Detection of Threats to IoT Devices using Scalable VPN-forwarded Honeypots *

Amit Tambe
Aalto University, Finland

Yan Lin Aung
Singapore University of Technology and Design, Singapore

Ragav Sridharan
Singapore University of Technology and Design, Singapore

Martín Ochoa
Cyxtera Technologies

Nils Ole Tippenhauer
CISPA Helmholtz Center for Information Security, Germany

Asaf Shabtai
Ben Gurion University, Israel

Yuval Elovici
Singapore University of Technology and Design, Singapore

ABSTRACT

Attacks on Internet of Things (IoT) devices, exploiting inherent vulnerabilities, have intensified over the last few years. Recent large-scale attacks, such as Persirai, Hakai, etc. corroborate concerns about the security of IoT devices. In this work, we propose an approach that allows easy integration of commercial off-the-shelf IoT devices into a general honeypot architecture. Our approach projects a small number of heterogeneous IoT devices (that are physically at one location) as many (geographically distributed) devices on the Internet, using connections to commercial and private VPN services. The goal is for those devices to be discovered and exploited by attacks on the Internet, thereby revealing unknown vulnerabilities. For detection and examination of potentially malicious traffic, we devise two analysis strategies: (1) given an outbound connection from honeypot, backtrack into network traffic to detect the corresponding attack command that caused the malicious connection and use it to download malware, (2) perform live detection of unseen URLs from HTTP requests using adaptive clustering. We show that our implementation and analysis strategies are able to detect recent large-scale attacks targeting IoT devices (IoT Reaper, Hakai, etc.) with overall low cost and maintenance effort.

KEYWORDS

High-Interaction IoT Honeypot; Network Traffic Analysis; Intrusion Detection; Attack Attribution; Adaptive Clustering

1 INTRODUCTION

Internet of Things (IoT) is envisioned as a network of *things* that have physical or virtual representation in the digital world, sensing/actuation capability, a programmability feature and are uniquely identifiable [16]. IoT devices such as smart TVs or smart speakers are becoming increasingly appealing to consumers [24]. Gartner's study estimates almost 20.4 billion IoT devices to be in use by the year 2020 [25]. Due to the nature of emerging markets for IoT devices, manufacturers focus their attention mainly on the core functionalities of products and rush to introduce them in the market [27]. Security aspects of these devices are thus often neglected. Consequently, IoT devices having security vulnerabilities are launched in the market thereby exposing them to targeted exploits in large-scale attacks [5, 20, 27]. Recent large-scale attacks such as Hakai [2] and IoT Reaper [26] demonstrate the gravity of threats faced by IoT devices, exploiting multiple vulnerabilities present due to the heterogeneous nature of IoT devices. Further, exploitation of IoT devices to attack critical infrastructure has become a common attack vector, raising significant concerns for the stakeholders involved [15].

Goal – The goal of this work is to detect new attack waves targeting IoT devices, in particular, the ones leveraging 0-day vulnerabilities for specific devices. Honeypots are commonly used to learn about attacks "in the wild". By utilizing honeypots for IoT devices, we aim to detect large-scale attacks that are able to compromise a large class of IoT devices (such as Mirai that uses easy/default credentials to get shell access to a device), as well as attacks that exploit vulnerabilities in specific IoT devices (such as Persirai that uses faulty access control in specific versions of the embedded web server).

Honeypots for IoT devices can be implemented by emulating a selected set of services (e.g. Telnet [19]). As a result, attacks using services that are not emulated cannot be detected. Moreover, attackers could try to identify honeypots (e.g., by checking error handling, or detecting emulation or virtualization [19]). Additionally, due to their heterogeneous nature, reusing an already emulated IoT device to represent another device cannot guarantee replication of vulnerabilities.

Our contribution – We propose to detect novel attacks on IoT devices by leveraging real devices to build a scalable VPN-forwarded honeypot. In our work, we use real IoT devices to faithfully expose their behavior without requiring occasional enhancement of exposed environment to adapt to changing attack vectors (as in the case of emulated devices). However, using only a few real devices might not provide sufficient network traffic to gain valuable insight. On the other hand, an increasing number of real devices is neither cost-effective nor scalable. Hence, to increase the statistical chance of being

*Pre-print of paper accepted for publication at CODASPY 2019

attacked, we create multiple VPN tunnels, forwarding the traffic of a single physical device to several IPs worldwide. VPN tunnels are used to acquire public IP addresses and expose IoT devices on the Internet as part of our proposed honeypot. At the same time, connecting to multiple VPN servers in different countries allows devices in the honeypot to establish a geographically diverse virtual presence, which is essential for detecting large-scale attacks.

Next, we propose two live traffic analysis strategies for detection and examination of potentially malicious traffic received by honeypot. The first one starts with suspicious events that indicate potential compromises, such as anomalous communication attempts made by an IoT device in our honeypot. By backtracking into the network traffic of the device under potential attack, we can detect the malicious command issued by an attacker that caused those anomalous communication attempts. The second strategy allows live detection of potential 0-day vulnerabilities by analyzing URLs from HTTP requests. Over a period of time, the proposed honeypot framework detected several large-scale attacks targeting IoT devices (Persirai, Hakai, etc), with overall low cost and maintenance effort.

Our contributions can be summarized as follows:

- Design of a honeypot framework that incorporates commercial off-the-shelf (COTS) IoT devices for high-interaction, utilizing low-cost commercial VPN providers
- An implementation of the proposed framework, demonstrating an automated, scalable, and economical approach to integrate COTS IoT devices
- Two live traffic analysis methods to detect large-scale attacks and subsequently 0-day vulnerabilities in IoT devices using our honeypot infrastructure

The rest of this paper is organized as follows. Section 2 provides a brief background on honeypots and recent large-scale attacks targeting IoT devices. We then present the design of our honeypot framework and live traffic analysis strategies in Section 3. The implementation of the honeypot is explained in Section 4, followed by the results of evaluation of network traffic captured by the proposed honeypot in Section 5. We summarize the related work in this field in Section 6 and conclude the paper in Section 7.

2 BACKGROUND

2.1 Honeypots

A Honeypot is a closely monitored computing resource that we want to be probed, attacked, or compromised. More precisely, a honeypot is *"an information system resource whose value lies in unauthorized or illicit use of that resource"* [21]. Honeypots have traditionally been used by security researchers to present decoy systems to attract attackers and learn from their behavior. With the knowledge gained from this, researchers can then apply techniques to prevent such attacks in the future. In general, honeypots can be classified as follows:

High and Low Interaction – High interaction honeypots present real systems to attackers [17]. The main advantage of having real systems is that there is no reason to emulate anything, thus making them more convincing. These type of honeypots allow attackers to gain full access to the system enabling to gain in-depth information about attacks and therefore qualitative results on attacker behavior. A drawback, however, is their cost of implementation and maintenance due to the usage of real systems, making them difficult to scale [21]. Low interaction honeypots, on the other hand, present fully or partially simulated or emulated environments (e.g. partially implemented network stacks). They limit attackers' interaction with the honeypot and hence generate more quantitative data [21]. Their real value lies in their ease to implement, scale and maintain.

