THEME ARTICLE: THE SPECTER OF MALICIOUS COMPUTING

N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders

Yair Meidan
Ben-Gurion University of the Negev

Michael Bohadana
Ben-Gurion University of the Negev

Yael Mathov
Ben-Gurion University of the Negev

Yisroel Mirsky
Ben-Gurion University of the Negev

Asaf Shabtai
Ben-Gurion University of the Negev

Dominik Breitenbacher
Singapore University of Technology and Design

Yuval Elovici
Ben-Gurion University of the Negev and Singapore University of Technology and Design

The proliferation of IoT devices that can be more easily compromised than desktop computers has led to an increase in IoT-based botnet attacks. To mitigate this threat, there is a need for new methods that detect attacks launched from compromised IoT devices and that differentiate between hours- and milliseconds-long IoT-based attacks. In this article, we propose a novel network-based anomaly detection method for the IoT called N-BaIoT that extracts behavior snapshots of the network and uses deep autoencoders to detect anomalous network traffic from compromised IoT devices. To evaluate our method, we infected nine commercial IoT devices in our lab with two widely known IoT-based botnets, Mirai and BASHLITE. The evaluation results demonstrated our proposed method's ability to accurately and instantly detect the attacks as they were being launched from the compromised IoT devices that were part of a botnet.

As the number of Internet of Things (IoT) devices deployed dramatically increases worldwide[1] and the traffic volume of IoT-based DDoS attacks reaches unprecedented levels,[1–3] the need for timely detection of such attacks has become imperative for mitigating the risks associated with

1536-1268/18/$33.00 ©2018 IEEE

them. Instantaneous detection promotes network security, as it expedites the alerting and disconnection of compromised IoT devices from the network, thus stopping the botnet from propagating and preventing further outbound attack traffic.

Botnets such as Mirai typically have several distinct operational steps,[1] namely *propagation*, *infection*, *command-and-control (C&C) communication*, and *execution of attacks*. Unlike most previous studies on botnet detection, which addressed the early steps, we focus on the last step. We concentrate on large enterprises, which are expected to face an ever-growing range and quantity of IoT devices, normally connecting to their networks via Wi-Fi (short-range communications like Bluetooth and ZigBee are not in our current scope). These devices can be either self-deployed (for example, smart smoke detectors) or dynamically introduced from the outside by employees and visitors (for example, BYO wearables).

Assuming that botnet attacks are unlikely to disappear, we address the following fundamental question: Given a large number of heterogeneous IoT devices connected to an organizational network, is it possible to devise a centralized, automated method that is highly effective and accurate in detecting compromised IoT devices that have been added to a botnet and used to launch attacks?

For detecting attacks launched from IoT bots we propose N-BaIoT, a network-based approach for the IoT that uses deep learning techniques to perform anomaly detection. Specifically, we extract statistical features that capture behavioral snapshots of benign IoT traffic, and train a deep autoencoder (one for each device) to learn the IoT device's normal behaviors. The autoencoders attempt to compress snapshots. When an autoencoder fails to reconstruct a snapshot, it is a strong indication that the observed behavior is anomalous (the IoT device has been compromised and is exhibiting an unknown behavior). An advantage of using deep autoencoders is their ability to learn complex patterns—for example, of various device functionalities. This results in an anomaly detector with hardly any false alarms. We empirically show that our autoencoders' false-alarm rate is considerably lower than three other algorithms commonly used for anomaly detection.[4]

This approach to detecting infected IoT devices has three main benefits:

- *Heterogeneity tolerance*. Compared to classical computing environments, the IoT domain is highly diverse.[2,3] However, by profiling each device with a separate autoencoder, our method addresses the growing heterogeneity of IoT devices.
- *Open world*. Typically in deep learning applications, models are trained to classify based on labels provided by experts (for example, malicious or benign). However, our autoencoders are trained to detect when a behavior is abnormal. Thus, our method can detect previously "unseen" botnet behaviors, which is important given the continuously evolving variants[2] of existing botnets or emergence of new botnets, which already make most detection methods obsolete.[5]
- *Efficiency*. In the enterprise scenario, it is common to monitor the traffic data of all connected hosts, but the amount of monitored traffic is prohibitively large to store and use for training deep neural networks (DNNs). Our method uses incremental statistics to perform the feature extraction, and the training of the autoencoders can be performed in a semi-online manner (train on a batch of observations and then discard). The training is therefore practical, and there is no storage concern. Additionally, our method is network-based so it does not consume any computation, memory, or energy resources from the (typically constrained) IoT devices. Thus, our method does not jeopardize their functionality or impair their lifespan. Our focus on the attack operational step (as opposed to the early steps) also makes our method indifferent to the botnet propagation protocols and the possibly encrypted[5] C&C channels.

