DeepPower: Non-intrusive and Deep Learning-based Detection of IoT Malware Using Power Side Channels

Fei Ding, Hongda Li, Feng Luo, Hongxin Hu, Long Cheng, Hai Xiao and Rong Ge
Clemson University
{feid, hongdal, luofeng, hongxih, lcheng2, haix, rge}@clemson.edu

ABSTRACT

The vulnerability of Internet of Things (IoT) devices to malware attacks poses huge challenges to current Internet security. The IoT malware attacks are usually composed of three stages: intrusion, infection and monetization. Existing approaches for IoT malware detection cannot effectively identify the executed malicious activities at intrusion and infection stages, and thus cannot help stop potential attacks timely. In this paper, we present DeepPower, a non-intrusive approach to infer malicious activities of IoT malware via analyzing power side-channel signals using deep learning. DeepPower first filters raw power signals of IoT devices to obtain suspicious signals, and then performs a fine-grained analysis on these signals to infer corresponding executed activities inside the devices. DeepPower determines whether there exists an ongoing malware infection by conducting a correlation analysis on these identified activities. We implement a prototype of DeepPower leveraging low-cost sensors and devices and evaluate the effectiveness of DeepPower against real-world IoT malware using commodity IoT devices. Our experimental results demonstrate that DeepPower is able to detect infection activities of different IoT malware with a high accuracy without any changes to the monitored devices.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**; • **Computer systems organization** → *Embedded hardware*; • **Computing methodologies** → *Neural networks*.

KEYWORDS

IoT; malware detection; non-intrusive; power side channels; deep learning

ACM Reference Format:
Fei Ding, Hongda Li, Feng Luo, Hongxin Hu, Long Cheng, Hai Xiao and Rong Ge. 2020. DeepPower: Non-intrusive and Deep Learning-based Detection of IoT Malware Using Power Side Channels. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security (ASIA CCS '20), October 5–9, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3320269.3384727

ASIA CCS '20, October 5–9, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-6750-9/20/10…$15.00
https://doi.org/10.1145/3320269.3384727

1 INTRODUCTION

As Internet of Things (IoT) has become an integral part of our lives, these devices provide great convenience and efficiency while also introduce unexpected risks and new threats. There have been lots of high-profile attacks against IoT devices being reported recently [3, 4, 47]. These compromised IoT devices can be leveraged to launch Distributed Denial of Service (DDoS) attacks [3] or Permanent Denial of Service (PDoS) attacks [47]. Existing research divides the IoT malware attacks into three stages: *intrusion*, *infection*, and *monetization* [44, 46]. Through an in-depth analysis of IoT malware, we find that there exist several common activities in many IoT malware attacks, such as usage exploitation, malware downloading, permission change, and binary execution. These activities are the most basic activities to launch IoT malware attacks, and even unknown malware variants need to perform these activities. In order to minimize the loss caused by IoT malware, it is important to identify these malicious activities as early as possible.

In general, malware detection approaches can be divided into two categories: network-based and host-based approaches. The network traffic analysis [19, 42] has be commonly used for protecting IoT systems. When malicious traffic patterns are detected in IoT networks, warnings are generated. However, such a solution cannot examine the details of activities that are executed in IoT devices, which are important to enable fine-grained detection of IoT malware. Another possible solution is to detect these activities by host-based security mechanisms [2, 51]. Unfortunately, IoT devices are usually resource-constrained and always hard to support *full-fledged* detection solution. Even worse, these devices are designed based on different principles and mechanisms, which make it difficult to apply one general detection approach to all kinds of devices. Therefore, it is not practical to directly leverage traditional host-based security solutions (*e.g.*, antivirus) to secure IoT devices [61]. A feasible detection solution for IoT malware should be able to effectively monitor the infection activities while require minimal changes to the software or hardware of IoT devices. Intuitively, the *non-intrusive* monitoring (*i.e.*, side-channel analysis) of a system's runtime activities is such an ideal host-based method for the detection of IoT malware.

Non-intrusive power side-channel approaches have been recently developed to distinguish malicious and legitimate activities based on the power signals (*i.e.*, power consumption measurements) of embedded devices [9, 33, 60]. For example, WattsUpDoc [9] is an anomaly-based detection method to identify malware on medical devices. However, existing detection methods based on normal and abnormal patterns cannot be directly used for effective detection of IoT malware. Usually, they only consider deviations from normal baselines as abnormal but ignore the internal details of these anomalous behaviors. That said, they cannot differentiate

specific malicious activities and attack stages. Simply categorizing anomalous activities as an attack without considering different stages of attacks could inevitably cause a large number of false positives. Therefore, it is necessary to design a *fine-grained* approach to further identify executed activities (*i.e.*, commands) from these anomalous cases, and achieve an effective detection of IoT malware.

The IoT malware detection based on fine-grained power side-channel analysis requires us to address several critical technical challenges. First, there is a lack of in-depth analysis of activities of IoT malware due to the lack of relevant malware samples. A systematic analysis of IoT malware is important to identify their common activities. Second, though power signals provide a promising solution, they are notoriously noisy due to the fact that most IoT devices rely on the alternating current. It is unclear whether the power signals associated with different activities in IoT devices are distinguishable. It is even challenging to detect multiple consecutive activities as there are no clear boundaries in their corresponding power signals. Third, even if we can accurately infer activities from power signals, it is non-trivial to determine whether these activities are belonging to IoT malware attacks or benign usage.

In this paper, we address the above challenges and present DeepPower, a novel framework for non-intrusive detection of IoT malware based fine-grained power signals analysis. First, we identify common activity patterns of IoT malware from a set of known open-source malware and reverse engineering reports. Second, DeepPower performs an effective data preprocessing to tackle the problem of noisy power signals. By using deep learning techniques, DeepPower infers executed activities from corresponding power signals. Third, DeepPower performs a correlation analysis of these activities to determine whether there exist malware infections in IoT devices.

We make the following contributions in this paper:

- We perform a study of IoT malware infection process based on open-source IoT malware and reverse engineering reports. We categorize the malicious activities of IoT malware into one intrusion and five infection states, and identify common activities of IoT malware in the process of infecting devices.
- We develop effective data preprocessing methods to minimize the influence of noise in power signals, including a wavelet denoising method to remove background noise, and a feature extraction method to obtain unique signal patterns. These methods collectively improve the performance of the following malware detection task.
- We propose a new framework, DeepPower, for the non-intrusive detection of infection activities of IoT malware using power signals based on deep learning. DeepPower begins with a fast signal detection to identify suspicious activities, and then utilizes an attention-based Seq2Seq model to achieve a fine-grained analysis of these suspicious signals. After inferring these activities, DeepPower conducts a correlation analysis between activities and infection states, and outputs the final detection results.
- We implement the DeepPower system by using low-cost sensors and devices, and evaluate its effectiveness for non-intrusive detection based on real-world IoT malware using commodity IoT devices. Our experiment results show that DeepPower can detect the infection activities of different IoT malware with high accuracy (90.4% detection rate on average) without any changes to IoT devices.

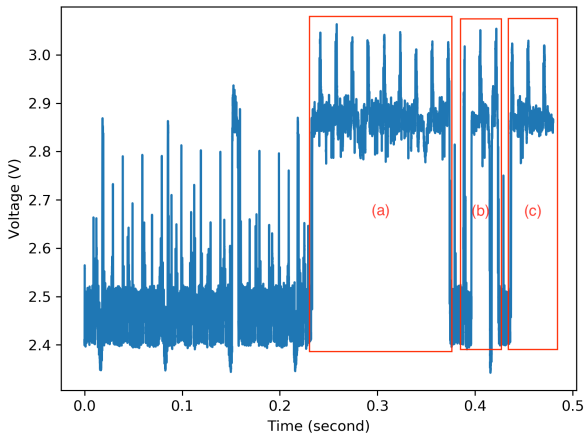

Figure 1: The power signals collected on a D-Link IP Camera during the infection process of IoT malware Mirai [41]. The red boxes (a), (b) and (c) indicate three different infection activities: login attempts, environment preparation and file downloading.

Roadmap. Section 2 describes the threat model and deployment scenarios. Section 3 presents our study of intrusion and infection process of IoT malware. Section 4 describes our DeepPower design. Implementation and evaluation details are discussed in Section 5. Section 6 reviews related work and Section 7 concludes this paper.

2 THREAT MODEL AND DEPLOYMENT SCENARIOS

Due to the software and hardware constraints in IoT devices, it still lacks an effective security mechanism to protect these devices. Many devices with known and unknown vulnerabilities (*e.g.*, default/weak passwords, unpatched bugs) are still being exploited. It is highly desirable for these resource-constrained devices to detect malware activities in a non-intrusive manner. The power side-channel analysis is one of ideal choices. Previous research shows that power signal analysis can achieve high detection rates of anomalous behaviors [9], but it inevitably results in some false positives due to noisy power signals. A traditional anomaly detection solution, however, doesn't further analyze these abnormal behaviors and thus only provides a coarse-grained detection.