2.2 Recent large-scale IoT attacks

It is challenging to build realistic honeypots for IoT devices mainly due to their heterogeneous nature [7]. Since IoT devices are becoming more affordable, it motivates us to design a high-interaction honeypot by incorporating real IoT devices. Such a honeypot would allow detection of new threats to IoT devices while keeping pace with evolving attack vectors.

Persirai – In May 2017, TrendMicro reported a new family of IoT botnet named Persirai [1]. This malware infects cameras that are susceptible to the vulnerability CVE-2017-8225 thereby allowing unauthorized access to the credentials of vulnerable cameras [12]. The perpetrator issues a specifically crafted HTTP request to retrieve *system.ini* file containing credentials, irrespective of the strength of the password. Once the attacker gains access to credentials, the device becomes part of a botnet and scans for more IoT devices with the same vulnerability to spread the malware.

Hakai – This malware campaign was reported recently by Palo Alto Networks [2]. Hakai leverages the source code of Mirai and Gafgyt malware families and continues the recent trend of incorporating multiple exploits affecting several classes of IoT devices. Attackers get access to the device using one of the credentials from a dictionary of predefined factory-state credentials of several devices. The malware further spreads using methods similar to Mirai. However, Hakai has evolved to support several new DDoS methods that were previously unused by Mirai variants.

3 HONEYPOT FRAMEWORK AND TRAFFIC ANALYSIS METHODS

The objective of our honeypot is to attract and detect (possible 0-day) large-scale attacks on IoT devices. We start by outlining the attacker model and then identify salient features that are necessary to design and implement a high-interaction honeypot incorporating real IoT devices.

3.1 Attacker Model

We consider two types of attackers: attackers conducting attacks manually, and attackers using automated scripts to conduct large-scale attacks. In case of manual attacks, we assume the goal of the attacker is to either explore exposed

devices or assess the effectiveness of exploits to be used in subsequent large-scale attacks, by first testing manually on a smaller set of devices. The attacker may interact with the devices in a variety of ways such as inspecting configuration of a printer or viewing a video on the camera. On the other hand, in case of automated attacks, the goal of the attacker is to identify as many vulnerable IoT devices as possible (of the order of thousands [13]) and exploit those to recruit in a botnet for conducting large-scale attacks. The motivation behind this is to rent such botnets on the underground market to conduct DDoS style attacks [8]. In both cases, we assume that the attacker is looking for real devices instead of emulated devices, because real IoT devices provide access to complete systems for exploitation and further proliferation using possible 0-day vulnerabilities.

3.2 Design Considerations

Stealth – The identity of a honeypot, by its nature, should stay concealed. Otherwise, it loses all of its value. When exploring IoT devices in the honeypot, an attacker conducting manual attacks may try fingerprinting those devices. The attacker may try to verify replies to executed commands, try to interact with devices like cameras by zooming, tilting, etc. It is essential, therefore, that the attacker be convinced of the devices being real and present in the actual geolocations indicated by their IP addresses. Some IoT devices (like IP cameras) could unintentionally disclose Service Set Identifiers (SSIDs) of the surrounding Wi-Fi networks through their Wi-Fi settings. Similarly, devices with microphones can capture audio, revealing inconsistencies from the purported geolocation. These need to be prevented as either can reveal the actual location of the honeypot.

Credibility and Robustness – To be credible, a honeypot needs to be realistic. For example, an IP camera will be more credible if it shows realistic visuals instead of simply displaying an empty wall. Attractive visuals closely mimicking realistic scenarios increase the value of a honeypot by making it credible. Further, a honeypot should itself be secure, not allowing attackers to take advantage of and compromise other devices within the honeypot or outside. The value of a honeypot increases if it can detect intrusion attempts and prevent the proliferation of malware.

3.3 Honeypot Features

In this section, we describe the salient features of our proposed high interaction honeypot.

Real devices – To make the honeypot credible, we use real COTS IoT devices. We note that real devices allow full access to the underlying system, thereby maximizing attack surface for attackers. An attacker trying to fingerprint real devices, by way of verifying return values of executed commands or faithful reproduction of actions like camera zoom, can be assured of their authenticity. In addition, we note that using real devices allows for easier integration of new COTS devices. To integrate a new device, we do not need to emulate anything or gain low-level access to the device. Also, we argue that novel (0-day) attacks on embedded devices (e.g. Persirai that exploits a vulnerability in the web server [11]), can only be fully observed on devices that expose this vulnerability.

Apart from using real devices, we also ensure that high interaction devices, such as IP cameras, broadcast pertinent and realistic videos, to enhance the credibility of our honeypot. The goal is to maximize the attacker's interactions with devices in the honeypot, as long as possible.

Interaction Restrictions – To prevent IoT devices from exposing SSIDs of surrounding Wi-Fi networks, wireless functionality of the devices should ideally be disabled. This can be achieved either by software or hardware means. While the software-based approach is low effort and convenient, it could be circumvented by an attacker after gaining access to the device. In order to prevent this, we prefer a hardware-based approach to prevent the attacker from turning on Wi-Fi remotely or scanning neighboring Wi-Fi networks. Similarly, devices such as IP cameras have sensors (e.g. microphones) that could leak unwanted context information of the devices (e.g. conversations in a lab setup). We need to identify a list of such IoT devices, especially with known vulnerabilities and disable their Wi-Fi chips and microphones.

Forwarding via VPN connections – The devices in our honeypot will have to be reachable via a public IP accessible by attackers. VPN providers offer such public IP addresses from servers located in various countries, thereby providing geographic diversity. Such diversity portrays the presence of a device in a different location than its actual location. IP addresses belonging to the pool of a large VPN provider could potentially be identified as such, via online service [9]. However, being heuristic in nature, such identifications are still prone to false positives [9]. In addition, we note that services to identify VPN IPs usually require payment (which makes them unattractive for a careful attacker). On the other hand, even if free usage is allowed, it is usually restricted to only a small number of requests, making them impractical when conducting a large-scale attack consisting of hundreds of IPs. As result, despite the existence of such services, the efforts involved in using those services (in terms of time, cost and utility) would make it unattractive for attackers intending to conduct large-scale attacks.

Salient Attribute Extraction – Once the honeypot is deployed and starts gathering large amounts of network data, it is essential to parse and analyze traffic data efficiently. Crucial attributes of data (such as attacker location, source IP reputation, etc.) need to be extracted for data analyses. Once these attributes are extracted and stored in a database they can be reused later for data visualization.