The contributions of this article can be summarized as follows:

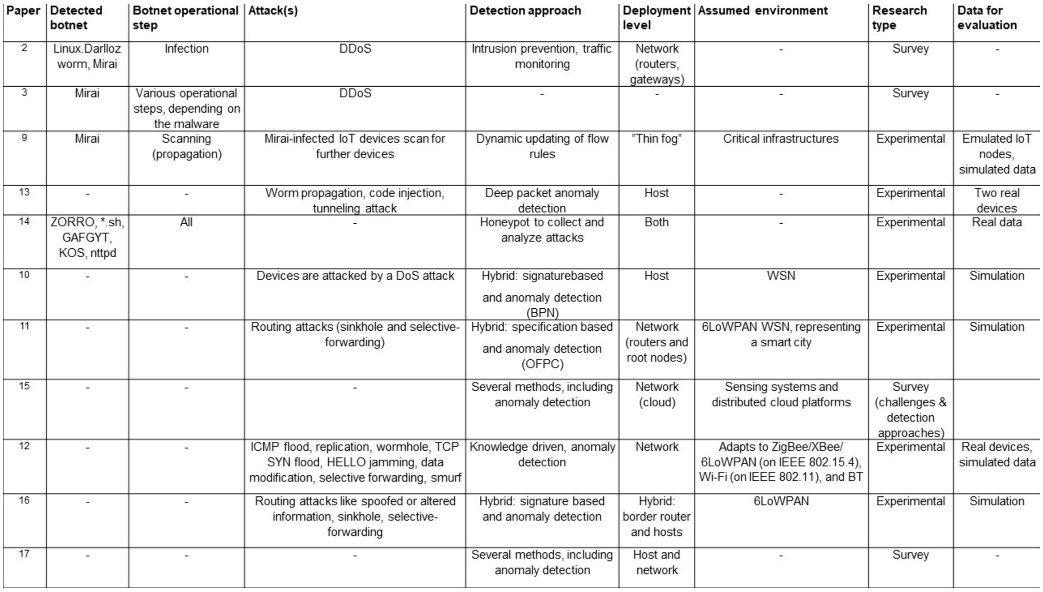
1. To the best of our knowledge, we are the first to apply autoencoders to IoT network traffic for anomaly detection as a complete means of detecting botnet attacks. Even in the larger domain of network traffic analysis, autoencoders have not been used as fully automated standalone malware detectors but rather as preliminary tools for either feature learning[6] or dimensionality reduction,[7] or at most as semi-manual outlier detectors that substantially depend on human labeling for subsequent classification[8] or further inspection by security analysts.[4]

2. Unlike previous experimental studies on the detection of IoT botnets or IoT traffic anomalies that relied on emulated or simulated data,[9–12] we perform empirical evaluation with real traffic data, gathered from nine commercial IoT devices infected by authentic botnets from two families. We examine Mirai and BASHLITE, two of the most common IoT-based botnets, which have already demonstrated[1] their harmful capabilities. To enable reproducibility and address the lack of public botnet datasets,[5] particularly for the IoT, we share our network traces at http://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT.

RELATED WORK

The botnet detection methods suggested thus far can be categorized based on the specific operational step to be detected and the detection approach. Table 1 is based on this categorization and further summarizes previous studies on the detection of IoT-related anomalies, botnets, and malware attacks.

Table 1. Prior studies on the detection of IoT-related anomalies, botnets, and malware attacks.



Paper	Detected botnet	Botnet operational step	Attack(s)	Detection approach	Deployment level	Assumed environment	Research type	Data for evaluation
2	Linux.Darlloz worm, Mirai	Infection	DDoS	Intrusion prevention, traffic monitoring	Network (routers, gateways)	-	Survey	-
3	Mirai	Various operational steps, depending on the malware	DDoS	-	-	-	Survey	-
9	Mirai	Scanning (propagation)	Mirai-infected IoT devices scan for further devices	Dynamic updating of flow rules	"Thin fog"	Critical infrastructures	Experimental	Emulated IoT nodes, simulated data
13	-	-	Worm propagation, code injection, tunneling attack	Deep packet anomaly detection	Host	-	Experimental	Two real devices
14	ZORRO, *.sh, GAFGYT, KOS, nttpd	All	-	Honeypot to collect and analyze attacks	Both	-	Experimental	Real data
10	-	-	Devices are attacked by a DoS attack	Hybrid: signaturebased and anomaly detection (BPN)	Host	WSN	Experimental	Simulation
11	-	-	Routing attacks (sinkhole and selective-forwarding)	Hybrid: specification based and anomaly detection (OFPC)	Network (routers and root nodes)	6LoWPAN WSN, representing a smart city	Experimental	Simulation
15	-	-	-	Several methods, including anomaly detection	Network (cloud)	Sensing systems and distributed cloud platforms	Survey (challenges & detection approaches)	
12	-	-	ICMP flood, replication, wormhole, TCP SYN flood, HELLO jamming, data modification, selective forwarding, smurf	Knowledge driven, anomaly detection	Network	Adapts to ZigBee/XBee/ 6LoWPAN (on IEEE 802.15.4), Wi-Fi (on IEEE 802.11), and BT	Experimental	Real devices, simulated data
16	-	-	Routing attacks like spoofed or altered information, sinkhole, selective-forwarding	Hybrid: signature based and anomaly detection	Hybrid: border router and hosts	6LoWPAN	Experimental	Simulation
17	-	-	-	Several methods, including anomaly detection	Host and network	-	Survey	-

Previous IoT-related botnet detection studies[9,13] focused mainly on the early steps of propagation and communication with the C&C server. However, given that botnet attacks continue to mutate on a daily basis[1] and become increasingly sophisticated,[2] we anticipate that some of these mutations will eventually bypass existing methods of early detection. Moreover, mobile IoT devices might get contaminated when connected to external networks. For instance, smartwatches might connect to dubious free Wi-Fi networks when their owners arrive at airports. Hence, monitoring organizational networks for identifying the early steps of infection alone is insufficient. Accordingly, we focus on a later step of botnet operation, when IoT bots begin launching cyberattacks. In that sense, N-BaIoT adds a last-line-of-defense security layer. It instantly detects the IoT-based attacks and minimizes their impact by issuing an immediate alert that recommends the isolation of any compromised device from the network until it is sanitized.