Taking the IoT malware Mirai [41] as an example, as shown in Figure 1, during its infection process on a D-Link IP Camera, there exist at least three different waveforms (a), (b) and (c), which indicate three different infection activities: *login attempt*, *environment preparation*, and *file downloading*. The traditional anomaly detection method can only categorize these three different waveforms as abnormal cases, but cannot tell us what activities behind these

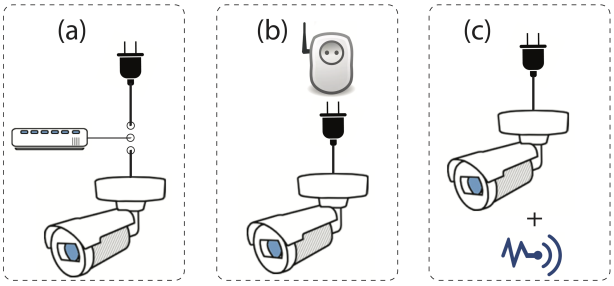

Figure 2: Three deployment scenarios of our proposed detection solution: (a) implementing it as an independent monitor; (b) integrating it into the Smart Plug; and (c) adding a power sensor to an IoT device.

suspicious signals. In fact, it is obvious that these three power waveforms are different. Thus, it is desirable to perform a fine-grained analysis of these waveforms, and identify corresponding activities that cause these abnormal waveforms.

2.1 Threat Model

In this paper, we focus on Linux-based IoT devices, which are the main target of IoT malware [3, 10]. Remote attackers may exploit different vulnerabilities [61] (such as device firmware flaws, unprotected authentication, and vulnerable applications) for compromising IoT devices. In the intrusion process, the attacker can get a remote shell and execute several commands to infect devices or cause a severe system damage. The infection process has several states/steps for preparing the environment, and downloading and executing the malware binary. We don't consider the situation where multiple attackers attack the same device simultaneously and execute their own infection activities in parallel. We aim at the detection of IoT malware attacks, which perform malicious or destructive behaviors on devices by injecting commands. Although most of the IoT malware targets the Linux-based devices running BusyBox [29], which combines many common UNIX utilities into a single executable, our design is not limited to these Linux-based IoT devices running BusyBox, and generally applicable to any Linux-based devices.

2.2 Deployment Scenarios

Our research goal is to design a low cost and non-intrusive IoT malware detection solution, which can support various flexible deployment scenarios. It has a strong advantage of being inexpensively deployable for *critical* IoT devices, *i.e.*, IP cameras, rather than for every IoT device. There are several feasible deployment solutions in real world as shown in Figure 2. Figure 2 (a) demonstrates a deployment scenario where our detection solution can be implemented as an independent monitoring system, which is provided by third-party vendors. Such a plugin system is placed between an IoT device that needs to be protected and an AC-power adaptor. For the second deployment scenario shown in Figure 2 (b), since there are already Smart Plug products on the market that can monitor the energy consumption of devices [5], it would be easy to integrate our solution into such Smart Plug products. In fact, it is also possible to extend Smart Plug products to monitor multiple IoT devices at the same time. As shown in Figure 2 (c) for the third deployment scenario, IoT device vendors could integrate power sensors into some IoT devices, which means that it would be much easier to monitor the power consumption of those devices. Actually, internal power sensors have already been integrated into today's smartphones. Previous studies have shown the possibility to use the power consumption for detecting malware in smartphones [40].

3 IDENTIFYING INTRUSION AND INFECTION ACTIVITIES OF IOT MALWARE

In this work, we collect and analyze open-source IoT malware including Mirai [41], Linux.wifatch [58], and Lightaidra [16], and reverse engineering reports from security researchers [3, 23]. We observe that there exist two typical stages when IoT malware attempts to compromise a device. The first stage is *intrusion process*, at which attackers try to log into IoT devices. The second one is *infection process*, at which several activities are executed to prepare the environment, and download and execute malware binary files to infect the devices. Through the infection process, several activities are used to infect vulnerable devices.

3.1 Intrusion Process of IoT Malware

After the intrusion process, the attacker is able to get a remote shell and execute several commands to infect devices or cause a severe damage. Usually, IoT devices are exposed to a number of security attacks, including weak/default passwords, lack of encryption, backdoor, and zero-day vulnerabilities. As listed in Table 1, we categorize the intrusion process into two types: vulnerable authentication, and unpatched or zero-day vulnerabilities.

Vulnerable Authentication: Authentication plays an essential role in IoT security solutions. A poor authentication mechanism can lead to a device to be compromised easily and thus attackers can get its remote shell. For example, most of IoT malware compromise and infect devices through weak/default passwords on Telnet, and hijack them to launch DDoS attacks. An increasingly popular cyber-attack, Permanent Denial of Service (PDoS) attack, also uses the same security flaws as Mirai [47] to compromise the victim's hardware.

Unpatched or Zero-day Vulnerabilities: Another category of intrusion is based on exploiting unpatched or new vulnerabilities, which can also be used to launch DDoS attacks, such as OKIRU/SATORI [7] and Amnesia [59]. IoTroop [8] is a new type of IoT botnet that is designed to scan device's vulnerabilities instead of brute forcing passwords. BASHLITE is a malware which infects Linux-based IoT devices by exploiting the GNU Bash vulnerability known as ShellShock (CVE-2014-6271) [30]. It allows attackers to execute malicious code via a crafted environment [12]. TheMoon [32] is a worm, which was firstly discovered in 2014 by exploiting a vulnerability (CVE-2014-9853) in ASUS routers. It allows attackers to bypass the authentication process and execute malicious commands via a NET_CMD_ID_MANU_CMD packet to UDP port 9999 [13].

Table 1 also lists several top security threats, such as buffer overflow and backdoor. For example, Linux.Darlloz [15] is a worm by utilizing the PHP 'php-cgi' Information Disclosure Vulnerability

Table 1: The intrusion summary of reported IoT malware.

Intrusion Type	Attack Type	Security Issue	IoT Malware Attacks
Vulnerable Authentication	DDoS attacks	Weak/default passwords or bypass	Mirai [4], Remaiten [35]
	PDoS attacks	Weak/default passwords	BrickerBot [47]
	Brute force attacks	Not limit password attempts	Linux/NyaDrop [39], LuaBot [39]
Unpatched Vulnerabilities	DDoS attacks	CVE-2017-17215,RCE, Shellshock	Amnesia [59], IoTroop [8]
	Crypto-currency mining	PHP Vulnerability (CVE-2012-1823)	Linux.Darlloz [4]
	Buffer overflow attacks	Insecure HANP Protocol	DSP-W215 Smart Plug [14]
	Backdoor attacks	Pre-auth RCE vulnerability of IP Camera	Affect 1250+ models [27]

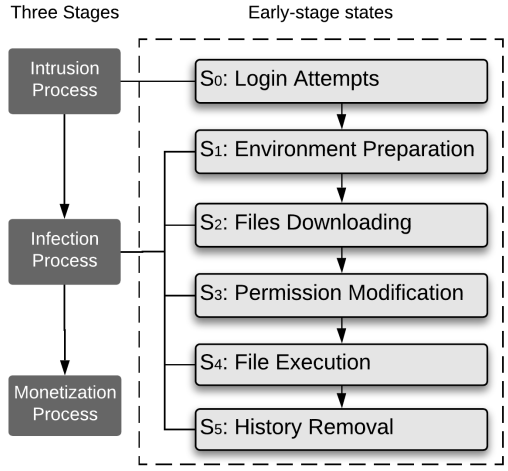

Figure 3: Three stages of IoT malware attacks [44, 46]. The intrusion and infection models (S_0-S_5) in our analysis.

(CVE-2012-1823), which allows remote attackers to execute arbitrary code by placing command-line options in the query string [11]. In addition, the D-Link DSP-W215 Smart Plug can be exploited to get a root shell by stack overflow, since it uses the insecure Home Network Administration Protocol (HNAP) [14]. Some new vulnerabilities about backdoor root account (CVE-2017-8224) have already been reported, which at least affect 1250+ Wireless IP Camera (P2) models [27]. Through the intrusion process, the attacker is able to get a remote shell and execute malicious commands to infect devices or cause a severe damage.

3.2 Infection Process of IoT Malware

Identifying the infection process is a critical step for IoT malware detection, which requires a comprehensive analysis of real-world attacks on IoT devices. Based on the infection activities observed from the IoT malware source code and the reverse engineering reports, and command patterns observed by IoTPOT [46], we summarize the commonly used commands and their corresponding states as shown in Table 2. It illustrates that multiple steps need to be executed to infect devices successfully. For instance, the wget command is used to download a bash script or malware binary to victim's device. The chmod command may be used to change the access permission to the malicious file. After these two operations, the malware begins to run an execute command of the downloaded malware and become active. As shown in Table 2, most of the IoT malware take advantage of these three commands to compromise vulnerable devices. The rm command and kill or killall command also appear often in our collection. The former one is used to delete all malicious files when they are successfully executed. The latter one may be used to terminate the telnet, SSH, and HTTP services to prevent victim's device to be compromised again.

Take the IoT Mirai malware for example, the infection log shows that it will execute several commands after successful login. The first step is to check and customize the environment. It executes "/bin/busybox ps;" command to display the processes running on the system, and tries to find a folder with the write and execute permissions to download malware files by using "/bin/busybox cat /proc/mounts;" command. When it successfully executes echo, cat, and rm commands, a suitable folder is found to save the malware files. Then it continues to execute the following commands: "cd /; /bin/busybox cp /bin/echo dvrHelper; >dvrHelper; /bin/busybox chmod 777 dvrHelper;", to test whether it can create a new executable file. Before it starts to download the malware binary, it uses the command "/bin/busybox cat /bin/echo\r\n" to parse the ELF (Executable and LinkableFormat) header and obtain the architecture information. Next, the malware tries to download the binary file by the command wget and change the permision of downloaded file by chmod command. After successful execution, it removes the infection history on the device as quickly as possible.