Maintenance – Maintaining a honeypot and ensuring that it performs its intended function is a critical requirement. As the proposed honeypot scales by addition of new physical devices, maintaining it manually becomes increasingly challenging. In these circumstances, it is essential to automate the following tasks of maintenance: (1) addition and setup of new devices in the honeypot including associated VPN-forwarders and

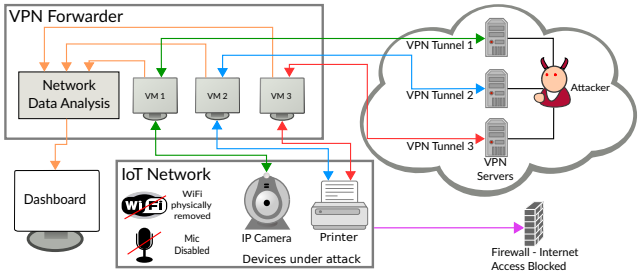


Figure 1: High level design of honeypot

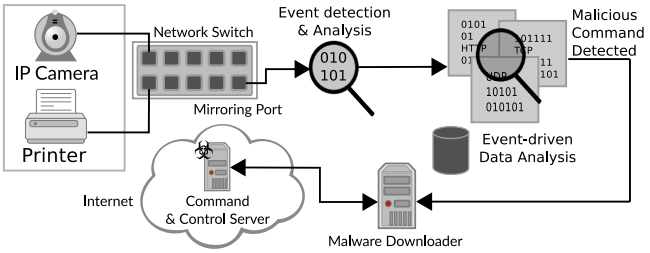


Figure 2: Steps taken during traffic analysis of outbound connections

traffic capturing, (2) removal of a device and (3) periodic monitoring and logging of the health status of honeypot.

Putting it all together – Combining all these aspects, we envisage a high-level design of the proposed honeypot as shown in Figure 1. The design components include:

- Attacker – Attacker is a malicious entity targeting our honeypot. This entity could be a person attacking manually or an automated script.
- VPN Tunnels – These are public IP tunnels provided by a VPN service provider connecting attackers to VPN-forwarder.
- VPN-forwarder – These are Virtual Machines (VMs) that acquire public IPs by establishing VPN tunnels. They forward traffic back and forth between attackers on the Internet and the real IoT devices in the internal network. The VMs expose the same ports as open ports on the real devices in their factory default state to provide a credible environment.
- IoT Network – This is the VLAN that hosts all the real IoT devices that are part of the honeypot. A firewall is set up to prevent any newly initiated connections from leaving the VLAN (for example, to prevent malware from propagating to hosts outside the VLAN). Wireless connectivity of all devices connected to this VLAN is disabled physically to prevent unintended exposure of SSIDs of neighboring wireless networks. For devices where physically disabling wireless connectivity is not possible (like motion sensor, smart plug, etc.), portable low-cost RF-enclosures can be deployed to block wireless signals from outside.
- Firewall – This blocks any outbound connections initiated from the IoT Network, thereby restricting the unintentional proliferation of malware.
- Dashboard – The dashboard component displays real-time visualization of network traffic data after performing analysis on it. It displays important data (e.g. successful logins) and other suspicious behavior (e.g. outbound connection attempts).

3.4 Network Traffic Analysis Methods

Outbound Connection Attribution. The first traffic analysis method attributes attack commands based on detection of suspicious outbound connections as shown in Figure 2.

Outbound Connection Detection – We continuously capture all network traffic flowing to and from all IoT devices in our honeypot for monitoring and analysis. In particular, we use lightweight rules to detect outbound connections from devices. Even though all unsolicited traffic to a honeypot is potentially suspicious [21], spontaneous outbound connections can be considered as potentially malicious (e.g. connections to Command and Control (C&C) servers).

Event-driven Data Analysis – Once a suspicious event such as an anomalous outbound connection attempt is detected, we deploy computationally expensive packet inspection to determine the malicious command (such as simple WGET commands in a shell, HTTP configuration change requests, etc.) that caused the outbound connection. We achieve this by backtracking in the previously captured incoming traffic.

Malware Downloading – After obtaining the malicious command that caused the outbound connection, the downloader component is triggered. It parses the obtained command and establishes a separate outbound connection to the potentially malicious server. After establishing a connection, the downloader then retrieves malware from the potential C&C server. The malware downloaded is then analyzed further.

Live Detection of Novel HTTP Requests. In the second analysis method, we process live network traffic and extract URLs from HTTP GET requests. The URLs are then forwarded to a customized adaptive clustering component [10] based on unsupervised learning, which we call as **HON**eypot Novel HTTP Request **AN**alyzer (HONAN). Once deployed in the honeypot, HONAN detects novel (i.e. unseen) URLs, forms clusters and raises an alarm whenever a new cluster is found.

4 IMPLEMENTATION OF VPN-FORWARDED IOT HONEYPOT

In this section, we describe the implementation of a high interaction IoT honeypot based on our proposed design.

4.1 Honeypot Implementation

Honeypot with Heterogeneous IoT Devices – The first step in implementing the proposed honeypot is selection of IoT devices that will be exposed on the Internet. We decided to include different types of IoT devices in the honeypot to provide a rich attack surface so as to detect recent threats

Table 1: Details of IoT devices used in honeypot

Device	Model	Vulnerabilities (CVE)
	IP Cameras	
D-Link*	DCS-942L	2017-7852
VStarcam*	C7837	2017-5674
Aztech*	WIP C409HD	2017-9765
Trendnet†	TV-IP7621C	–
Trendnet†	TV-IP410	Yes [4]
Sineoji†	PT528V	2017-9765
	Other Devices	
HP Printer*	HP 6830	–
D-Link Router†	DIR-615	2017-7404
Smartthings Hub*	STH-ETH-200	Yes [6]
Smart Plug*	Belkin	Yes [23]
Motion Sensor*	Belkin	Yes [18]

* Factory state default password
† Modified password

and potential vulnerabilities in those devices. A basic query on Shodan search engine for IoT devices returns millions of results [22]. IP cameras form a majority of these results, closely followed by home routers, printers, etc. Starting with the most targeted device in recent attacks – IP cameras, we identified a list of cameras from different manufacturers, especially with known vulnerabilities, to be incorporated in our proposed honeypot. Similarly, we also identified other types of IoT devices, some of which already have known vulnerabilities. Table 1 shows a list of the real devices used in our honeypot. For some devices we retained the default factory state passwords, while for others the password was changed from factory default. The 'Vulnerabilities' column indicates identified vulnerabilities for those devices with the Common Vulnerabilities and Exposures (CVE) identifier (if available).

Device Preparation –Next, to prevent those devices from accidentally disclosing neighboring Wi-Fi SSIDs thereby risking exposure of the honeypot location, we disabled Wi-Fi and microphones on the IP cameras at the hardware level. For example, on D-Link DCS-942L camera, we removed the oscillator of Wi-Fi chip (MT7601U). For other devices, where removing Wi-Fi chips was not possible (such as smart plug or motion sensor), we made use of an electromagnetically shielded enclosure to prevent nearby wireless signals from reaching those devices.

As explained in the previous section, we aimed to have realistic video images to avoid early detection of the honeypot by attackers. For IP cameras, we achieved this objective in two ways: (1) by constructing a setup involving jewelery items and (2) replaying pre-recorded video of an industrial setting. This method of projecting videos does not apply to other device types (such as printer or smart plug).