Botnet detection approaches are either host-based[10,13] or network-based.[9,11,12,15] We consider host-based techniques less realistic because not all IoT manufacturers can be relied on to install designated host-based anomaly detectors on their products; there is limited access to some IoT devices (for example, wearables), so the installation of software on end devices cannot be enforced; the constrained computation and power of most IoT devices impose constraints on the complexity and efficiency of host-based anomaly detection algorithms, which also might consume energy and computation from the devices and thus harm their functionality; and in the enterprise scenario we assume, where various and numerous IoT devices connect to the organizational network, a single nondistributed solution is preferred.

A hierarchical taxonomy of network-based botnet detection approaches, not limited to the IoT domain, was proposed by Sebastián García, Alejandro Zunino, and Marcelo Campo.[5] One of the detection sources they surveyed was honeypots, which have commonly been used for collecting, understanding, characterizing, and tracking botnets[14] but are not necessarily useful for detecting compromised endpoints or the attacks emanating from them. Moreover, honeypots normally require a substantial investment in procurement or emulation of real devices, data inspection, signature extraction, and keeping up with mutations. According to Garcia and his colleagues,[5] normal networks constitute an alternative detection source, where network intrusion detection systems monitor traffic data continuously and automatically while using pattern matching to detect signs of undesirable activities. Such patterns may rely on signatures identified by honeypots, DNS traffic with a potential C&C server, traffic anomalies,[13] data mining, or hybrid approaches.[10,11] Similar to Douglas H. Summerville, Kenneth M. Zach, and Yu Chen,[13] we find that the anomaly-based approach is best suited for detecting compromised IoT devices because these connected appliances are typically task-oriented (for example, specifically designed to detect motion or measure humidity). Accordingly, they execute fewer and potentially less complex network protocols, and exhibit traffic with less variance than PCs. As such, detecting deviations from their normal patterns should be more accurate and robust.

García and his coauthors surveyed many detection algorithms[5] but did not cite artificial neural networks or even mention autoencoders. Studies on these subjects within the greater domain of cybersecurity have been published more recently, yet they are dissimilar to our approach, unrelated to the IoT, and often not directly connected to botnets. For instance, Ignacio Arnaldo and his colleagues;[6] Yuancheng Li, Rong Ma, and Runhai Jiao;[7] and Yang Yu, Jun Long, and Zhiping Cai[18] applied shallow autoencoders for preliminary feature learning and dimensionality reduction, followed by random forests, deep belief networks, and softmax regression, respectively, for classification and fine-tuning. Although Kalyan Veeramachaneni and his colleagues[8] extended autoencoders for outlier detection, they still required security analysts to actively label data for subsequent supervised learning. Closer to our approach, Aaron Tuor and his coauthors[4] apply deep learning to system logs for detecting insider threats. Differently from us, they use DNNs and recurrent neural networks (RNNs), and depend on further manual inspection.

In conclusion, unlike previous approaches we learn from benign data by training deep autoencoders for each device, and use them as standalone automatic tools for instantaneous detection of existing and unseen IoT botnet attacks.

PROPOSED DETECTION METHOD

Our proposed method for detecting IoT botnet attacks relies on deep autoencoders for each device, trained on statistical features extracted from benign traffic data. When applied to new (possibly infected) data of an IoT device, detected anomalies may indicate that the device is compromised. This method consists of four main stages: data collection, feature extraction, training an anomaly detector, and continuous monitoring.

Data Collection

We capture the raw network traffic data (in pcap format) using port mirroring on the switch through which the organizational traffic typically flows. To ensure that the training data is clean of malicious behaviors, an IoT network's normal traffic is collected immediately following the device's installation in the network.

Feature Extraction

Whenever a packet arrives, we take a behavioral snapshot of the hosts and protocols that communicated this packet. The snapshot obtains the packet's context by extracting 115 traffic statistics over several temporal windows to summarize all of the traffic that has originated from the same IP in general (source IP), originated from both the same source MAC and the same IP address (source MAC-IP), been sent between the source and destination IPs (channel), and been sent between the source to destination TCP/UDP sockets (socket).

We extract the same set of 23 features capturing the above (see Table 2) from 5 time windows: the most recent 100 ms, 500 ms, 1.5 sec, 10 sec, and 1min. These features can be computed quickly and incrementally and thus facilitate real-time detection of malicious packets. Additionally, although generic these features can capture specific behaviors like source IP spoofing,[2] characteristic of Mirai's attacks. For instance, when a compromised IoT device spoofs an IP address, the features aggregated by the source MAC-IP, source IP, and channel will immediately indicate a large anomaly due to the unseen behavior originating from the spoofed IP address.

Table 2. Extracted features.