However, IoT malware families are diverse and have versatile ways to achieve the same purpose by different commands. For example, to find a folder with write and execution permissions, IoT malware can utilize the command sequence of "cd /tmp || cd /var/run || cd /mnt || cd /root || cd /;". Some malware may use the command combination of "cat /proc/mounts | grep r". In addition, a new IoT malware botnet, called IoTroop [48], uses the "echo" command to convey the vulnerability information of devices to attackers for a further infection. Thus, it is necessary to model the infection process to analyze the activities across various IoT malware families.

As shown in Figure 3, we further divide the infection process into five states. As a result, our analysis typically focuses the intrusion and infection processes, which includes six states, *login attempts*, *environment preparation*, *downloading files*, *permission modification*, *executing files*, and *deleting infection history*. Note that the intrusion and infection stages we studied are host-based and different from the infection dialog process of network traffic by Gu *et al.* [19]. Compared to other recent studies on IoT malware analysis, focusing on malware execution behaviors of open-source botnets [3, 4], our

Table 2: The commonly used commands (operations) during the infection process. S_1~S_5 are different infection states shown in Figure 3.

	S_1	S_1	S_1	S_1	S_1	S_1	S_1	S_2	S_3	S_4	S_5	S_5	
IoT Malware	cd	cat	cp	echo	mkdir	ps	grep	wget	chmod	execute	rm	kill	References
Linux.wifatch								•	•		•	•	Source code [58]
Lightaidra			•					•	•	•			Source code [16]
Mirai		•	•	•	•	•		•	•	•	•	•	Source code [41]
Psyb0t								•	•	•		•	Reverse Eng. [28]
Chuck Norris								•	•	•	•		Reverse Eng. [28]
Kaiten	•							•	•	•	•		Reverse Eng. [20]
Linux.Darlloz	•		•	•	•			•	•	•	•		Reverse Eng. [15]
BASHLITE	•			•				•		•			Reverse Eng. [26]
XOR.DDoS	•	•				•	•	•	•	•	•		Reverse Eng. [36]
IRCTelnet					•			•	•	•	•		Reverse Eng. [38]
LizKebab	•			•				•		•			Reverse Eng. [37]
Remaiten		•		•				•	•	•		•	Reverse Eng. [35]
TheMoon	•							•	•	•	•		Reverse Eng. [32]
NyaDrop	•	•		•						•	•		Reverse Eng. [39]
Hajime	•							•	•	•			Reverse Eng. [55]
Amnesia		•		•						•	•		Reverse Eng. [59]
BrickerBot		•									•		Reverse Eng. [47]
PERSIRAI.A								•	•	•			Reverse Eng. [53]
IMEIJ.A								•	•	•			Reverse Eng. [52]
DvrHelper	•								•	•			Reverse Eng. [54]
Okiru								•	•	•			Reverse Eng. [7]

analysis is concentrated on the common infection activities, where different variants of malware are likely to share common infection behaviors. Therefore, our method is able to detect new malware variants. Based on the in-depth understanding of malware infection activities, we next discuss how to accurately detect them using power signals.

4 DEEPPOWER DESIGN

A fine-grained analysis of IoT malware allows us to understand what kinds of activities on devices are expected to appear during the IoT malware infection process. The major challenge is how to accurately detect these activities on resource-constrained IoT devices. In this section, we introduce the system architecture of DeepPower and the technical details behind it.

4.1 System Overview

DeepPower focuses on the detection of infection activities of IoT malware attacks, which are more common and general than other activities. Figure 4 illustrates the overall system architecture of DeepPower. It consists of four phases: (1) detection of suspicious signals; (2) preprocessing of suspicious signals; (3) inferring activities from suspicious signals; and (4) infection process modeling and correlation analysis of inferred activities. In phase (3), it contains training (model learning) stage and testing (malware detection) stage.

The first phase takes the power signals of the monitored device as input and quickly detects suspicious power signals. The purpose of this phase is to filter out most of the signals and only retain a small number of suspicious signals for further fine-grained analysis. Considering that the suspicious signals contain a lot of noises, in the second phase, DeepPower performs an effective data preprocessing to reduce noises and extract useful features that will be used by a sequence to sequence (Seq2Seq) model. In the third phase, the Seq2Seq model is developed to infer the activities from the preprocessed suspicious signals. We choose the Seq2Seq model because it has been applied with a great success in various tasks, such as neural machine translation [34] and automatic speech recognition (ASR) [6], which are similar to the activity prediction problem in this work. Finally, to determine whether a malware infection process exists in the device, the last phase performs a correlation analysis of the inferred activities against the infection process model, and calculates a weighted score for each state.

4.2 Detection of Suspicious Signals

Recently, many machine learning techniques have been proposed to perform anomaly detection tasks based on power signals [9], which model permissible activities and detects deviations. When power signals are identified as outliers, it is unclear whether these deviations are caused by anomalous activities or by noises, since power signals always contain lots of random and periodic noises introduced by the power supply (*i.e.*, 50/60 Hz AC). Therefore, it is reasonable to consider these deviations as suspicious activities, and perform a further fine-grained analysis of them. In this phase, we use a deep autoencoder to detect suspicious activities with low complexity and in an unsupervised manner.

A deep autoencoder is a multi-layer feed-forward neural network, which encodes the input into low-dimensional representation based on non-linear transformations, and reconstructs the original

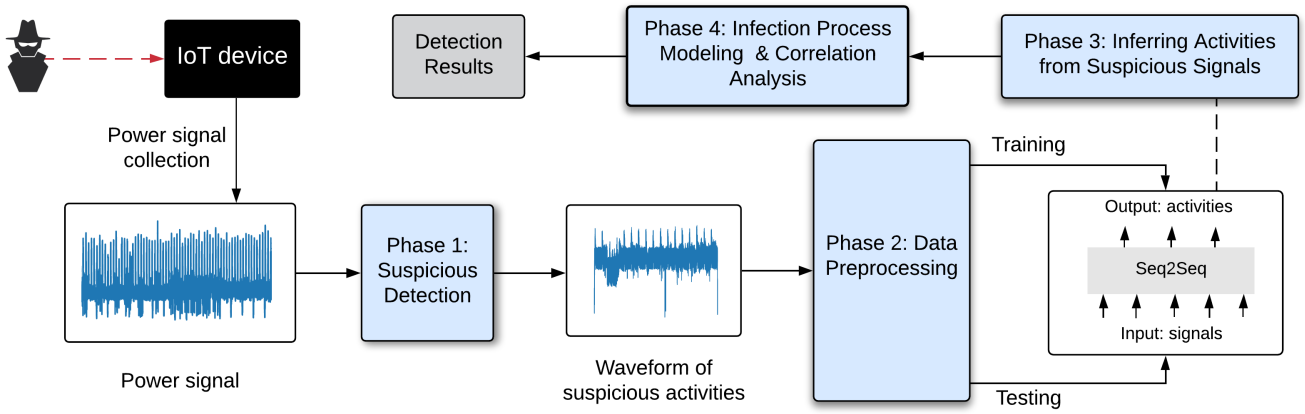

Figure 4: DeepPower system overview.

input based on the representation. Unfortunately, the outliers and noise in power signals reduce the representation quality and bring great challenges to the standard denoising autoencoders, which requires clean training data. Herein, we utilize a Robust Deep Autoencoder (RDA) model [62] to first isolate the suspicious parts in power signals and then train an autoencoder on the remaining portion. This model is suitable for handling different noise intensities and patterns in heterogeneous devices. The power signals X are splitted into two parts, $X = L_D + S$, where L_D represents the part that can be accurately reconstructed by an autoencoder, and S indicates the outliers and noise that are difficult to reconstruct. We use the following optimization objective:

$$\begin{aligned} min_\theta ||L_D - D_\theta(E_\theta(L_D))||_2 + \lambda||S||_1 \\ s.t.\ X - L_D - S = 0, \end{aligned} \tag{1}$$

where X is the input power signals, E_θ is an Encoder, and D_θ is a Decoder. S contains the suspicious portions, and L_D represents the remaining parts. $||\cdot||_2$ denotes the L^2-norm and $||\cdot||_1$ denotes L^1-norm (the absolute-value norm). λ is a hyperparameter that controls the sparsity level of S. A small λ will isolate more parts of raw power signals into S as suspicious portions.

4.3 Preprocessing of Suspicious Signals

The objective of this phase (Phase 2 in Figure 4) is to preprocess suspicious signals to obtain high-quality features for activity inference. There exist two issues that need to be solved. First, the AC power supply causes periodic appearance of strong peaks during (a), (b) and (c) activities in Figure 1. Our experiments show that these peaks randomly appear anywhere in a periodic way. It is necessary to remove these peaks from the raw signals before the feature extraction. Second, due to the intrinsic properties of power signals, the time domain-based signal processing method cannot discriminate the fine-grained characteristics of different activities accurately. We need an effective solution to extract unique features from suspicious signals.