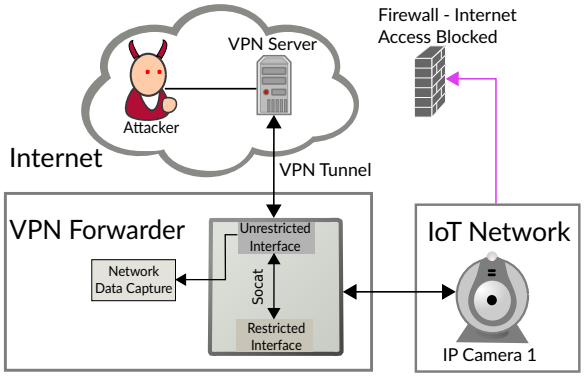


Figure 3: Implementation details for a VPN-forwarder

VPN Provider – We chose a suitable VPN provider, by conducting latency tests for several VPN service providers. Based on the observed latencies and geographical diversity of the provider's servers, we rely on the services of 'VPN Provider1' (real name of selected VPN service provider is withheld intentionally).

VPN-forwarder – This component plays a crucial role in the implementation of the proposed honeypot by exposing the IoT device's interface on the Internet and then forwarding traffic between the attacker and real device. As shown in Figure 3, the VPN-forwarder is implemented as a virtual machine running lightweight Ubuntu operating system. It has two network interfaces:

Unrestricted Interface It has access to the Internet and establishes a VPN tunnel with VPN Provider1. It is used by attackers to gain access to devices in the honeypot.

Restricted Interface This interface connects to the isolated VLAN 'IoT Network' to which all physical devices are connected. The VLAN secures the honeypot by blocking any outbound communication. Hence, any malware that could possibly infect the honeypot is prohibited from proliferating to other hosts outside the honeypot.

The VPN-forwarder uses Socat[1] to transfer traffic from the attacker on unrestricted interface to the real device through the restricted interface, and the responses of the device back to the attacker. The network traffic going through the VPN-forwarder machine is captured using the tool TShark[2] on the unrestricted interface of the VPN-forwarder. This traffic collection happens continuously and gathered pcaps are saved on the file system of the physical host hosting the virtual machines.

Visualization and Data Analytics Dashboard – We use the dashboard to process and visualize raw data captured by honeypot. First, we continuously capture network traffic in honeypot by using TShark. To reduce processing of unwanted raw data, an automated script periodically extracts information such as HTTP credentials, HTTP URLs, geolocation information of IP addresses, etc. from the raw data Tshark's 'ek' option allows to generate JSON format output of that

[1] https://linux.die.net/man/1/socat
[2] https://www.wireshark.org/docs/man-pages/tshark.html

information, for importing into Logstash and subsequent indexing in Elasticsearch. Further, we use several custom scripts that query elasticsearch to select the data that is most pertinent (for example the number of successful logins or outbound connection attempts by IP cameras). The results of these queries are sent to Kibana component of the dashboard for visualization. Simultaneously, we also store URLs from HTTP GET requests filtered by TShark in a separate CSV file. HONAN continuously monitors this file using adaptive clustering to detect novel URLs and displays resulting clusters on a dashboard created using Shiny[3] visualization package.

Health Monitor and Framework Scalability – Building and running a honeypot also entails efforts to ensure that it runs continuously without disruption. It would be infeasible to carry out this effort manually, once the honeypot scales with addition of new devices or new VPN-forwarders. Therefore, we have prepared a script called 'Health Monitor', which runs periodically and performs a series of checks ensuring the correct functionality of the honeypot. These checks test whether: (1) VPN-forwarders are running as expected, (2) VPN connections are active, (3) TShark is capturing network traffic, and (4) end-to-end communication (from VPN-forwarders to IoT devices) is functional. The data produced by this script is logged in a database and eventually gets visualized on the Kibana dashboard to monitor the overall status of the honeypot.

Similarly, we have devised custom scripts to automate the process of addition and removal of devices to and from the honeypot. These scripts automate the task of setting up a new device including its associated VPN-forwarder, network interfaces, VPN tunnels, traffic capturing using TShark and traffic forwarding to the devices using Socat. Automating these tasks greatly reduces manual efforts and is important for scalability.

Scalability – In order to be scalable, the proposed honeypot must be able to incorporate the increasing number of IoT devices, VPN forwarders, and public IP addresses. One single VPN forwarder virtual machine (VM) (running lightweight Ubuntu operating system) requires minimum 256MB RAM and 4GB hard disk space. Currently, we run these on a physical host with 64GB RAM and 3TB of hard disk. If we restrict the memory used by Virtual Box (our visualization software) to 60GB, then a maximum of 240 VMs could be hosted on the physical host. Therefore, one physical host can accommodate up to 240 virtual machines connecting to 240 distinct public IP addresses. If more than 240 VMs are required, an additional physical host could be integrated in our honeypot increasing the capacity of the honeypot to 480 VMs. We have prepared scripts to automate the addition of new virtual machines and collection of network traffic irrespective of the physical host they are running on. Further, all physical IoT devices in the honeypot are connected to a 24-port switch. In our configuration, four ports of the switch are utilized for – (1) incoming Internet connection, (2) mirroring port, (3) management port and (4) connection to the physical host. Thus, one switch can accommodate 20 physical IoT devices which would then be portrayed as multiple virtual devices across the Internet. Adding more than 20 physical devices is possible by incorporating another network switch.

Local Coordinating Server – Apart from the main components and tools mentioned above, we implement a local coordinating server incorporating several other tools. MySQL database is used to store most of the static data such as IP addresses of devices, public IP addresses of VPN-forwarder machines and dynamic data such as outbound connection attempts detected. The tool Socat is used to establish a bidirectional communication channel for transferring data and forwarding traffic from the attackers to the devices and back. TShark is used to capture two types of network traffic data – one between attackers and devices, and another to capture inter-device and device's outbound communication. Rsync[4] is used to transfer json files between different machines.

Running cost – The honeypot implementation makes use of a desktop machine to host all the VPN-forwarders and store captured network traffic, and a second desktop machine to host the analytics dashboard. Both these machines are a one-time investment. The most important hardware investment in the honeypot is the IoT devices. The total one-time investment of all the real devices in the honeypot is $1034. The sole recurring cost for our setup is the fees paid to the VPN service provider. These amount to $5.9 per month per IP. Thus, for our entire honeypot consisting of 44 public IP addresses, the monthly recurring cost is $259.6, making it a cost-effective approach in studying attacker behavior for threat detection. Apart from this, the Health Monitor script periodically checks for any issues in the working of the honeypot and reports those immediately. This reduces the maintenance efforts as well. On an average, two man-hours of maintenance effort are required per week to keep the honeypot running.

