IP-Based IoT Device Detection

Hang Guo
USC/CS Dept and Information Sciences Institute
hangguo@isi.edu

John Heidemann
USC/CS Dept and Information Sciences Institute
johnh@isi.edu

ABSTRACT

Recent IoT-based DDoS attacks have exposed how vulnerable the Internet can be to millions of insufficiently secured IoT devices. To understand the risks of these attacks requires learning about these IoT devices—where are they, how many are there, how are they changing? In this paper, we propose a new method to find IoT devices in Internet to begin to assess this threat. Our approach requires observations of flow-level network traffic and knowledge of servers run by the manufacturers of the IoT devices. We have developed our approach with 10 device models by 7 vendors and controlled experiments. We apply our algorithm to observations from 6 days of Internet traffic at a college campus and partial traffic from an IXP to detect IoT devices.

ACM Reference Format:
Hang Guo and John Heidemann. 2018. IP-Based IoT Device Detection. In *IoT S&P'18: ACM SIGCOMM 2018 Workshop on IoT Security and Privacy , August 20, 2018, Budapest, Hungary.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3229565.3229572

1 INTRODUCTION

Internet-of-Things (IoT) devices, such as Internet-connected cameras, smart light-bulbs, and smart TVs, are surging in both sales and installed base. Gartner forecasts the global IoT installed bases will grow from 3.81 billion in 2014 to 20.41 billion in 2020 [3].

Despite rapid growth, there is an increasing concern about the vulnerability of IoT devices and the security threats they raise for the Internet ecosystem. A significant risk is that compromised IoT devices can be used to mount large-scale Distributed Denial-of-Service (DDoS) attacks. In 2016, a botnet consisting of over 100k compromised IoT devices launched a series DDoS attacks that set records in attack bitrates. These attacks included a 620 Gb/s attack against cybersecurity investigation site KrebsOnSecurity.com (2016-09-20) [5], an estimated 1 Tb/s attack on French cloud-computing provider OVH (2016-09-23) [9], and an estimated 1.2 Tb/s attack against DNS provider Dyn (2016-10-21) [2]. Botnet sizes in these attacks were estimated at 145k [9] or 100k [2], from the Mirai botnet. Source code to the botnet was released [7], showing it targeted IoT devices.

To understand these threats requires knowledge of locations, distribution, and growth of IoT devices. The primary contribution of this paper is our new method to detect IoT devices from observations of Internet traffic, combined with knowledge of the servers run by IoT manufacturers that these devices contact (which we call *device servers*). Our method detects both IoT devices with public IP as well as those behind NAT while preserving IoT users' privacy by extracting minimal information (anonymized IPs) from IoT devices' traffic. Our second contribution is to apply our method to flow-level traffic from a college campus and partial traffic from a IXP to detect real-world IoT devices (§3). We believe our algorithm and results could help guide the development of future improvements to IoT security.

2 METHODOLOGY

Our approach to detecting IoT devices follows the insight that most IoT devices exchange traffic regularly with servers run by their manufacturers. If we know these servers, we can identify IoT devices by watching traffic for these packet exchanges. Since servers are usually unique for each class of IoT device, we can also identify the types of devices. We acquire prior knowledge of device servers from devices we or others own.

Our approach therefore depends on identifying servers to look for (§2.1) and looking for these servers in traffic (§2.2). We have considered other signals such as traffic timing or rates, but we use traffic presence because it is robust when NATs mix traffic from multiple devices.

2.1 Identifying Device Server Names

Our approach depends on knowing what servers devices talk to. Our goal is to find domain names for all servers that IoT devices regularly and uniquely talk to. However, we need to remove server names that are shared across multiple types of devices, since they would otherwise produce false detections of IoT devices.

IoT S&P'18, August 20, 2018, Budapest, Hungary

ACM ISBN 978-1-4503-5905-4/18/08…$15.00
https://doi.org/10.1145/3229565.3229572

Identify Server Candidate Names: We bootstrap our list of candidate server names by purchasing samples of IoT devices and recording who they talk to. We describe the list of devices we purchased in §3.1, and provide the information we learned as a public dataset [4].

For each IoT device we purchase, we boot it and record the traffic it sends. We extract the domain name of server candidates from type A DNS requests made by target IoT devices in operation. We capture DNS queries at the ingress side of recursive DNS resolver to mitigate effects of DNS caching.

Filtering Server Candidate Names: We exclude domain names for two kinds of servers that cause potential false positives in detection. One is called *third-party servers*: servers not run by the IoT manufacturer that are often shared across many devices. The other is *human-facing servers*: servers that also serve human.

Third-party servers usually offer public services like time (NTP), DNS, news, music streaming and video streaming. If we include them, they would cause many false positives because they interact with many different clients.

We consider a server domain name S run by some third party if for an IoT product P, neither P's manufacturer nor the sub-brand P belongs to (if there is one) is a substring of the S's domain (regardless of case). We define domain of a URL as the immediate left neighbor of the URL's public suffix. (We identify public suffix based on public suffix list from Mozilla Foundation [8]). We use Python library tldextract for domain extraction instead of reinventing the wheel [6].