Our data preprocessing method is described as follows. First, a 100-point simple moving average (SMA) filter is applied to reduce the effect of small fluctuations and smooth the power signals. Figure 5(a) shows the power signals of *wget* command by using the SMA filter. Second, we apply the wavelet denoising method to remove the strong background noise generated from the AC power, as shown in Figure 5(b), while keeping as much useful information as possible (Figure 5(c)). Third, we transform the filtered signals into mel-scaled spectrograms and use them as the input of the later prediction tasks. These frequency domain-based features make the power signals of different activities more distinguishable while reducing input dimensionality and computational overhead. To verify whether this preprocessing method is effective, we perform a classification task on individual activities of power signals. The classification result also demonstrates the distinguishability of the individual activity's power signal. Furthermore, while analyzing the situation of multiple consecutive activities, there are no boundaries between their waveforms of different activities. A natural choice is to apply a Seq2Seq model to infer activities from power signals.

4.4 Inferring Activities from Suspicious Signals

After preprocessing the suspicious signals, this phase (Phase 3 in Figure 4) utilizes deep learning techniques to infer activities from the signals. It aims to sequentially infer the activities sequence from their corresponding power signals. First, DeepPower employs an effective attention-based Seq2Seq architecture with Long Short-Term Memory (LSTM) networks [24, 34] to figure out the problem about no boundaries between the power signals of different activities. This Seq2Seq model allows it to focus on the specific parts of the power signals to output one activity in the target sequence every time. To achieve a better prediction performance, DeepPower utilizes the convolutional layer to capture high-level features from the processed suspicious signals and feeds these high-level features into the Seq2Seq model. Second, to infer activities from suspicious signals, we need to build the training dataset and testing dataset to train and test this Seq2Seq model. However, given a power signal, it is unclear that how many activities it contains, and which part of power signal should each activity correspond to. It is difficult to build the dataset of time-aligned pairs of activities sequence (output) and power signals (input). We address this dataset issue through two main steps. First, we concatenate the power signal of individual

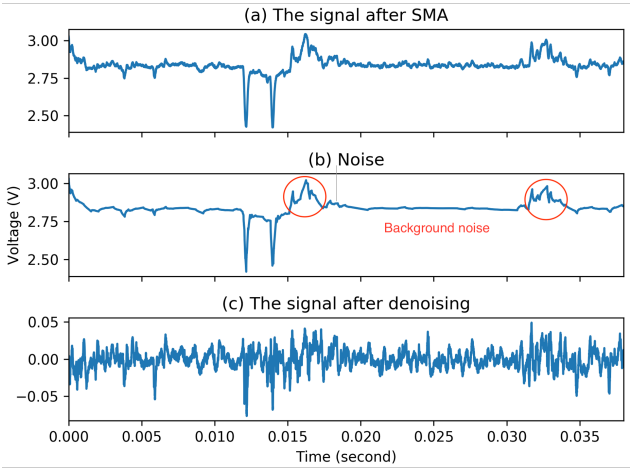

Figure 5: The wavelet denoising of *wget* command on D-Link IP Camera: (a) the signal after SMA; (b) background noise; and (c) the signal after denoising. (a), (b) and (c) share the same x coordinate.

activity to create the power signals of multiple activities. Second, this Seq2Seq task requires to further process the training dataset by labeling a power signal multiple times. The power signal of one activity can usually be converted to dozens of frames for feature extractions, which means that dozens of frame inputs correspond to one output. This extreme imbalance between the input and output lengths makes it difficult for the Seq2Seq model to accurately align the short target sequence to the long power frame sequence. Therefore, instead of setting one label per activity in target sequence, we set each activity with multiple repeated labels according to the power signal length of this activity. This labeling method enables the model to predict each activity of the target sequence and the boundaries of each activity. In addition, it also makes our model focus more on the activities that the corresponding signals have longer lengths, which are more energy-consuming and sensitive.

For a given power waveform sample $X = \{x_1, ..., x_n\}$, we map it to a target sequence $y = \{y_1, ..., y_m\}$. The variable n is the length of the power waveform of one or more activities. The m means the number of activities contained in the power waveform. The target sentence y is generated one target y_t at a time based on the probability:

$$P(y_t|y_{<t}, c_t) = softmax(W_s\tilde{h}_t) \tag{2}$$

where the attentional hidden state $\tilde{h}_t$ is computed as:

$$\tilde{h}_t = \tanh(W_c[c_t; h_t]) \tag{3}$$

h_t is the hidden state of the Decoder, c_t is the input context vector, and [;] indicates the concatenation operation. W_s and W_c are trainable parameters.

$$c_t = \sum_s \alpha_t(s)\bar{h}_s \tag{4}$$

Here, the context vector c_t is derived from all the hidden states of the Encoder. At each time step t, the score function is used to compare the target hidden state h_t and all the encoder hidden state $\bar{h}_s$, and the result is normalized to derive a variable-length alignment weight vector α_t:

$$\alpha_t(s) = \frac{\exp(score(h_t, \bar{h}_s)))}{\sum_{i=1}^{l} \exp(score(h_t, \bar{h}_i)))} \tag{5}$$

where the alignment weight α_t indicates that the parts in the input are the most likely to help in predicting the output in the target sequence.

4.5 Infection Process Modeling & Correlation Analysis

To detect IoT malware via the infection process, this phase (Phase 4 in Figure 4) first models the infection process and then performs a correlation analysis of the inferred activities based on the infection process model. The malware analysis (Section 3) shows that IoT malware usually use the following different approaches to infect devices: (1) downloading scripts and execution; (2) creating new files and execution; (3) direct injection of commands; and (4) running different commands interactively. We also find that even given the sequence of the same commands, some attackers run the entire sequence multiple times to meet different architecture requirements, while others tend to repeatedly run parts of the command sequence to achieve the same goal. In addition, there are multiple activities that can achieve the same purpose. For instance, both of the *wget* command and the *tftp* command can be used to download the malware to the device for execution. To handle such complexity and diversity, we map all these activities in target sequence to corresponding infection states according to Table 2, which enables us to better understand the semantics of the entire target sequence. Then, we correlate these states from inferred activities against our infection model (Figure 3) to identify whether an IoT device is infected. According to Table 2, there are different combinations of states that are required for the infection of an IoT malware. To transform these combinations into a scoring system, we employ a regression model to estimate the state weights and a threshold value.

5 IMPLEMENTATION AND EVALUATION

In this section, we implement a prototype of DeepPower and evaluate our method on real-world settings. We perform a number of experiments to answer the following questions:

- Can we implement DeepPower in a plug-and-play manner using low-cost sensor and devices (Section 5.1) ?
- Can DeepPower detect the power signals of suspicious activities by the autoencoder model (Section 5.2) ?
- Can the power signals associated with different activities be distinguishable and what is the classification performance of the individual activity (Section 5.3) ?
- What is the DeepPower performance on identifying multiple activities based on power signals during cross-device prediction (Section 5.4) ?
- What is the DeepPower performance in detecting infection activities of real-world IoT malware based on fine-grained power signals analysis (Section 5.5) ?
- Can DeepPower be resistant to potential evasion attacks during the detection of infection activities (Section 5.6) ?

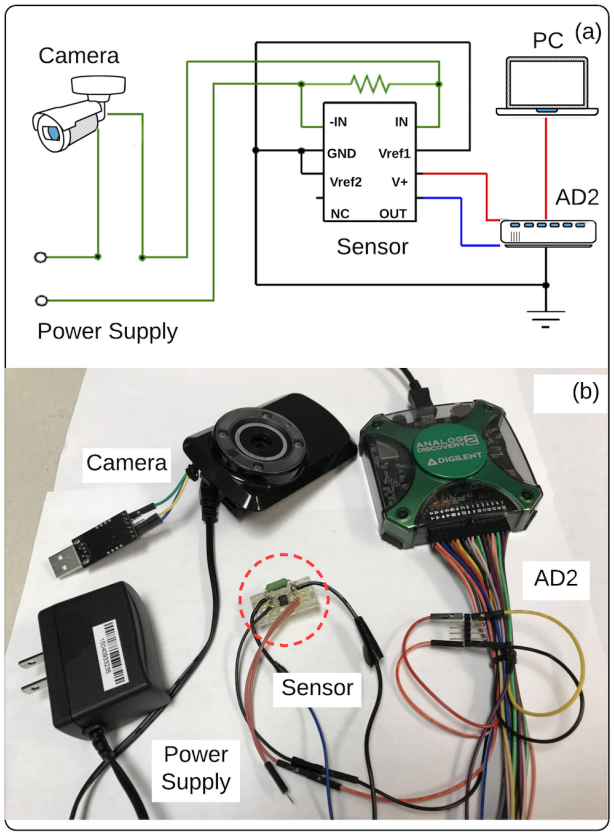

Figure 6: The wiring diagram (a) and experimental setup (b) for DeepPower system (AD2: Analog Discovery2), the green lines represent the sensor wires, the red lines denote the power supply, the black lines are the ground wires, and the blue wires connect the AD2 with the sensors. The power supply, shunt, and AD2 board need to be properly grounded.

5.1 Experimental Setup

Most IoT devices have separate alternative current to direct current (AC-DC) converters, which can serve as the power monitoring point without any hardware or software modification. In this work, we measure the power consumption of the entire IoT device, by inserting a precision resistor (0.3 Ω) between the load and the power supply. We choose the Analog AD8210 current sensor and the USB oscilloscope Analog Discovery2 to monitor power usage with 1-MHz sampling rate. Figure 6 shows the wiring diagram and experimental setup for our system. The entire system has its own separate power supply and does not interfere with the power consumption of the monitored device. There are two main reasons for choosing such configuration. First, this plug-and-play setting ensures that it can be easily deployed to monitor the critical and sensitive IoT devices. Second, choosing a low sampling rate (1 MHz), which is three orders of magnitude lower than other work [33], can reduce the cost of the entire implementation, making our approach more practical.