Value	Statistic	Aggregated by	Total Num. of Features
Packet size (of outbound packets only)	Mean, Variance	Source IP,* Source MAC-IP,** Channel, Socket***	8
Packet count	Number	Source IP, Source MAC-IP, Channel, Socket	4
Packet jitter (the amount of time between packet arrivals)	Mean, Variance, Number	Channel	3
Packet size (of both inbound and outbound together)	Magnitude, Radius, Covariance, Correlation coefficient	Channel, Socket	8

* The source IP is used to track the host as a whole.

** The source MAC-IP adds the capability to distinguish between traffic originating from different gateways and spoofed IP addresses.

*** The sockets are determined by the source and destination TCP or UDP port numbers. For example, all of the traffic sent from 192.168.1.12:1234 to 192.168.1.50:80 (traffic flowing from one socket to another).

Further details and the datasets themselves are publicly available at http://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT

Training an Anomaly Detector

As our base anomaly detector, we use deep autoencoders and maintain a model for each IoT device separately. An autoencoder is a neural network trained to reconstruct its inputs after some compression. The compression ensures that the network learns the meaningful concepts and the relation among its input features. If an autoencoder is trained on benign instances only, it will succeed at reconstructing normal observations but fail at reconstructing abnormal observations (unknown concepts). When a significant reconstruction error is detected, we classify the given observations as anomalous.

We optimize each trained model's parameters and hyperparameters such that when applied to unseen traffic the model maximizes the *true positive rate* (TPR, detecting attacks once they occur) and minimizes the *false positive rat*e (FPR, wrongly marking benign data as malicious). For training and optimization, we use two separate datasets that only contain benign data, from which the model *learns* patterns of normal activity. The first dataset is the *training set* (DS_{trn}) and is used for training the autoencoder, given input parameters such as the *learning rate* (η, the size of the gradient descent step) and the number of *epochs* (complete passes through the entire DS_{trn}). The second dataset is the *optimization set* (DS_{opt}) and is used to optimize these two hyperparameters (η and epochs) iteratively until the *mean square error* (MSE) between a model's input (original feature vector) and output (reconstructed feature vector) stops decreasing. Stopping at this point prevents overfitting DS_{trn}, thus promoting better detection results with future data. DS_{opt} is later used to optimize a *threshold* (tr) that discriminates between benign and malicious observations and, finally, the *window size* (ws), by which the FPR is minimized.

Once the model training and optimization is complete, the tr* is set. This anomaly threshold, above which an instance is considered anomalous, is calculated as the sum of the sample mean and standard deviation of MSE over DS_{opt}:

$$tr^* = \overline{MSE}_{DS_{opt}} + s(MSE_{DS_{opt}})$$

Preliminary experiments revealed that deciding whether a device's packet stream is anomalous based on a single instance enables very accurate detection of IoT-based botnet attacks (high TPR). However, benign instances were too often (5–7 percent of cases) falsely marked as anomalous. Thus, we base the abnormality decision on a *sequence* of instances by implementing a majority vote on a moving window. We determine the minimal window size ws* as the shortest sequence of instances, a majority vote that produces 0 percent FPR on DS_{opt}:

$$ws^* = \arg\min_{|ws|}(|\{packet \in ws \mid MSE(packet) > tr^*\}| > \frac{ws}{2})$$

Continuous Monitoring for Anomaly Detection

Eventually, we apply the optimized model to feature vectors extracted from continuously observed packets to mark each instance as benign or anomalous. Then, we use a majority vote on a sequence (the length of ws*) of marked instances to decide whether the entire respective stream is benign or anomalous. Consequently, an alert can be issued upon the detection of an anomalous stream, as it might indicate malicious activity on the IoT device.

EMPIRICAL EVALUATION

Our experiments strived to authentically represent IoT devices deployed in an enterprise setting infected by real-world botnets and executing genuine attacks.

Lab Setup

To replicate a typical organizational data flow, we collected the traffic data from IoT devices connected via Wi-Fi to several access points, wire connected to a central switch that also connects to a router. To sniff the network traffic, we performed port mirroring on the switch and recorded the data using Wireshark. To evaluate our detection method as realistically as possible, we also deployed all of the components of two botnets (see Figure 1) in our isolated lab and used them to infect nine commercial IoT devices (see Table 3).

