[4] I. Arghire, "IoT Botnet Used in Website Hacking Attacks," https://www.securityweek.com/iot-botnet-used-website-hacking-attacks.

[5] I. Ilascu, "Mirai Code Still Runs on Many IoT Devices," https://www.bitdefender.com/box/blog/iot-news/mirai-code-still-runs-many-iot-devices/.

[6] A. Team, "Reaper Madness," https://asert.arbornetworks.com/reaper-madness/.

[7] Y. S. M. Kuzin and V. Kuskov, "New trends in the world of IoT threats," https://securelist.com/new-trends-in-the-world-of-iot-threats/87991/.

[8] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu, "Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XIV. New York, NY, USA: ACM, 2015, pp. 5:1–5:7. [Online]. Available: http://doi.acm.org/10.1145/2834050.2834095

[9] G. Gu, P. Porras, V. Yegneswaran, and M. Fong, "Bothunter: Detecting malware infection through ids-driven dialog correlation," in *16th USENIX Security Symposium (USENIX Security 07)*. Boston, MA: USENIX Association, 2007. [Online]. Available: https://www.usenix.org/conference/16th-usenix-security-symposium/bothunter-detecting-malware-infection-through-ids-driven

[10] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic," in *Network and Distributed System Security Symposium (NDSS)*, 2008.

[11] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-independent Botnet Detection," in *Proceedings of the 17th Conference on Security Symposium*, ser. SS'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 139–154. [Online]. Available: http://dl.acm.org/citation.cfm?id=1496711.1496721

[12] J. Habibi, D. Midi, A. Mudgerikar, and E. Bertino, "Heimdall: Mitigating the Internet of Insecure Things," *IEEE Internet of Things Journal*, vol. 4, no. 4, pp. 968–978, Aug 2017.

[13] H. H. Pajouh, R. Javidan, R. Khayami, D. Ali, and K. K. R. Choo, "A Two-layer Dimension Reduction and Two-tier Classification Model for Anomaly-Based Intrusion Detection in IoT Backbone Networks," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2016.

[14] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, D. Breitenbacher, A. Shabtai, and Y. Elovici, "N-baiot: Network-based detection of iot botnet attacks using deep autoencoders," *CoRR*, vol. abs/1805.03409, 2018. [Online]. Available: http://arxiv.org/abs/1805.03409

[15] V. Thangavelu, D. M. Divakaran, R. Sairam, S. S. Bhunia, and M. Gurusamy, "Deft: A distributed iot fingerprinting technique," *IEEE Internet of Things Journal*, pp. 1–1, 2018.

[16] A. Hamza, D. Ranathunga, H. H. Gharakheili, M. Roughan, and V. Sivaraman, "Clear as mud: Generating, validating and applying iot behavioral profiles," in *Proceedings of the 2018 Workshop on IoT Security and Privacy*, ser. IoT S&#38;P '18. New York, NY, USA: ACM, 2018, pp. 8–14. [Online]. Available: http://doi.acm.org/10.1145/3229565.3229566

[17] T. D. Nguyen, S. Marchal, M. Miettinen, M. H. Dang, N. Asokan, and A. Sadeghi, "Dïot: A crowdsourced self-learning approach for detecting compromised iot devices," *CoRR*, vol. abs/1804.07474, 2018. [Online]. Available: http://arxiv.org/abs/1804.07474

[18] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Using Machine Learning Techniques to Identify Botnet Traffic," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, Nov 2006, pp. 967–974.

[19] A. Kumar and T. J. Lim, "Early Detection Of Mirai-Like IoT Bots In Large-Scale Networks Through Sub-Sampled Packet Traffic Analysis," in *Proceedings of the 2nd Future of Information and Communication Conference, Springer Lecture Notes in Networks and Systems (To be published)*, vol. 70, 2019.

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

[21] W. Grange, "Hajime worm battles Mirai for control of the Internet of Things," https://symc.ly/2q1q7Mj.

[22] L. Constantin, "Your Linux-based home router could succumb to a new Telnet worm, Remaiten," https://bit.ly/2QUIADs.

[23] M. Ballano, "Is there an Internet-of-Things vigilante out there?" https://symc.ly/2Hh9fuB.

[24] C. Cimpanu, "BrickerBot Author Retires Claiming to Have Bricked over 10 Million IoT Devices," https://bit.ly/2BkaUvd.

[25] C. X. C. Zheng and Y. Jia, "IoT Malware Evolves to Harvest Bots by Exploiting a Zero-day Home Router Vulnerability," https://bit.ly/2SVVh2x.

[26] A. Anubhav, "Masuta : Satori Creators' Second Botnet Weaponizes A New Router Exploit," https://bit.ly/2FGgav7.

[27] K. Hayashi, "Linux Worm Targeting Hidden Devices," https://symc.ly/2CnM786.

[28] Radware, "Reaper Botnet," https://bit.ly/2HeVMn5.

[29] C. Z. C. Xiao and Y. Jia, "New IoT/Linux Malware Targets DVRs, Forms Botnet," https://bit.ly/2VX7JRN.

[30] O. Security, "Exploit Database," https://www.exploit-db.com/.

[31] scikit learn, "Machine Learning in Python," https://scikit-learn.org/stable/.