4.2 Implementation of Network Traffic Analysis Methods

To implement our proposed methods for live traffic analysis, we took the following steps:

Outbound Connection Attribution

Data Capturing and Monitoring – We capture and monitor all traffic going to and from IoT devices in the honeypot. In particular, we detect traffic originating from any of the devices. To capture such traffic, it is necessary to tap into the network interface of devices in the honeypot. We achieve this by using a TP-Link switch (TL-SG1024DE) to host all real devices and use mirroring port capability of the switch. The mirroring port mirrors traffic that is generated on another port of the switch. This technique lets us capture all traffic at one place (mirroring port) without resorting to individual

[3]https://shiny.rstudio.com/

[4]https://linux.die.net/man/1/rsync

capturing on devices. We use TShark for capturing traffic on the mirroring port and TShark's 'ringbuffer' option to rotate captured PCAP files. Every PCAP file is parsed as soon as it is rotated by TShark.

Event Detection and Reputation Check – Next, we monitor every PCAP file for identifying (potentially) suspicious events like outbound connections. Outbound attempts are detected as attempted connections to IP addresses outside the VLAN of honeypot, specifically to public IP addresses. Outbound TCP connections are identified using TCP flags in the captured traffic (SYN flag set and ACK reset). On the other hand, outbound UDP traffic is identified as traffic originating from devices in honeypot (known source IP) and destined for VLAN outside of honeypot. Once outbound connections are detected, we obtain Autonomous System Number (ASN) information for those IP addresses. We subject those IPs to reputation check using VirusTotal, which provides an interface to many aggregated antivirus products and online scan engines. Next, we examine outbound DNS requests for potentially malicious domain names. All the data from this first round of lightweight parsing are stored in the database for further analysis.

Outbound Connection Attribution – We argue that a malicious outbound connection attempt (potentially to C&C server) is the effect of an attack command issued previously by an attacker. We, therefore, perform packet inspection of historical data, to attribute every outbound connection attempt to its corresponding attack command. We start backtracking by identifying the outbound IP, IP of the device and the protocol over which the device communicates, and then filtering historical PCAP data accordingly. We inspect a maximum of 30 min of historical data. This is because our goal is to detect large-scale attack campaigns that are generally short-lived. We perceive this as typical attacker behavior in such attack campaigns and therefore argue that an outbound connection would immediately follow an exploit attempt, for the attacker to maximize his advantage. We backtrack in incremental steps until we identify the packet and hence the command that caused the outbound attempt. We thus attribute an attack (exploit attempt) to an outbound connection. It may happen that some outbound connections cannot be attributed to their attack commands/exploit attempts. This can happen when an attacker tries to craft malware binary on the device manually. However, as our results in Section 5 show, we are able to attribute the high percentage of outbound connection attempts to attack commands.

Malware Downloading –Performing outbound connection attribution reveals the original attack command that caused the outbound connection. The next step is

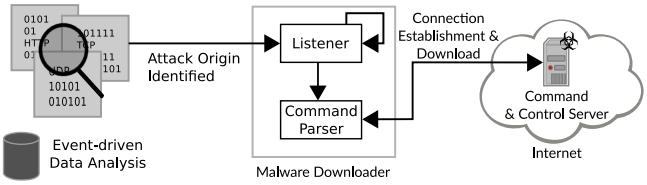


Figure 4: Detailed view of malware downloader

to complete this connection with the malicious server and download attack payload. Allowing this operation on the devices in honeypot poses a risk of unknown malware taking control of devices and possibly the honeypot along with neighboring networks. As such, we devise a separate component, called 'malware downloader', that runs on Ubuntu OS and performs this step of malware downloading. Figure 4 shows the steps taken by the malware downloader when downloading malware. We implemented this component for WGET and TFTP commands. It includes a listener that continuously listens for new messages (a message indicates new attack command was attributed to an outbound connection). Every new message is then parsed to extract the command and then, if required, modified (example, conversion of TFTP options from busybox to Ubuntu). The attack command is then executed to establish a connection with the potential C&C server and download malware.

Novel HTTP Request Analyzer

Implementation of HONAN, which performs unsupervised adaptive clustering of URLs is shown in Figure 5. First, network traffic from mirroring port of our honeypot

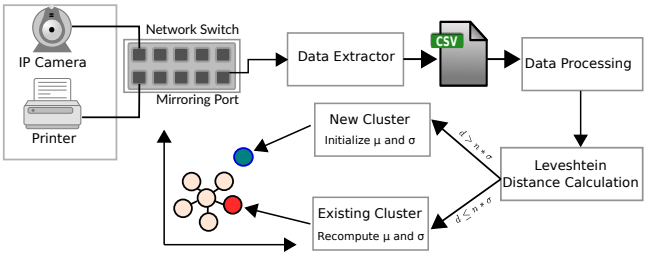


Figure 5: Implementation of adaptive clustering

is processed in real-time. Using TShark tool, timestamps and HTTP GET request URLs are extracted and stored in a CSV file. Typically, URLs consist of three parts: hostname, path and parameters. A series of regular expressions are applied on the URLs to represent them in a generic way while preserving their inherent variations.

After applying regular expressions, HONAN uses adaptive clustering to detect previously unseen URLs that can be potentially malicious. In this technique, we first calculate the distance of incoming URL with the centroids, c_μ, of existing clusters using Levenshtein distance

string metric. If the distance to the closest cluster d is within a threshold of $n \times \sigma$ (σ = standard deviation of the cluster), we add the incoming URL to the closest cluster and recompute the mean, μ, and σ of that cluster. Otherwise, if $d \geq n \times \sigma$, we form a new cluster and initialize c_μ with the incoming URL and σ with a large number (50 in our case). n is a user-defined parameter which is experimentally set to 1 in our work. As more URLs get added to a cluster, μ (mean of cluster) gets updated using equation 1 and σ is updated using equation 2. Finally, to recompute cluster centroids, we calculate the mean salience value of the URLs in the cluster using Term Frequency - Inverse Document Frequency (TF-IDF) algorithm. TF-IDF scores the importance of tokens in a URL based on their occurrences within the URL and the cluster.

$$\mu = \frac{n_c \times \mu + d}{n_c + 1} \tag{1}$$

$$\sigma = \sqrt{\left| \frac{\left(n_c * \left(\sigma^2 + \mu^2\right) + d^2\right)}{n_c + 1} - \mu^2 \right|} \tag{2}$$

where n_c = number of URLs in existing cluster and d = distance to closest cluster.

5 RESULTS

Our honeypot implementation projected 12 physical IoT devices as 44 exposed devices in 15 cities across nine countries. This setup has been running for 15 months and accumulated over 365GB of network traffic data. This section presents observations from the analysis performed on the captured traffic data.

5.1 Live Network Traffic Analysis

When performing network traffic analysis, our goal is to detect large-scale attacks that function by compromising devices in the honeypot. To detect compromise of a physical device we utilize knowledge gained from detecting outbound connections and analyzing HTTP requests. Since IoT devices typically do not make outbound connections, we consider all outbound connections as suspicious and potentially malicious. We therefore continuously parsed traffic data for detecting outbound connections and applied our threat detection methods to detect attacker commands that caused those outbound connections. Similarly, we also considered unseen URLs detected by HONAN as suspicious.