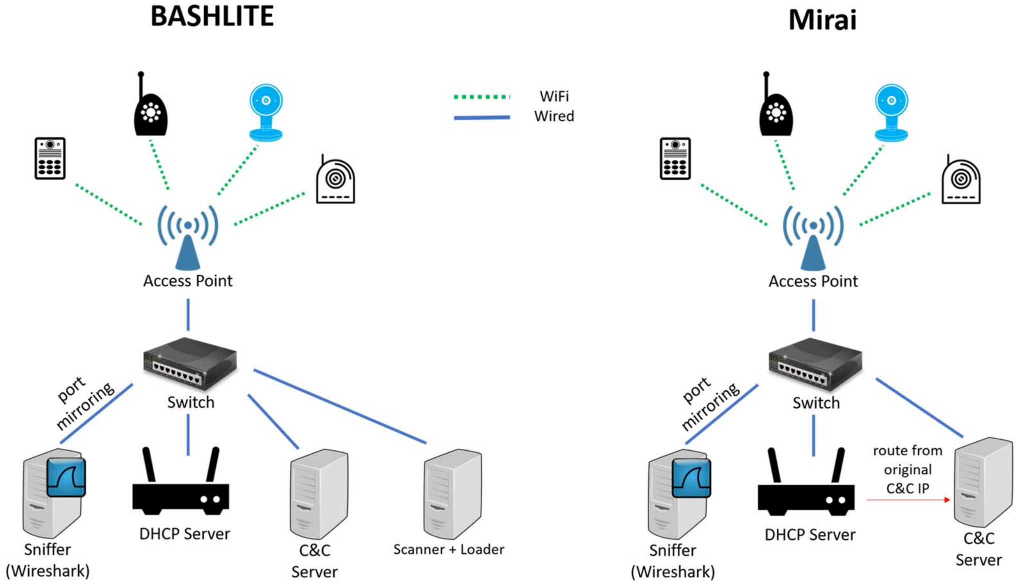
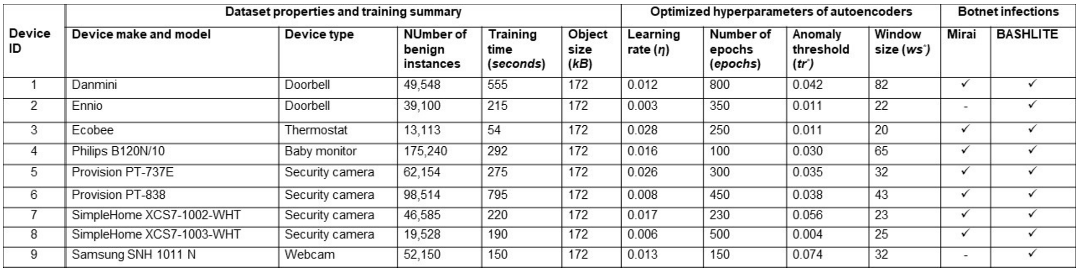


Figure 1. Lab setup for detecting IoT botnet attacks.

Table 3. Overview of the training stage.



Device ID	Dataset properties and training summary					Optimized hyperparameters of autoencoders				Botnet infections	
	Device make and model	Device type	NUmber of benign instances	Training time (*seconds*)	Object size (*kB*)	Learning rate (η)	Number of epochs (*epochs*)	Anomaly threshold (tr^*)	Window size (ws^*)	Mirai	BASHLITE
1	Danmini	Doorbell	49,548	555	172	0.012	800	0.042	82	✓	✓
2	Ennio	Doorbell	39,100	215	172	0.003	350	0.011	22	-	✓
3	Ecobee	Thermostat	13,113	54	172	0.028	250	0.011	20	✓	✓
4	Philips B120N/10	Baby monitor	175,240	292	172	0.016	100	0.030	65	✓	✓
5	Provision PT-737E	Security camera	62,154	275	172	0.026	300	0.035	32	✓	✓
6	Provision PT-838	Security camera	98,514	795	172	0.008	450	0.038	43	✓	✓
7	SimpleHome XCS7-1002-WHT	Security camera	46,585	220	172	0.017	230	0.056	23	✓	✓
8	SimpleHome XCS7-1003-WHT	Security camera	19,528	190	172	0.006	500	0.004	25	✓	✓
9	Samsung SNH 1011 N	Webcam	52,150	150	172	0.013	150	0.074	32	-	✓

Botnets Deployed

We deployed two of the most common IoT botnets, BASHLITE and Mirai, in our lab and collected traffic data before and after the infection.

BASHLITE (also known as Gafgyt, Q-Bot, Torlus, Lizard-Stresser, and Lizkebab) is one of the most infamous types of IoT botnets, and its code and behavior can be found in other IoT malware as well. To launch an attack, the botnet infects Linux-based IoT devices by brute forcing default credentials of devices with open Telnet ports. In our research, the IoT devices were infected using the binaries from the IoTPOT dataset[14] (namely Gafgyt). To adjust the attacks to our lab, the IP address of the C&C server was extracted from the malware's binary, and all of the network traffic to this IP was routed to a server in our lab that functions as a C&C server. Once a new bot connected to this server and was under its control, this server was able to command the infected device to launch attacks.

We deployed Mirai using its published source code (https://github.com/jgamblin/Mirai-Source-Code). The experimental setup included a C&C server and a server with a scanner and loader. The scanner and loader components are responsible for scanning and identifying vulnerable IoT devices, and loading the malware to the vulnerable IoT devices detected. Once a device was infected, it automatically started scanning the network for new victims while waiting for instructions from the C&C server.

Attacks Executed

We executed and tested the following attacks in our lab.

BASHLITE Attacks

1. Scan: Scanning the network for vulnerable devices
2. Junk: Sending spam data
3. UDP: UDP flooding
4. TCP: TCP flooding
5. COMBO: Sending spam data and opening a connection to a specified IP address and port

Mirai Attacks

1. Scan: Automatic scanning for vulnerable devices
2. Ack: Ack flooding
3. Syn: Syn flooding
4. UDP: UDP flooding
5. UDPplain: UDP flooding with fewer options, optimized for higher packets per second

Experimental Results and Discussion

Each of the nine sets of benign data we collected in our lab, corresponding to the nine IoT devices, was divided chronologically into three equidimensional sets: DS_{trn} for training the autoencoder, DS_{opt} for parameter optimization, and the benign part of DS_{tst} for estimating the FPR. To imitate real-world settings and thus assess N-BaIoT more realistically, we made sure to incorporate traffic from the entire (normal) lifecycle of the devices. Particularly, in each of the three sets of each IoT device we included not only traffic data of frequent actions (for example, a webcam transmitting video) but also infrequent actions (for example, accessing a webcam via the mobile app, moving in front of it, or booting it).