We choose IP cameras to evaluate the detection solution based on the fact that the most known attacks, *i.e.*, Mirai botnet, targeted such devices [3, 4]. IP cameras contain representative hardware modules and complex software applications, such as image sensor and light sensor, which require special software for remote viewing, motion detection and day/night mode shift. In other words, the power signals of these devices are complex and dynamic, and not as fixed and repeated as SCADA [9], PLC [60] or MCU [33]. Table 3 lists the detailed information of three real-world IP cameras used in this work. Without loss of generality, we install the OpenWrt 15.05.1 firmware, the mjpg_streamer driver and light sensor daemon on D-Link DCS-934L (D-934L) to ensure that it works normally. This firmware allows us to test the default password attack. The camera of ESCAM G02 (E-G02) has a vulnerability that allows us to test on the original firmware. Xiaofang 1S (X-1S) allows us to run a modified firmware on it through SD-card hacks [18]. In our experiments, these IoT devices represent three different noise levels of power signals: strong, medium and weak, which ensure broad coverage of our evaluation.

Based on hardware configuration discussed above, we collect the power signals of these IoT devices in a real-world environment. The power signals generated by all relevant software are reflected in the background of power signals. For each time window of the power signals, we first perform a fast detection of the suspicious activities to identify potential anomalies. Then, these suspicious signals are further processed to obtain the waveform of suspicious activities through smoothed z-score algorithm and the Run Length Encoding (RLE) approach. To identify what type of activities are included in a suspicious signal waveform, we need to perform a fine-grained analysis. Since the signals are noisy, we first reduce the noise by the Wavelet method. Our experiments show that the Wavelet configuration of db2 type and 7 level can achieve the satisfactory noise reduction results. Then, to perform the spectral feature extraction, we adopt the following parameters: the length of FFT window is 2048, the number of samples between successive frames is 512 and n_mels is 32. All experiments are conducted on a server with 2 NVIDIA K40 GPUs with the CUDA 9.0 toolkit installed.

The main problem that DeepPower is expected to address is the fine-grained detection of malware infections on resource-constrained devices. We do not assume that DeepPower is only for less complex devices. Considering that IoT devices are becoming increasingly complicated, with more functionalities equipped, this may diversify the background information of power signals. In this case, we can add more relevant signals to the training data set to ensure that DeepPower can still predict reasonable results.

5.2 Detection of Suspicious Activities

To detect the suspicious activities, we implement the autoencoder model using TensorFlow. For the Encoder E_θ and Decoder D_θ, we follow multi-layer neural network $E_\theta(L_D) = Z(L_D) = logit(WL_D + b_E)$ and $E_\theta(L_D) = \overline{L}_D = logit(W^T Z(L_D) + b_D)$ [31], where W projects the input dimension to a lower dimensional space, W^T indicates a projection from the low-dimensional space back to the original input dimension, b_E and b_D are the bias terms. We use one hidden layer that projects the input signals from 1000 dimensions to 256 dimensions. The hyperparameter λ is set to 20 to isolate a

Table 3: The summary of real-world IoT devices being tested in our experiments.

IoT Device	CPU Arch	Firmware	Power Usage	Noise Level	Attack Type
D-Link (D-934L)	MIPS	OpenWRT rt305x	5V, 1A	Noise-dominant	Default password
ESCAM (E-G02)	ARM	Original version	5V, 1.6A	Medium-intensity	CGI vulnerabilities
Xiaofang (X-1S)	MIPS	Modified version	5V, 1A	Low-noise	SD-card hacks

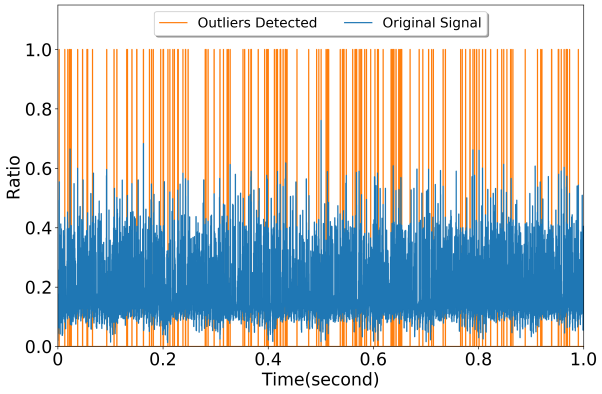

Figure 7: The detection of suspicious activities by autoencoder model on DCS-934L.

small portion of power signals as the noise or outliers. To train this model, we sample the power signals with 1-second window size, and collect 100 normal power signal samples for each device. Then these 1-second samples are downsampled to 100 kHz from 1 MHz. After normalizing the signals of each device, we mix all the signals together as the training data to train an autoencoder model that can be applied on different devices. Figure 7 demonstrates the detection results on DCS-934L. The outliers detected by the model are caused by the suspicious activities. The following experiments will map these suspicious power signals to their corresponding activities.

5.3 Discriminative Analysis of Activities

To infer activities from power signals, we first perform signal classification task to evaluate the distinguishability of the individual command's power signal after data preprocessing. This task is conducted by ResNet-50 Convolutional Neural Networks (CNN) model [22] in PyTorch. This experiment is implemented on the DCS-834L IP Camera. We should notice that the signal length of each activity is different, and if the entire signals of the activities are fed into the neural networks, the model will learn to identify these activities mainly by their signal lengths rather than the patterns of their signals. Therefore, we split the power signal of each activity to multiple 10 ms chunks without overlap, where the chunks from the same activity have the same label. After this step, our training data set and testing data set have 78,220 and 7,309 fix-length power signal samples, respectively. In our experiments, we set the batch size to 128, weight decay to 1e-2, learning rate to 1e-4, and the max epoch to 70. To train the model, we use the stochastic gradient descent (SGD) optimizer and reduce the learning rate by a factor of 10 when the validation loss is not improved for five consecutive epochs.

Table 4 shows the confusion matrix of 11 different activities for the classification task. Compared with the commands in Table 2, we group the *chmod*, *rm*, *mkdir* and *cp* command into the METADATA operation. On the monitored device, there exists an intrusion type to obtain a root shell by remote attackers, which is labeled as login activity. If there is a non-existent or unrecognized command, the corresponding power signal is labeled as UNKNOWN activity. As shown in the table, most of the power signal chunks are classified correctly as their true activity labels. The model achieves the highest prediction accuracy on UNKNOWN activity. For the signals with longer activities, *e.g.*, *grep* and *wget*, the prediction results are not as good as other activities. A possible reason is that some CPU and IO operations involved in one activity are similar to other activities. Because we split the power signal of each activity into equal-length chunks, some signal chunks from different activities have highly similar characteristics. Still, these different patterns of the same activity can be used to identify the corresponding activity label. The classification results show that the power signals of the different individual activities are indeed distinguishable.

However, this classification model with the fix-length chunks cannot be used to directly predict multiple activities from power signals for the following reasons. First, the main reason is that power signals are very complicated in real-world. Given a suspicious power signal, it is difficult to determine how many activities it contains and the location of each activity. Thus the fix-length model can result in poor classification performance since it divides the power activities into one chunk. Second, even though it can utilize the sliding windows to avoid the problem of inappropriate segmentation, this solution has high overhead and introduces new problems, such as how to combine the results of each chunk into a sequence. Third, this model outputs the result of each chuck independently, which will result in the loss of the dependency information of the neighboring chunks and the global information of the entire sequence. Therefore, we apply the Seq2Seq model to directly map the power signals to the executed activities sequence according to the order in which they are executed. This model can not only overcome the above challenges, but also be trained end-to-end.

5.4 Multiple Activities Prediction

To predict multiple activities based on power signals, we implement the Seq2Seq model using TensorFlow. The Encoder employs 3 convolutional layers with 16-length filter and 2 strides to extract high-level features, and a one-layer Bidirectional LSTM neural networks with 256-dim hidden units to capture contextual information in the input. The Decoder employs a one-layer LSTM neural network with 256-dim hidden units to output the target sequence in an autoregressive manner. To train the model, we use the SGD optimizer and a learning rate of 1e-06. The dropout scheme with a

Table 4: The confusion matrix (%) of the individual activity's classification on DCS-934L.

	cat	cd	exe	echo	ps	grep	login	kill	wget	M	U
cat	**73.86**	0.59	4.55	1.19	0.79	3.17	2.38	0.79	5.15	5.94	1.58
cd	2.27	**85.45**	3.18	5.45	0.	0.	0.	2.73	0.45	0.	0.45
exe	3.26	0.65	**76.02**	1.14	0.16	2.45	6.36	2.28	0.98	5.22	1.47
echo	1.31	3.93	6.11	**72.93**	0.	3.06	1.75	9.17	0.44	0.87	0.44
ps	1.00	0.	0.50	0.10	**82.5**	9.90	2.00	0.10	1.60	1.20	1.10
grep	2.50	0.40	3.10	0.90	14.30	**67.30**	4.90	0.40	4.00	1.20	1.00
login	0.	0.	2.00	0.40	0.90	4.30	**83.90**	0.30	2.30	2.70	3.20
kill	2.47	2.06	6.58	12.76	0.	2.06	1.65	**66.67**	3.29	2.06	0.41
wget	3.29	0.09	1.46	0.	2.56	3.20	4.47	1.19	**74.98**	6.39	2.37
M	4.21	0.34	8.08	0.46	0.34	1.37	6.71	1.02	2.28	**72.92**	2.28
U	1.59	0.	0.40	1.39	0.40	0.20	3.37	0.20	1.79	2.78	**87.90**

Table 5: The cross-device prediction results of several true multiple activities on three IP Cameras (L: login, P: ps, U: UNKNOWN, M: METADATA, C: cat, D: cd, W: wget).