Outbound Connections –We parsed all network traffic to detect TCP and UDP connections that were initiated by devices in the honeypot to an IP address that lies outside its VLAN. We analyzed 365GB of PCAP file data and detected over 150 million connection attempts to

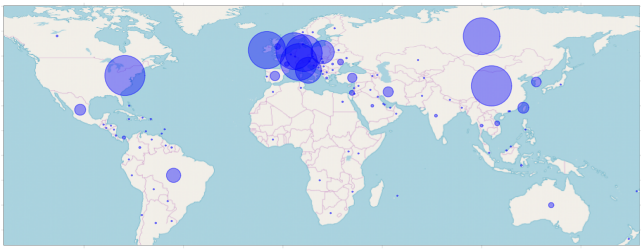


Figure 6: Geographical diversity of outbound connections from honeypot. The size of circle depicts number of connections attempted to IP addresses in that geolocation.

4642 distinct IP addresses. Those 150 million attempts included several retries since we block outbound connections from the honeypot. We analyzed the geolocations of all those 4642 IP addresses to understand the locations of potential C&C servers and observed that connections were made to IP addresses in 109 countries. Figure. 6 shows a map representing the geographical diversity of outbound connections made by devices in the honeypot. Additionally, as part of outbound connection discovery, we also detected DNS requests made by devices and found requests to 4720 unique domains. Those also included device manufacturer domains, presumably for device software updates.

Attack Attribution – After detecting outbound connection attempts, we applied our attack attribution method. Knowing the destination IP and protocol of communication, we backtracked into PCAP traffic to attribute attack commands to corresponding outbound connections. For example, upon detecting a connection attempt to *TARGETIP* via TFTP protocol, we backtracked into PCAP traffic to detect the attack command as

```
/bin/busybox tftp -g -l dvrHelper -r dlr.arm
TARGETIP;
```

We were able to attribute attack commands for 4527 out of 4642 (97.5%) distinct IPs. This high percentage of attack attribution leads to extraction of malicious commands from PCAP traffic and downloading of malware.

Lightweight Backtracking – Backtracking performed for attack attribution was lightweight. 4018 IPs out of 4527 (89%) were attributed by backtracking between 0 to 5 seconds only. We attributed the remaining 509 IP addresses by backtracking up to 120 seconds. We were unable to attribute the remaining 105 (3%) IPs. In those cases, the attacker created malware binaries manually on the devices which caused outbound connections.

Malware Downloading – We ran the malware downloader component for a period of 45 days from 1^{st} July 2018. In this time period, we allowed the malware downloader to execute attack commands detected from outbound

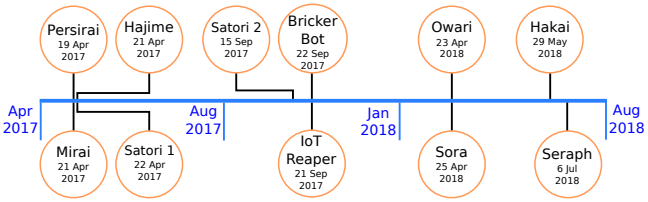
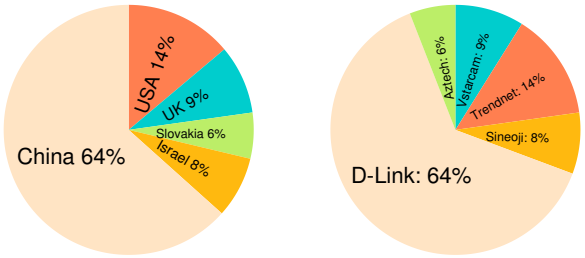


Figure 7: Timeline of malware families detected automatically (as new clusters) by our honeypot.



(a) Origin **(b) Targeted vendor**

Figure 8: Classification of incoming traffic to honeypot

connections and download potential malware. This resulted in 180 distinct files being downloaded. Those files consisted of 78 malicious binaries and 102 other text files. The 78 distinct malware binaries collected belong to multiple processor architectures, such as ARM and MIPS. The remaining text files gave further instructions to download actual malware binaries.

Novel HTTP Request Analyzer – HONAN analyzed 365GB of PCAP files captured from 11th April 2017 to 30th July 2018. We trained HONAN with URLs extracted from network traffic PCAP files captured between 11th Apr and 26th Jun 2017. We removed potential attack URLs before training and noted 2752 clusters. After training was completed, we tested HONAN with previously excluded attack URLs and found new clusters for each of those. After training and testing, we deployed HONAN on our honeypot on 31st July 2017. Since then, HONAN has identified 1259 new clusters till 30th July 2018. These new clusters included clusters representing large-scale attacks on IoT devices such as Persirai, IoT Reaper, Owari and Hakai. A malware family may utilize different HTTP request URLs to compromise multiple vulnerabilities in IoT devices (e.g., as the case with IoT Reaper). As a single attack can consist of multiple exploit URLs, it can be represented by more than one cluster. Further, after deploying HONAN, we noted new clusters that representing novel HTTP requests which did not particularly target IoT devices.

5.2 Malware Activity Detected by Honeypot

The honeypot implementation using our proposed design has been running since April 2017. Since then, we incorporated both our automated live traffic analysis methods into the honeypot. These automated methods have effectively detected several IoT malware attacks while the attack campaigns were active. Figure 7 shows the timeline of the honeypot. It depicts various malware detected by honeypot since its inception.

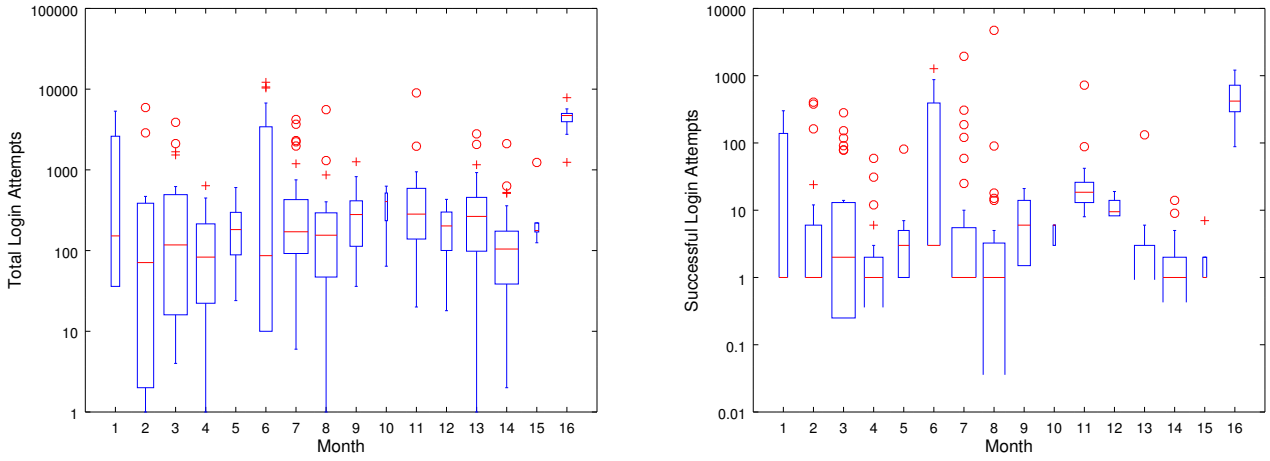
5.3 Honeypot Traffic Characterization

Incoming Data Classification –Continuing our automated live analysis, we analyzed 365GB of traffic received by honeypot to gather statistics on incoming traffic. We designed automated scripts to parse live network traffic to extract various pieces of information. The scripts obtained country of origin of incoming connections (to understand geographic presence of attackers), ports that received maximum traffic (to understand type of attacks), and amount of traffic received by various devices (to understand attacker preferred targets and therefore vulnerabilities in devices).