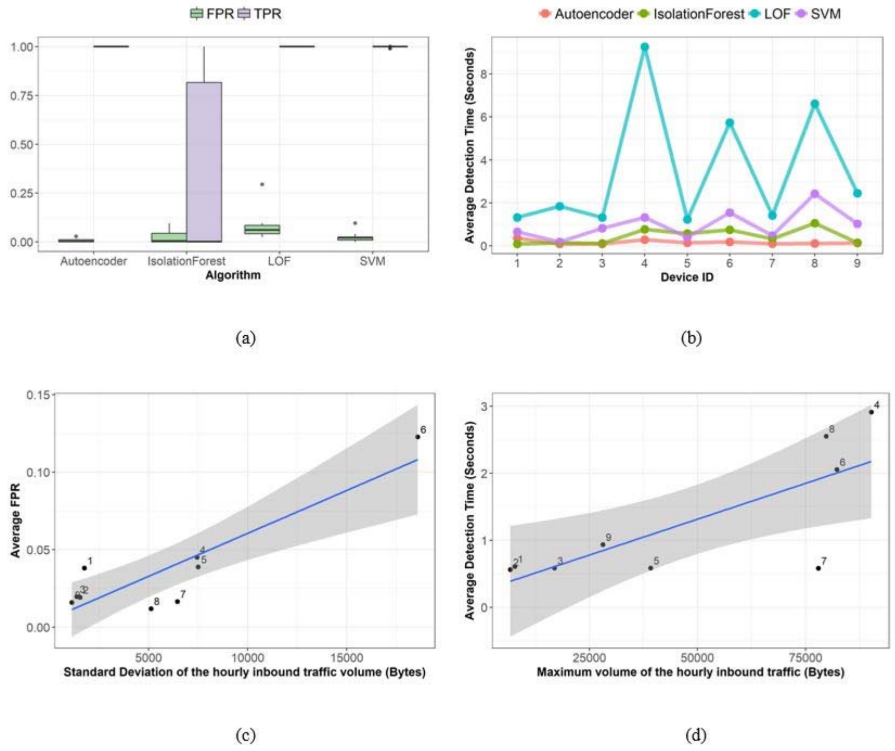
For training and optimization, we used Keras. Each autoencoder had an input layer whose dimension is equal to the number of features in the dataset (115). As noted by Ignacio Arnaldo and his colleagues[6] and by Li, Ma, and Jiao,[7] autoencoders effectively perform dimensionality reduction internally, such that the code layer between the encoder(s) and decoder(s) efficiently compresses the input layer and reflects its essential characteristics. In our experiments, four hidden layers of encoders were set at decreasing sizes of 75 percent, 50 percent, 33 percent, and 25 percent of the input layer's dimension. The next layers were decoders, with the same sizes as the encoders but with an increasing order (starting from 33 percent). Table 3 provides technical details about the training stage with a focus on the dataset properties, the optimized hyperparameters of the autoencoders, and the botnet infections.

Following autoencoder training and optimization, we used the same (benign) data to train three other algorithms commonly used[4] for anomaly detection: local outlier factor (LOF), one-class support vector machine (SVM), and IsolationForest. We optimized their hyperparameters exactly as we did for the autoencoders, including tr* and ws*. Finally, we executed all of the above attacks with the same duration via Mirai and BASHLITE's C&C servers. Then we extracted the features from the malicious data and appended each benign part of DS_{tst} (previously mentioned) to the respective malicious part of DS_{tst}, to form a single test dataset per IoT device with both benign and malicious instances.

The experimental results on DS_{tst} (see Figure 2) are promising:

- Our method succeeded in detecting every single attack launched by every compromised IoT device (TPR of 100 percent). As Figure 2a shows, LOF and SVM reached similar TPRs—much better than IsolationForest, which demonstrated an inferior and highly variable TPR.
- Our method also raised the fewest false alarms. It demonstrated a mean FPR of 0.007 ± 0.01, lower and more consistent than for SVM (0.026 ± 0.029), IsolationForest (0.027 ± 0.041), and LOF (0.086 ± 0.081).
- Our method required only 174 ± 212 ms to detect the attacks, and frequently much less time. As Figure 2b shows, for most of the evaluated IoT devices the average detection time of our method was lower than that of all the other methods. Assuming that the detection of attack-related anomalies can automatically trigger an immediate isolation of the compromised IoT device from the network, launched attacks can be stopped in less than a second. This is a substantial reduction from the typical duration of DDoS attacks,[19] whose distribution normally ranges between 20 and 90 seconds, plus a long tail where 10 percent of the attacks continue more than a day and 2 percent last longer than a month.