True seq	Prediction (%)
L	**L (92.2)**, P (7.3), WM (0.5), WMU (0.1)
PU	**P (63.5)**, PU (15.4), WU (23.1)
CU	**CU (82.5)**, MU (17.5)
DMMU	**DMMU (50.8)**, MMU (12.6), MM (36.6)
CMMU	**CMM (80.1)**, MM (19.9)
WUU	**WUU (66.4)**, WMU (33.6)

20% dropout rate and the early-stopping method are used to avoid overfitting. We also clip gradients by the global norm to prevent exploding gradient, and the max value is set to 5.0. To address the issue of lack of time-aligned labeled dataset, we concatenate the power signal of individual activity to synthesize consecutive power signals of multiple activities. Then, we utilize the power signals of real multiple activities to fine-tune the pretained model from the synthesized data set. The training dataset has 21,601 samples and the validation dataset has 1,731 samples. To verify the feasibility of the synthesized dataset, we execute multiple activities on the device and collect their corresponding signals as testing data, which is composed of 1,800 samples.

Table 5 shows the cross-device prediction results of the Seq2Seq model from the power signals of true multiple activities on three IP Cameras. During this cross-device prediction, we normalize the signals from different devices and mix them together to train a single model for different devices. For the short sequence (L, PU and CU), our model is able to predict multiple activities with a high accuracy. The reason is that short sequences have a greater chance of being transferred between the signal patterns with the same activity. And it is relatively easy for the model to capture context dependency of their relationships when there are less activities. For the long sequence (DWWU, CMMU and WUU), our model can correctly predict the vast majority of activities. It indicates that the model tends to focus on the unique features of signals while ignoring irrelevant parts introduced by data synthesis.

We further demonstrate the alignment results between the power signals and the executed activities using Mirai malware. Figure 8 shows the prediction results of DeepPower. DeepPower can exactly locate and recognize each command of the corresponding power signal. During the prediction of Figure 8(a), we find that the first 30 ms power signal is mapped to the wget command, and the remaining part can be identified as two METADATA operations. In addition, the model estimates a high probability that the power signal could be mapped to the login operation in Figure 8(b). The experimental results show that it is feasible to utilize power analysis to detect the infection activities of Mirai malware. Our approach is able to identify each activity and decide its location from the corresponding power signal. To further confirm the effectiveness of our solution, we evaluate this method on more real-world IoT malware.

5.5 Infection Detection of Real-world Malware

After inferring the activities from the power signals, we map the activity sequence to the state sequence. The correlation analysis of the state sequence enables us to understand the semantics of the entire sequence and reach a reasonable conclusion about the malware infection. We employ a weighted threshold scoring method that aggregates the total scores S of each detected state (Figure 3). DeepPower observes a sufficient number of states and calculates a minimum threshold ϵ to determine whethe an IoT device is infected. S represents how likely these states belong to the malware infection process, and can be expressed as:

$$S = \sum_{i=1}^{5} w_i \cdot S_i, \tag{6}$$

where w_i indicates whether a state S_i is detected (1 and 0 represent presence and absence, respectively), and S_i represents the score of the state i.

Table 6: The scores of individual states.

State	S1	S2	S3	S4	S5
Score	0.1067	0.2845	0.1421	0.3197	0.1470

To make our scoring system reasonable, we collect 887 Linux malware Bash scripts from VirusShare [56] and 974 Linux Bash scripts for normal usage from GitHub, and map the commands in these scripts to the corresponding states according to Table 2. Then, the desired weight of each state is estimated from the regression model as shown in Table 6. This table doesn't include the state

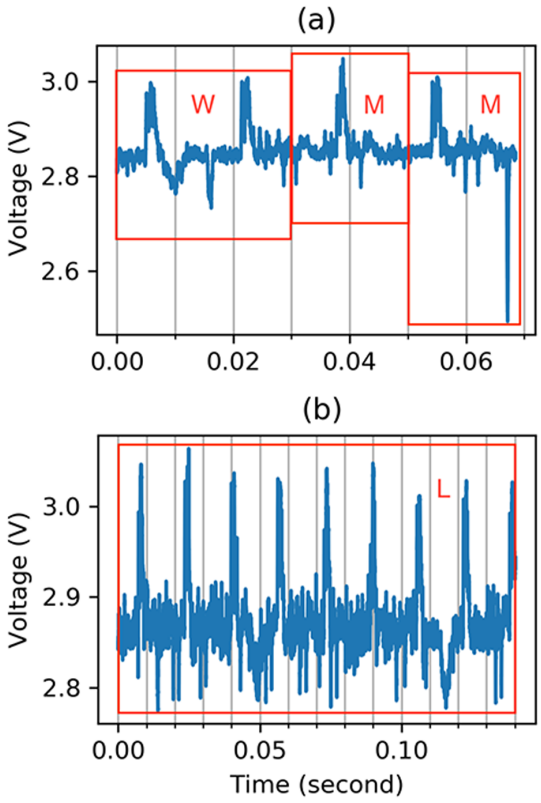

Figure 8: The alignment between power signals and executed activities of IoT Mirai malware (L: login, W: wget, M: METADATA), (a) and (b) are for different activities.

S_0, because the Bash scripts don't contain information for intrusion processes. Thus, our correlation analysis only considers five states (S_1-S_5) contained in the infection process. In fact, DeepPower can accurately identify the state S_0. In section 5.3, we still consider this state during inferring activities from signals, but ignore its contribution to the detection results when performing correlation analysis.

We apply these weights to estimate the threshold between the malware Bash scripts and normal Bash scripts. As illustrated in Figure 9, we observe that all but 8 malware scripts score are below 0.6, and all but 38 normal scripts score are above 0.6. Thus, it is reasonable to set the threshold to 0.6 in our system, which only introduces **1.3%** false positives and **0.3%** false negatives. In order to further reduce the false positive rate, we perform a more comprehensive analysis of these false positive scripts. We find that these scripts can be divided into two categories: installation scrips and upgrading scripts according to their purposes of usage. For the former one, it is reasonable to mark the installation process as known normal activities and exclude it from the detection results. For the latter one, taking the open-source OpenWrt system as an example [45], the most obvious difference between a normal upgrading process and a malware infection process is that the upgrading process typically utilize two more commands: the sha256sum command for checking

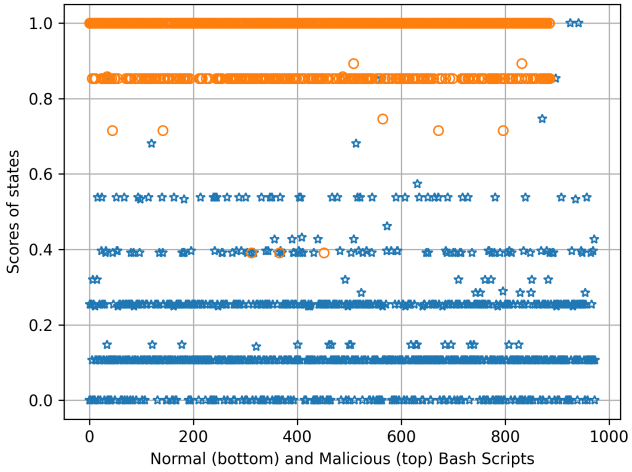

Figure 9: The scoring plot of 887 malware Bash scripts (yellow color) and 974 normal Bash scripts (blue color).

the firmware checksum and the reboot command for restarting during upgrading. Because the reboot command will free up RAM, this command is unlikely to appear in the malware infection process. This unique upgrading pattern can be used by our system to eliminate false positives caused by firmware upgrades.

Based on the score of each state (Table 6) and the threshold, we evaluate DeepPower on 5 representative real-world IoT malware [56] that represent 5 different malware types. Especially, Mirai and Lizkebab are two typical open-source malware, which infects IoT devices in order to launch DDoS. BASHLITE is another open-source malware, which affects IoT devices using security bugs in Unix Bash Shell [30]. Tsunami is a reported IoT bot malware that can launch web-based attacks [57]. Bourne-Again shell (BASH) Script is a general Malware DownLoader with size of 1KB and can be used to download and execute any real-world malware. Table 7 illustrates our detection results. The detection experiments are repeated for 160 times for each malware. The results show an average accuracy of 90.4% for all 3 devices on 5 representative real-world malware, suggesting that our approach is promising for the detection of real-world malware.

Table 7: The detection accuracy of real-world IoT malware's infection activities of DeepPower on three devices.