Figure. 8a shows the distribution of connections received by devices in honeypot, per country. We acknowledge that attackers can tunnel their traffic through IPs other than their own, to misrepresent their true location. We observe that majority connections originated from USA and China. We observed that 54% connections received by honeypot were on Telnet port, while HTTP ports received almost all of remaining 46% connections. We observed that IP cameras received majority of connections in the honeypot, thereby indicating greater attacker interest in those IoT devices as compared to other devices like IP printer, smart switch, etc. We conjecture this is because several recent large-scale attacks on IoT devices target IP cameras. As shown in Figure. 8b, we note that D-Link DCS-942L IP camera received most connections among IP cameras.

Login Attempts – One of the intentions behind using real devices in the honeypot is to provide a more realistic environment to attackers as compared to emulated devices and thus garner quality network traffic. The motivation behind our automated live traffic analysis is to observe attacker behavior. For attackers to exploit devices and execute malicious instructions, it is necessary to first access the devices. A successful login is the most straightforward way to gain such access. It is therefore appropriate to analyze the collected data and search for login attempts made on the devices.

Figure. 9 shows a boxplot of login attempts per month, by attackers, via HTTP and Telnet, since the honeypot became operational in April 2017. Each box represents the range of number of login attempts in a month, while the red line inside every box represents the median for



(a) All login attempts **(b) All successful logins**

Figure 9: Boxplot showing comparison of all login attempts and successful logins

that data. Outliers are depicted as small red circles or crosses, depending on their distance from median.

We parsed the collected network traffic to detect all login attempts made by attackers. We detected attempts made via HTTP and Telnet protocols. In all 610076 HTTP requests were received from 32290 distinct public IP addresses. Of these, 86748 requests were attempts to login using HTTP authentication header, with 9877 attempts being successful. We observed over 23000 distinct credentials being used in login attempts (including several dictionary attacks). In case of single camera that had its telnet port exposed, we detected 147663 login attempts, from 4923 distinct IP addresses. Of these, 13959 attempts were successful. Thus, we detected 23836 combined successful logins over HTTP and Telnet. We further scrutinized the successful logins to distinguish those that caused outbound connections from other "benign" successful logins. We detected 5136 such logins that caused outbound connections.

Figures 9a and 9b show a comparison of total login attempts made by attackers and the successful attempts. Figure 9a shows that the number of login attempts increased steadily, as can be seen from the gradual rise in median values. Figure 9b represents successful logins and depicts many outliers. Even though all successful logins are signals for further inspection, outliers with large values are considered as excellent candidates for further probing of captured network traffic to detect large-scale attacks. An examination of the successful credentials used by attackers reveals that either they were default factory state credentials of the devices or retrieved through exploits of commonly known vulnerabilities. We conjecture the reason being increasing number of automated scripts which assume that most IoT devices are configured with unchanged factory state credentials. Such scripts are easy for the attackers to generate and yield quick results.

Honeypot Detection – Apart from attracting attackers, maintaining stealth is also an important characteristic of a honeypot. We therefore verified whether the devices exposed by the honeypot have been detected as belonging to a honeypot, using a tool created by Shodan, called honeyscore[5]. The tool returns a score between 0.0 (not a honeypot) and 1.0 (is a honeypot) for queried IP addresses. Two out of 44 VPN-forwarders exposed by our honeypot were assigned the highest score of 0.3, none was detected as a honeypot.

5.4 Discussion

The proposed design allowed us to use few real devices and project them as multiple devices, geographically distributed around the world. Our design accommodated various kinds of IoT devices, thereby ascertaining its versatility and scalability. Based on the amount of traffic received, analysis presented and shodan honeyscores, we are certain that the honeypot was effective in attracting attackers and was realistic enough to motivate attackers to perform logins and other intrusion attempts.

However, we would like to acknowledge a few shortcomings with our design and implementation. First, a determined and skilled attacker may be able to detect our honeypot. For example, if the attacker can detect that multiple VPN-forwarders represent the same physical device, it can raise suspicion. However, the probability of an attacker discovering this is low. We observed from

[5]https://honeyscore.shodan.io

our collected network traffic that no attacker (single IP) discovered all VPN-forwarders representing same device.

Similarly, our real cameras currently show engaging but fabricated scenes. Though convincing and realistic at first glance, they may not fool a determined attacker for too long. Placing devices in realistic locations is an alternative. However, that may raise privacy concerns.

6 RELATED WORK

Recently, few implementations of honeypots have emerged with the goal of better understanding vulnerabilities, threats and large-scale attacks targeting IoT devices. Pa Pa et al. were the first to propose a honeypot exclusively for IoT [19]. Their work focuses on capturing Telnet-based attacks on various IoT devices by means of a honeypot called IoTPOT. It consists of a low-interaction frontend responder cooperating with a high-interaction backend virtual environment called IoTBOX. Anirudh et al. presented a honeypot model for mitigating DoS attacks exploiting IoT devices [3]. Their model implements a low-interaction honeypot and works alongside an intrusion detection system (IDS). Their honeypot handles suspicious events detected by the IDS to avert a DoS attack. Luo et al. presented an automatic way to build an IoT honeypot called IoTCandyJar [14]. It utilizes publicly available IoT devices on the Internet to gather responses for its own requests. They applied heuristics and machine learning techniques to customize the scanning procedure for improving response logic. This extends the session longevity leading to higher chances of capturing malware. All these existing works implement honeypots for IoT devices by primarily creating virtual environment or by emulating services. Unlike previous works, Guarnizo et al. propose a high-interaction honeypot platform called SIPHON, incorporating real IoT devices [7]. They utilized cloud service providers (like Amazon AWS) to expose IoT devices on the Internet over geographically distributed IP addresses. However, such IP addresses might be identified as honeypot by the attacker as they belong to autonomous systems of cloud services. Our approach overcomes this limitation by utilizing public IP addresses assigned by VPN servers. For an attacker executing a large-scale attack with thousands of devices in his botnet, significant time and resources are required to uncover the true identity of those VPN IP addresses. Moreover, we make our honeypot stealthier by physically disabling Wi-Fi connectivity and microphones. Disabling Wi-Fi connectivity prevents attackers from scanning neighboring SSIDs while disabling microphones prevents leakage of conversations. More importantly, we propose and implement two live traffic analysis methods to detect novel HTTP requests and attribute attacks corresponding to outbound connections. To the best of our knowledge, we are the first to propose and implement a comprehensive honeypot system that incorporates real IoT devices, performs automated live traffic analysis and captures malware.