In terms of TPR, FPR and detection time, the deep autoencoders exemplified superiority for most devices. This is probably due to the ability of deep architectures to learn nonlinear structure mapping and approximate complex functions.[7] Additionally, the constrained complexity of deep autoencoders, imposed by the reduced dimensionality in the hidden layers, prevents them from learning the trivial identity function.[4] Therefore, deep autoencoders tend to fit common patterns better than uncommon ones. This is beneficial for IoT devices, as they normally are task-oriented, so their specified functionality should translate into few normal traffic patterns. Despite this tendency to fit common traffic patterns (generated by frequent actions), the autoencoders succeeded in capturing patterns of the infrequent actions (for example, boots) as well, demonstrated through low FPR. In real-world applications, the FPR can be adjusted by manipulating the tr* and/or ws*, though with some cost of TPR and detection times.



(a)

(b)

(c)

(d)

Figure 2. Experimental results using the test set. (a) Method detection accuracy. (b) Method detection time. (c) Average false positive rate (FPR) explained by traffic characteristics. (d) Detection time explained by traffic characteristics.

CONCLUSION

Although the autoencoders in our experiments obtained an FPR of zero on most IoT devices in a test set, the difference in the FPR among the remaining IoT devices led us to further analyze our data. We observed that the Philips B120N/10 baby monitor demonstrated the highest FPR relative to the other devices; it also produced the largest amount of traffic (see Table 3), so one could expect that the abundance of training instances would result in more robust machine learning models. However, this device also has the most diverse set of capabilities, as it is equipped with a two-way intercom function, motion detection, audio detection, and several other sensors for ambient light, temperature, and humidity. Given this, it might be more difficult to capture its normal behavior, and therefore future observations may be subject to more categorization errors.

Accordingly, we hypothesize that the difficulty in capturing the normal traffic behavior varies among IoT devices, and that this difficulty may be correlated with the device's capabilities and the network communications it normally produces. A similar notion was raised by Elisa Bertino and Nayeem Islam,[2] who argue that the specialized functionality of today's IoT devices leads to predictable behaviors. In turn, the ease of establishing baseline behaviors for IoT devices facilitates anomaly detection as a means of detecting attacks. To this end, interesting questions arise:

- Can the predictability of IoT devices' traffic behavior be quantified?
- Can the relation between the predictability level and the static features of IoT devices (for example, number and type of sensors, memory size, operating system) or dynamic features (for example, number of unique destination IPs per hour, variance of the ratio between outgoing and incoming traffic) be formalized?
- Can these features be ranked based on their influence on this predictability level?

We presume that the predictability of traffic behavior can be directly translated into performance measures of anomaly detection. For example, an IoT device with a high level of traffic predictability would make any anomalous action stand out, and thus the TPR should increase and detection times should decrease. For empirical validation we extracted static and dynamic features from the (benign) training set. Then we trained regression models to study these features' effect on the average FPR and detection times, obtained on the test set by the four detection methods we evaluated. Figures 2c and 2d depict our preliminary findings via the features found most significant. Figure 2c shows how an increase in the variability of inbound traffic translates (*p*-value = 0.019) into a larger average FPR. This makes sense, as lower predictability is prone to manifest through unpredictable (yet benign) traffic behaviors, falsely identified as anomalous. Figure 2d shows how an increase in the maximal volume of inbound traffic promotes (*p*-value = 0.001) longer detection times. As we optimize ws* to reach a 0 percent FPR on DS_{opt}, lower predictability leads to higher ws* (more instances for majority voting) and subsequently higher detection times.

Ultimately, a solid predictability score can be leveraged by large organizations to ensure network functionality and limit the impact that compromised devices might have on the network. That is, security policies might not allow the connection of IoT devices with low predictability scores to their networks, since they pose difficulties in attack detection. In our future work we plan to further define and investigate the subject of traffic predictability, both theoretically and empirically.

As another extension to the current study, we also plan to evaluate transfer learning techniques by assessing the accuracy of models trained on specific devices when they are applied to identical devices, possibly when connected to other organizational networks. This can help save time (for example, organizations can deploy models previously learned elsewhere, without the need to collect data and train the models themselves) and detect compromised IoT devices that have been contaminated prior to connecting to the organizational network, such that the organization has no benign data of them for model training.

ACKNOWLEDGMENTS

We thank Yan Lin Aung, Amit Subhashchandra Tambe, Simon Dzanashvili, and Tar Wolfson for their valuable contributions.

REFERENCES

1. C. Kolias et al., "DDoS in the IoT: Mirai and Other Botnets," *Computer*, vol. 50, no. 7, 2017, pp. 80–84.
2. E. Bertino and N. Islam, "Botnets and Internet of Things Security," *Computer*, vol. 50, no. 2, 2017, pp. 76–79.
3. R. Hallman et al., "IoDDoS—The Internet of Distributed Denial of Service Attacks: A Case Study of the Mirai Malware and IoT-Based Botnets," *Proc. 2nd Int'l Conf. Internet of Things, Big Data, and Security* (IoTBDS 17), 2017, pp. 47–58.
4. A. Tuor et al., "Deep Learning for Unsupervised Insider Threat Detection in Structured Cybersecurity Data Streams," *Proc. AAAI 2017 Workshop on Artificial Intelligence for Cybersecurity*, 2017; https://arxiv.org/pdf/1710.00811.pdf.
5. S. García, A. Zunino, and M. Campo, "Survey on Network-Based Botnet Detection Methods," *Security and Communication Networks*, vol. 7, no. 5, 2014, pp. 878–903.
6. I. Arnaldo et al., "Learning Representations for Log Data in Cybersecurity," *Proc. 1st Int'l Conf. Cyber Security Cryptography and Machine Learning* (CSCML 17), 2017, pp. 250–268.
7. Y. Li, R. Ma, and R. Jiao, "A Hybrid Malicious Code Detection Method Based on Deep Learning," *Int'l Journal Security and Its Applications*, vol. 9, no. 5, 2015, pp. 205–216.
8. K. Veeramachaneni et al., "AI^2: Training a Big Data Machine to Defend," *Proc. IEEE 2nd Int'l Conf. Big Data Security on Cloud, IEEE Int'l Conf. High Performance and Smart Computing, and IEEE Int'l Conf. Intelligent Data and Security* (BigDataSecurity-HPSC-IDS 16), 2016; doi.org/10.1109/BigDataSecurity-HPSC-IDS.2016.79.