IoT Malware	D-934L	E-G02	X-1S	Average
Mirai	92.9%	94.6%	90.6%	**92.7%**
Tsunami	92.6%	91.8%	88.5%	**91.0%**
Lizkebab	90.6%	89.7%	88.1%	**89.5%**
BASHLITE	88.9%	88.2%	87.5%	**88.2%**
BASH Script	92.3%	89.1%	89.7%	**90.4%**

To demonstrate the benefit of our fine-grained detection approach, we compare DeepPower with other side-channel based detection solutions in detecting infection activities of real-world IoT malware. Many existing side-channel analysis approaches mainly

focus on the anomaly detection for Programmable Logic Controllers (PLC) [60] and microcontroller unit (MCU) [33]. WattsUpDoc [9] is the most closest work to DeepPower, which uses power side channels for anomaly detection to identify malware in Windows-based embedded systems. WattsUpDoc simply categorizes anomalous activities as a malware without considering internal details of those anomalous activities. In contrast, DeepPower could conduct a fine-grained analysis of suspicious signals to output specific executed activities. We reproduce WattsUpDoc and evaluate its effectiveness of detecting IoT Malware, Mirai, on three IoT devices. Table 8 shows our comparison results, which indicate that our DeepPower system could achieve a significant improvement in both of true positive rate (TPR) and false positive rate (FPR), which are 92.7% and 2.9%, respectively, comparing with WattsUpDoc that can only achieve 84.2% TPR and 15.3% FPR in detecting Mirai's infection activities.

Table 8: The detection accuracy of Mirai's infection activities from two different side-channel methods.

Method	TPR	FPR
WattsUpDoc	84.2%	15.3%
DeepPower	92.7%	2.9%

5.6 Evasion Analysis

DeepPower focuses on the detection of infection activities based on power signals. Benefiting from the fine-grained analysis, our method could be resistant to potential attacks. To ensure the robustness of our solution, we examine several possible evasion attacks.

First, the attackers may attempt to evade the detection system by injecting several random or invalid commands. To maintain the effectiveness of their attacks, the random commands injected by the attackers will not change the patterns of the original infection. Because our fine-grained analysis is based on the sequence to sequence translation, which is from power signals to commands sequence, both of injected random commands and original commands can be identified by the order, in which they are executed. By conducting a correlation analysis of these identified commands, DeepPower can still focus on the original infection patterns and identify these activities as malicious. Considering the case of invalid commands injection, all of these invalid commands can be identified as unknown commands by the corresponding signals. Therefore, our method is still able to filter out these unknown commands and obtain the original infection patterns. We perform some preliminary experiments to test the robustness of our approach with respect to the detection of IoT malware Mirai. We randomly inject 1 and 3 valid/invalid commands and repeat each experiment 50 times. As shown in Figure 10, there is no significant change in the detection accuracy of our approach in each case after injecting random/invalid commands.

Second, it may be subject to an evasion attack that leverages customized binaries instead of system commands to reach the same purpose. For example, by analyzing the infection process of Mirai and Hajime, we find that echo command can also be used to transfer malware to a victim's device, in addition to wget and tftp. Especially, the Hajime malware [1] first uses echo command to drop a hex string to a very small file as an ELF binary. Then this ELF binary is executed to connect a pre-defined server to download the real malware. In other words, to achieve such an evasion attack, attackers still need to drop and execute their own tools on the victims' devices. Our previous experiments show that these dropping and execution activities are still detectable even when the attackers rely only on minimal operations.

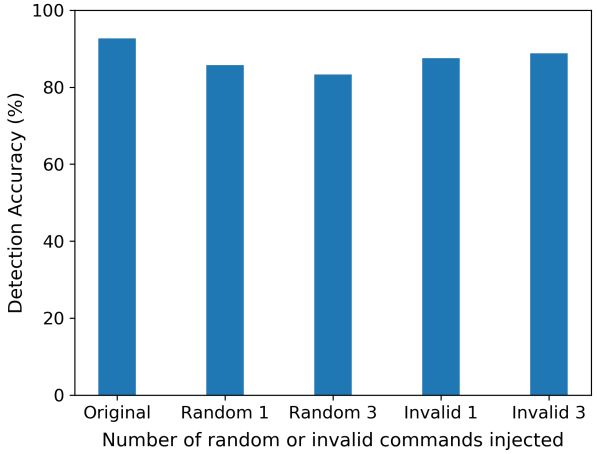

Figure 10: The comparison between original detection and detection after injecting 1 and 3 random/invalid commands.

6 RELATED WORK

The increasing threats of IoT malware have attracted significant attention in security research community. Despite network-based detection method is still a dominant research direction, there are other solutions that focus on low-overhead host-based detection.

6.1 Network-based Solution

Network-based solutions have been commonly used for protecting IoT systems [17, 25, 42, 44]. Gu *et al.* [19] presented BotHunter, a dialog correlation method that utilizes malware-specific signatures to recognize the malware infection for botnet detection in traditional networks. However, IoT network traffic is device-specific and depends on different environmental settings. Due to the diversity of IoT devices and manufacturers, it is impractical or non-scalable to create malware signatures [61]. It is also challenging to define normal baselines of IoT networks for anomaly detection. In addition, IoT Botnets have continued to evolve and adapt to the advanced techniques. For example, a newer version of Mirai, DvrHelper, is the first malware designed to bypass an anti-DDoS solution by using challenge-response policies and shared Google reCAPTCHA response token [54]. Although new network-based detection methods for IoT malware are constantly being proposed [42, 44], they still cannot identify more detailed activities that occur in IoT devices. Our detection approach is able to discovery detailed information about infection activities of IoT malware, which could be a promising complement to existing network-based solutions.

6.2 Host-based Solution

To secure IoT devices, one preferred solution is to update and patch buggy firmware for these vulnerable devices. However, due to lack of suitable facilities, it is difficult to keep track of the available patches and apply them to all unpatched devices. Also, not all devices are compatible with the available updates due to their outdated hardware. Considering the constraints of limited resources, only a few studies have focused on host-based IoT security solution. Sun *et al.* [51] proposed a cloud-based detection with reversible sketch for resource-constrained IoT devices to improve the security of the devices. Abbas *et al.* [2] presented a simple signature-based method that leverages a subset of signatures to detect a group of malware for IoT devices. Su *et al.* [50] proposed a light-weight detection method for IoT malware, based on a local and cloud-based malware detector. However, all those solutions require the installation of software in IoT devices, but not all devices can afford such runtime overhead.

6.3 Side-channel Analysis

The side-channel analysis has been recently developed to distinguish malicious and legitimate behaviors based on the power consumption of Supervisory Control And Data Acquisition (SCADA) devices [9], Programmable Logic Controllers (PLC) [60], and MicroController Unit (MCU) [33]. Especially, WattsUpDoc [9] utilizes an anomaly-based analysis to detect malware on medical devices. However, most of the existing side-channel analysis approaches focus on differentiating normal and abnormal patterns. They don't consider the internal details of anomalous cases and cannot identify the anomalous cases as specific malicious activities (*i.e.*, commands). To achieve the goal of non-intrusive detection, DeepPower can effectively infer specific infection activities and conduct a correlation analysis among activities to output final detection results. Apart from the power signals, Electromagnetic (EM) signals [21, 43] and radio-frequency (RF) emissions [49] have also been used for the anomaly detection of program execution. Our work selects the power side-channel signal because of its favorable properties: easy to collect, less susceptible to environmental influence, and closely correlated with the system's workload.

7 CONCLUSION AND FUTURE WORK

In this paper, we have conducted a systematic study of the IoT malware infection process. Based on an in-depth understanding of IoT malware infection patterns, we have introduced DeepPower, a non-intrusive and deep learning-based detection solution based on power side-channel analysis to discover IoT malware infection. We have first verified whether the processed signals of different individual activities are distinguishable. Based on our experiments, the mel-scaled spectrogram features are used to distinguish fine-grained power characteristics of activities accurately. Then, we have trained a Seq2Seq model to infer activities from the power signals. To examine its feasibility, we have conducted experimental verification on the real-world malware infection process, and our experimental results show that most of the infection processes can be accurately detected. For our future work, we will further improve the detection accuracy of IoT malware by choosing more effective detection models. Besides, due to the lack of effective score calculation method for state S_0, our current work doesn't consider the influence of this state on the detection results. An interesting future work is to estimate the score of S_0, and also combine DeepPower with network traffic analysis for a more comprehensive detection of IoT malware.

ACKNOWLEDGEMENT

This material is based upon work supported in part by the National Science Foundation (NSF) under Grant No. 1846291, 1700499, 2031002, 1642143, and 1759856, and the U. S. National Institute of Food and Agriculture (NIFA) under Grant No. 2017-70016-26051. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF and NIFA.