7 CONCLUSIONS

In this work, we proposed a design for a generic honeypot framework for IoT devices by utilizing VPN connections. We collected real and high-interaction traffic of attacks by achieving low effort exposure of COTS devices on multiple IP addresses that provided geographical diversity. We implemented the proposed framework with 12 COTS devices, including IP cameras, printers, smart plugs and other smart devices. These devices were exposed on 44 IP addresses in 15 cities across nine countries. We incorporated several precautions to detect compromises and mitigate proliferation of malware. We did not instrument the COTS devices directly to detect compromise, but relied on traffic analysis instead. We proposed and implemented two live traffic analysis methods and deployed those on the traffic being captured by the honeypot.

Based on the analysis of captured traffic, we observed 5136 successful and malicious login attempts resulting in outbound connections. These connections were attempted to 4642 distinct IP addresses across 109 countries. By applying our live traffic analysis methods, we were able to attribute attack commands to 97.5% of outbound IPs. Further, by incorporating live adaptive clustering method we were able to detect several active large-scale attack campaigns such as IoT Reaper, Hakai, etc. We temporarily allowed outbound connections to be established via malware downloader component, for a period of 45 days, resulting in capture of 78 distinct malware binaries. Based on application of two automated live traffic analysis methods, our honeypot was able to effectively detect and attribute attacks to their corresponding attack commands.

REFERENCES

[1] 2017. 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/

[2] 2018. Unit 42 Finds New Mirai and Gafgyt IoT/Linux Botnet Campaigns - Palo Alto Networks Blog. https://researchcenter.paloaltonetworks.com/2018/07/unit42-finds-new-mirai-gafgyt-iotlinux-botnet-campaigns/. (Accessed on 08/17/2018).

[3] M Anirudh, S Arul Thileeban, and Daniel Jeswin Nallathambi. 2017. Use of honeypots for mitigating DoS attacks targeted on IoT networks. In *Proceedings of Conference on Computer, Communication and Signal Processing (ICCCSP)*. IEEE, 1–4.

[4] console cowboys. 2012. Trendnet Camera (Multiple Products) - Remote Security Bypass. https://www.exploit-db.com/exploits/36680/.

[5] Ang Cui and Salvatore J Stolfo. 2010. A quantitative analysis of the insecurity of embedded network devices: results of a wide-area scan. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*. ACM, 97–106.

[6] Earlence Fernandes, Jaeyeon Jung, and Atul Prakash. 2016. Security analysis of emerging smart home applications. In *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 636–654.

[7] Juan David Guarnizo, Amit Tambe, Suman Sankar Bhunia, Martín Ochoa, Nils Ole Tippenhauer, Asaf Shabtai, and Yuval Elovici. 2017. SIPHON: Towards scalable high-interaction physical honeypots. In *Proceedings of the ACM Workshop on Cyber-Physical System Security*. ACM, ACM, 57–68.

[8] Thorsten Holz, Markus Engelberth, and Felix Freiling. 2009. Learning more about the underground economy: A case-study of keyloggers and dropzones. In *European Symposium on Research in Computer Security*. Springer, 1–18.

[9] IP Intelligence. 2018. Free Proxy / VPN / TOR / Bad IP Detection Service via API and Web Interface | IP Intelligence. https://getipintel.net/. (Accessed on 01/03/2017).

[10] Thommen George Karimpanal and Erik Wilhelm. 2017. Identification and off-policy learning of multiple objectives using adaptive clustering. *Neurocomputing* 263 (2017), 39–47.

[11] Pierre Kim. 2017. 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#pre-auth-info-leak-goahead.

[12] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Jeffrey Voas. 2017. DDoS in the IoT: Mirai and Other Botnets. *Computer* 50, 7 (2017), 80–84.

[13] Brian Krebs. 2016. KrebsOnSecurity Hit With Record DDoS. https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/.

[14] Tongbo Luo, Zhaoyan Xu, Xing Jin, Yanhui Jia, and Xin Ouyang. 2017. IoTCandyJar: Towards an Intelligent-Interaction Honeypot for IoT Devices. In *Proceedings of Blackhat*.

[15] Bill Miller and Dale Rowe. 2012. A Survey SCADA of and Critical Infrastructure Incidents. In *Proceedings of the 1st Annual Conference on Research in Information Technology (RIIT '12)*. ACM, New York, NY, USA, 51–56. https://doi.org/10.1145/2380790.2380805

[16] Roberto Minerva, Abyi Biru, and Domenico Rotondi. 2015. Towards a definition of the Internet of Things (IoT). *IEEE Internet Initiative* (May 2015). http://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Revision1_27MAY15.pdf

[17] Iyatiti Mokube and Michele Adams. 2007. Honeypots: concepts, approaches, and challenges. In *Proceedings of the annual southeast regional conference*. ACM, 321–326.

[18] Sukhvir Notra, Muhammad Siddiqi, Hassan Habibi Gharakheili, Vijay Sivaraman, and Roksana Boreli. 2014. An experimental study of security and privacy risks with emerging household appliances. In *Communications and Network Security (CNS), 2014 IEEE Conference on*. IEEE, 79–84.

[19] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. 2016. IoTPOT: A Novel Honeypot for Revealing Current IoT Threats. *Journal of Information Processing* 24, 3 (2016), 522–533.

[20] Mark Patton, Eric Gross, Ryan Chinn, Samantha Forbis, Leon Walker, and Hsinchun Chen. 2014. Uninvited connections: a study of vulnerable devices on the internet of things (IoT). In *Proceedings of Intelligence and Security Informatics Conference (JISIC)*. IEEE, 232–235.

[21] Niels Provos and Thorsten Holz. 2007. *Virtual honeypots: from botnet tracking to intrusion detection*. Addison-Wesley Professional.

[22] Shodan Project. 2017. Shodan. https://www.shodan.io/. (Accessed on 10/18/2017).

[23] Vijay Sivaraman, Dominic Chan, Dylan Earl, and Roksana Boreli. 2016. Smart-phones attacking smart-homes. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*. ACM, 195–200.

[24] Vijay Sivaraman, Hassan Habibi Gharakheili, Arun Vishwanath, Roksana Boreli, and Olivier Mehani. 2015. Network-level security and privacy control for smart-home IoT devices. In *Proceedings of Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 163–167.

[25] Rob van der Meulen. 2016. Gartner Says 8.4 Billion Connected Things Will Be in Use in 2017, Up 31 Percent From 2016. http://www.gartner.com/newsroom/id/3598917.

[26] Yegenshen. 2017. IoT_reaper: A Rappid Spreading New IoT Botnet. http://blog.netlab.360.com/iot_reaper-a-rappid-spreading-new-iot-botnet-en/.

[27] Tianlong Yu, Vyas Sekar, Srinivasan Seshan, Yuvraj Agarwal, and Chenren Xu. 2015. Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things. In *Proceedings of the ACM Workshop on Hot Topics in Networks*. ACM.