9. M. Özçelik, N. Chalabianloo, and G. Gür, “Software-Defined Edge Defense against IoT-Based DDoS,” *Proc. 2017 IEEE Int’Conf. Computer and Information Technology* (CIT 17), 2017; doi.org/10.1109/CIT.2017.61.
10. H. Sedjelmaci, S.M. Senouci, and M. Al-Bahri, “A Lightweight Anomaly Detection Technique for Low-Resource IoT Devices: A Game-Theoretic Methodology,” *Proc. 2016 IEEE Int’l Conf. Communications* (ICC 16), 2016; doi.org/10.1109/ICC.2016.7510811.
11. H. Bostani and M. Sheikhan, “Hybrid of Anomaly-Based and Specification-Based IDS for Internet of Things Using Unsupervised OPF Based on MapReduce Approach,” *Computer Comm.*, vol. 98, 2017, pp. 52–71.
12. D. Midi et al., “Kalis—A System for Knowledge-Driven Adaptable Intrusion Detection for the Internet of Things,” *Proc. 2017 IEEE 37th Int’l Conf. Distributed Computing Systems* (ICDCS 17), 2017, pp. 656–666.
13. D.H. Summerville, K.M. Zach, and Y. Chen, “Ultra-Lightweight Deep Packet Anomaly Detection for Internet of Things Devices,” *Proc. 2015 IEEE 34th Int’l Performance Computing and Comm. Conf.* (IPCCC 15), 2015; doi.org/10.1109/PCCC.2015.7410342.
14. Y.M.P. Pa et al., “IoTPOT: A Novel Honeypot for Revealing Current IoT Threats,” *J. Information Processing*, vol. 24, no. 3, 2016, pp. 522–533.
15. I. Butun, B. Kantarci, and M. Erol-Kantarci, “Anomaly Detection and Privacy Preservation in Cloud-Centric Internet of Things,” *Proc. 2015 IEEE Int’l Conf. Communication Workshop* (ICCW 15), 2015, pp. 2610–2615.
16. S. Raza, L. Wallgren, and T. Voigt, “SVELTE: Real-Time Intrusion Detection in the Internet of Things,” *Ad Hoc Networks*, vol. 11, no. 8, 2013, pp. 2661–2674.
17. B.B. Zarpelo et al., “A Survey of Intrusion Detection in Internet of Things,” *J. Network and Computer Applications*, vol. 84, 2017, pp. 25–37.
18. Y. Yu, J. Long, and Z. Cai, “Network Intrusion Detection through Stacking Dilated Convolutional Autoencoders,” *Security and Communication Networks*, vol. 2017, 2017; doi.org/10.1155/2017/4184196.
19. N. Blenn, V. Ghiëtte, and C. Doerr, “Quantifying the Spectrum of Denial-of-Service Attacks through Internet Backscatter,” *Proc. 12th Int’l Conf. Availability, Reliability and Security* (ARES 17), 2017; doi.org/10.1145/3098954.3098985.

ABOUT THE AUTHORS

Yair Meidan is a PhD candidate in the Department of Software and Information Systems Engineering (SISE) at Ben-Gurion University of the Negev (BGU). His research interests include machine learning and IoT security. Contact him at yairme@post.bgu.ac.il.

Michael Bohadana is an MSc student in the SISE Department at BGU. His research interests include reverse engineering and IoT security. Contact him at bohadana@post.bgu.ac.il.

Yael Mathov is a PhD student in the SISE Department at BGU. Her research interests include IoT security and reverse engineering. Contact her at yaelmath@post.bgu.ac.il.

Yisroel Mirsky is a PhD candidate in the SISE Department at BGU. His research interests include machine learning and time-series anomaly detection. Contact him at yisroel@post.bgu.ac.il.

Asaf Shabtai is an assistant professor in the SISE Department at BGU. His research interests include computer and network security, and machine learning. Shabtai received a PhD in information systems from BGU. Contact him at shabtaia@bgu.ac.il.

Dominik Breitenbacher is a research assistant at the iTrust Centre of Cybersecurity at Singapore University of Technology and Design (SUTD). His research interests include IoT security and malware detection. Contact him at dominik@sutd.edu.sg.

Yuval Elovici is a professor in the SISE Department, director of the Telekom Innovation Laboratories, and head of the Cyber Security Research Center at BGU, as well as research director of iTrust at SUTD. His research interests include computer and network security, and machine learning. Elovici received a PhD in information systems from Tel-Aviv University. Contact him at yuval_elovici@sutd.edu.sg.