REFERENCES

[1] IoT Malware Droppers (Mirai and Hajime). https://0x00sec.org/t/iot-malware-droppers-mirai-and-hajime/1966, 2017.
[2] Muhamed Fauzi Bin Abbas and Thambipillai Srikanthan. Low-complexity signature-based malware detection for iot devices. In *International Conference on Applications and Techniques in Information Security*, pages 181–189, Singapore, 2017. Springer Singapore.
[3] Kishore Angrishi. Turning internet of things (iot) into internet of vulnerabilities (iov): Iot botnets. *arXiv preprint arXiv:1702.03681*, 2017.
[4] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the mirai botnet. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*, pages 1093–1110, 2017.
[5] Wemo Insight Smart Plug. https://www.belkin.com/us/p/P-F7C029/, 2019.
[6] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4960–4964, 2016.
[7] Huawei Home Routers in Botnet Recruitment. https://research.checkpoint.com/good-zero-day-skiddie/, 2017.
[8] IoTroop Botnet: The Full Investigation. https://research.checkpoint.com/iotroop-botnet-full-investigation/, 2017.
[9] Shane S Clark, Benjamin Ransford, Amir Rahmati, Shane Guineau, Jacob Sorber, Wenyuan Xu, Kevin Fu, A Rahmati, M Salajegheh, D Holcomb, et al. Wattsupdoc: Power side channels to nonintrusively discover untargeted malware on embedded medical devices. In *HealthTech*, 2013.
[10] Emanuele Cozzi, Mariano Graziano, Yanick Fratantonio, and Davide Balzarotti. Understanding linux malware. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 161–175. IEEE, 2018.
[11] CVE-2012-1823. https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2012-1823, 2012.
[12] CVE-2014-6271. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271, 2014.
[13] CVE-2014-9583. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-9583, 2014.
[14] Hacking the D-Link DSP-W215 Smart Plug. http://www.devttys0.com/2014/05/hacking-the-d-link-dsp-w215-smart-plug/, 2014.
[15] Hey Zollard, leave my Internet of Things alone! http://www.deependresearch.org/2013/12/hey-zollard-leave-my-internet-of-things.html, 2013.
[16] Fei Ding. Iot malware. https://github.com/ifding/iot-malware, 2017.
[17] Rohan Doshi, Noah Apthorpe, and Nick Feamster. Machine learning ddos detection for consumer internet of things devices. *arXiv preprint arXiv:1804.04159*, 2018.
[18] Xiaomi-Dafang-Hacks. https://github.com/EliasKotlyar/Xiaomi-Dafang-Hacks, 2019.
[19] Guofei Gu, Phillip A Porras, Vinod Yegneswaran, Martin W Fong, and Wenke Lee. Bothunter: Detecting malware infection through ids-driven dialog correlation. In *USENIX Security Symposium*, volume 7, pages 1–16, 2007.
[20] Michael Haag. Kaiten - Linux Backdoor. http://blog.michaelhaag.org/2013/12/kaiten-linux-backdoor.html, 2013.
[21] Yi Han, Sriharsha Etigowni, Hua Liu, Saman Zonouz, and Athina Petropulu. Watch me, but don't touch me! contactless control flow monitoring via electromagnetic emanations. In *Proceedings of the 2017 ACM SIGSAC Conference on*

Computer and Communications Security, pages 1095–1108. ACM, 2017.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[23] Ben Herzberg, Dima Bekerman, and Igal Zeifman. Breaking Down Mirai: An IoT DDoS Botnet Analysis. https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html, 2016.

[24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[25] Ionut Indre and Camelia Lemnaru. Detection and prevention system against cyber attacks and botnet malware for information systems and internet of things. In *2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 175–182. IEEE, 2016.

[26] Rhena Inocencio. BASHLITE Affects Devices Running on BusyBox. http://blog.trendmicro.com/trendlabs-security-intelligence/bashlite-affects-devices-running-on-busybox/, 2014.

[27] Multiple vulnerabilities found in Wireless IP Camera (P2P) WIFICAM cameras and vulnerabilities in custom http server. https://pierrekim.github.io/blog/2017-03-08-camera-goahead-0day.html, 2017.

[28] Marta Janus. Heads of the Hydra. Malware for Network Devices. https://securelist.com/heads-of-the-hydra-malware-for-network-devices/36396/, 2011.

[29] James A Jerkins. Motivating a market or regulatory solution to iot insecurity with the mirai botnet code. In *Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual*, pages 1–5. IEEE, 2017.

[30] Swati Khandelwal. BASHLITE Malware leverages ShellShock Bug to Hijack Devices Running BusyBox. https://thehackernews.com/2014/11/bashlite-malware-leverages-shellshock.html, 2014.

[31] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[32] Bing Liu. TheMoon - A P2P botnet targeting Home Routers. https://blog.fortinet.com/2016/10/20/themoon-a-p2p-botnet-targeting-home-routers, 2016.

[33] Yannan Liu, Lingxiao Wei, Zhe Zhou, Kehuan Zhang, Wenyuan Xu, and Qiang Xu. On code execution tracking via power side-channel. In *Proceedings of the ACM SIGSAC conference on computer and communications security*, pages 1019–1031, 2016.

[34] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[35] Michal Malik and Marc-Etienne M.Léveillé. Meet Remaiten - a Linux bot on steroids targeting routers and potentially other IoT devices. https://www.welivesecurity.com/2016/03/30/meet-remaiten-a-linux-bot-on-steroids-targeting-routers-and-potentially-other-iot-devices/, 2016.

[36] MMD-0037-2015 - A bad Shellshock & Linux/XOR.DDoS CNC "under the hood". http://blog.malwaremustdie.org/2015/07/mmd-0037-2015-bad-shellshock.html, 2015.

[37] MMD-0052-2016 - Overview of "SkidDDoS" ELF++ IRC Botnet. http://blog.malwaremustdie.org/2016/02/mmd-0052-2016-skidddos-elf-distribution.html, 2016.

[38] MMD-0059-2016 - Linux/IRCTelnet (new Aidra) - A DDoS botnet aims IoT w/ IPv6 ready. http://blog.malwaremustdie.org/2016/10/mmd-0059-2016-linuxirctelnet-new-ddos.html, 2016.

[39] MMD-0058-2016 - Linux/NyaDrop - a linux MIPS IoT bad news. http://blog.malwaremustdie.org/2016/10/mmd-0058-2016-elf-linuxnyadrop.html, 2017.

[40] Alessio Merlo, Mauro Migliardi, and Paolo Fontanelli. On energy-based profiling of malware in android. In *High Performance Computing & Simulation (HPCS), 2014 International Conference on*, pages 535–542. IEEE, 2014.

[41] Leaked mirai source code for research/ioc development purposes. https://github.com/jgamblin/Mirai-Source-Code, 2016.

[42] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: An ensemble of autoencoders for online network intrusion detection. *arXiv preprint arXiv:1802.09089*, 2018.

[43] Alireza Nazari, Nader Sehatbakhsh, Monjur Alam, Alenka Zajic, and Milos Prvulovic. Eddie: Em-based detection of deviations in program execution. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pages 333–346. ACM, 2017.

[44] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Minh Hoang Dang, N Asokan, and Ahmad-Reza Sadeghi. Dïot: A federated self-learning anomaly detection system for iot. *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2019.

[45] Upgrading OpenWrt firmware via CLI. https://openwrt.org/docs/guide-user/installation/sysupgrade.cli, 2019.

[46] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. Iotpot: analysing the rise of iot compromises. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*, 2015.

[47] "brickerbot" results in pdos attack.

[48] Why the World is Under the Spell of IoT_Reaper. https://blog.radware.com/security/2017/10/iot_reaper-botnet/, 2017.

[49] Samuel Stone and Michael Temple. Radio-frequency-based anomaly detection for programmable logic controllers in the critical infrastructure. *International Journal of Critical Infrastructure Protection*, 5(2):66–73, 2012.

[50] Jiawei Su, Danilo Vasconcellos Vargas, Sanjiva Prasad, Daniele Sgandurra, Yaokai Feng, and Kouichi Sakurai. Lightweight classification of iot malware based on image recognition. *arXiv preprint arXiv:1802.03714*, 2018.

[51] Hao Sun, Xiaofeng Wang, Rajkumar Buyya, and Jinshu Su. Cloudeyes: Cloud-based malware detection with reversible sketch for resource-constrained internet of things (iot) devices. *Software: Practice and Experience*, 47(3):421–441, 2017.

[52] New Linux Malware Exploits CGI Vulnerability. http://blog.trendmicro.com/trendlabs-security-intelligence/new-linux-malware-exploits-cgi-vulnerability/, 2017.

[53] Persirai: New Internet of Things (IoT) Botnet Targets IP Cameras. http://blog.trendmicro.com/trendlabs-security-intelligence/persirai-new-internet-things-iot-botnet-targets-ip-cameras/, 2017.

[54] The Reigning King of IP Camera Botnets and its Challengers. http://blog.trendmicro.com/trendlabs-security-intelligence/reigning-king-ip-camera-botnets-challengers/, 2017.

[55] Jornt van der Wiel, Vicente Diaz, Yury Namestnikov, and Konstantin Zykov. https://securelist.com/hajime-the-mysterious-evolving-botnet/78160/, 2017.

[56] VirusShare.com - Because Sharing is Caring. https://virusshare.com/, 2019.

[57] Zack Whittaker. Hacker explains how he put "backdoor" in hundreds of Linux Mint downloads. https://www.zdnet.com/article/hacker-hundreds-were-tricked-into-installing-linux-mint-backdoor/, 2016.

[58] Linux.Wifatch source repository. https://gitlab.com/rav7teif/linux.wifatch, 2015.

[59] Claud Xiao and Cong Zheng. New IoT/Linux Malware Targets DVRs, Forms Botnet. https://researchcenter.paloaltonetworks.com/2017/04/unit42-new-iotlinux-malware-targets-dvrs-forms-botnet/, 2017.

[60] Yu-jun Xiao, Wen-yuan Xu, Zhen-hua Jia, Zhuo-ran Ma, and Dong-lian Qi. Nipad: a non-invasive power-based anomaly detection scheme for programmable logic controllers. *Frontiers of Information Technology & Electronic Engineering*, 18(4):519–534, 2017.

[61] Tianlong Yu, Vyas Sekar, Srinivasan Seshan, Yuvraj Agarwal, and Chenren 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*, HotNets-XIV, pages 5:1–5:7, New York, NY, USA, 2015. ACM.

[62] Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 665–674. ACM, 2017.