Human-facing servers serve both human and device (note that all server candidates serve device because they are DNS queried by IoT devices in the first place). They may cause mis-classifying a laptop or cellphone (operated by human) as IoT devices.

We identify human-facing servers by if they respond to web requests (HTTP or HTTPS access) with human-focused content. This test is supported by the observation that retrieving HTML pages via HTTP or HTTPS is the most common method in which average users access web server; and consuming web content is the most common purpose why average users access web server.

We define *respond* as returning an HTML page with status code 200. We define *with human-focused content* as the existence of any web content instead of place-holder content. Typically place-holder content is quite short (for example, http://appboot.netflix.com shows place holder "Netflix appboot" and is just 487 bytes), so we treat HTML text longer than 630 bytes as human-focused content. We determined this threshold empirically from HTTP and HTTPS content at the 158 server domain names queried by our 10 devices (Table 1); it is a conservative choice to reduce false positives.

We call the remaining server domain names *device-facing manufacturer server*, or just *device servers*, because they are run by IoT manufacturers and serve devices only. We use device servers for detection.

Handling Shared Server Names: Some device server names are shared among multiple IoT device types from the same manufacturer and can cause ambiguity in detection.

If different device types share the exact set of device server names, then we cannot distinguish them and we simply treat them as the same device type.

If different device types share part of their device server names, we can't guarantee they are distinguishable. If we treat them as separate types, we risk false positives and confusing the two types. To avoid this problem when we detect device types sharing common server names, we report we detect at least one of these device types (a conservative choice). (Potentially we could look for unique device servers in each type; we do not currently do that.)

Tracking Server IP Changes: We search for device server by IP addresses in traffic, but we discover device servers by domain names in our test devices. We therefore need to track DNS resolution for server name, and track when it changes over time, and when it varies across networks. We assume that server names are long-lived, but the IP addresses sometimes change, and sometimes depend on location. We track changes of server name to IP mapping by resolving server names to IP addresses every hour. In §3.3, we show that nearly half of mappings are stable, but 58 of our 99 device-server names (collected in §3.1) change IPs at least once in a 2-month period. We also find that 40 of our 99 device-server names give different results based on location. In most cases we therefore collect DNS data at roughly the same time and from the same location as the trace.

2.2 Detecting IoT Devices with Device Server IPs

Our method detects IoT devices by identifying packet exchanges between IoT devices and device servers. For each specific type of device, we track a list of device server names that device talks to. We then define a threshold number of server names; we interpret the presence of traffic to that number of server names in traffic from a given IP address as indicating the presence of that IoT device.

Threshold Selection: Since some device servers may serve both devices and individuals (due to we use necessary condition to determine device server in §2.1 and risk mis-classifying human-facing manufacturer server as device server) and sometimes we might miss traffic to a server name due to observation duration or lost captures, we set a threshold of server names required to indicate the presence of each IoT device type. This threshold is typically a majority, but not all, of the server names we observe a representative device talk to in the lab.

Most devices talk to a handful of device server names (up to 20, from our laboratory measurements §3.1). For these devices, we require seeing at least 2/3 device server names to believe an IoT device exists at a given source IP address. Threshold 2/3 is chosen because for devices with 3 or more server names, requiring seeing anything more than 2/3 server names will be equivalent to requiring seeing all server names for some devices. For example, requiring at least 4/5 server names is equivalent to requiring all server names for devices with 3 to 4 device server names. (We do not consider devices with 1 to 2 device servers names here because for these devices, any thresholds larger than 1/2 are effectively requiring all server names.)

For devices that talk to many device server names (more than 20), we lower our threshold to 1/2. Typically these are very popular devices, and the manufacturer uses a large pool of server names. (For example, our Amazon_FireTV, as shown in Table 1, with 41 device server names.) Individual devices will talk to multiple device server names, but typically only a subset of the pool, at least over short observations.

3 RESULTS: IOT DEVICES IN THE WILD

To explore real-world distribution of IoT devices, we first extract device server names from 26 devices by 15 vendors (§3.1). We then apply detection to Internet flows at a college campus (§3.2) and partial traffic from an IXP (§3.3). (We later verify the accuracy of our approach in §4.)

3.1 Identifying Device Server Names

We use two sets of IoT devices in detection: 10 IoT devices we own (Table 1) and 21 IoT devices from data provided by the University of New South Wales [13]. We extract device server names from both sets of devices with method described in §2.1.

We show the count of server names we find from our 10 devices in Table 2. Of the 171 candidate server names from our 10 devices, about half (56%, 96) are third-party servers, providing time, news, or music streaming. As for the 75 manufacturer servers, only a small portion (7%, 5) of them are human-facing (like prime.amazon.com) while most (93%, 70) of them are device-facing manufacturer servers that will be used in detection. We also find two devices (TPLink_SmartPlug and TPLink_LightBulb) sharing their only server name (devs.tplinkcloud.com) and merge them to one meta-device for detection.

We manually examine the 171 server candidate names and confirm the classifications for most of them are correct (for 157 out of 171, ownership of server domain is verified by whois or websites). We find ownership of 11 server candidate names can not be verified. We list these as third-party servers and do not use them in detection. We find that 3 server candidate names (api.xbcs.net, heartbeat.lswf.net, and nat.xbcs.net) falsely classified as third-party server. We confirm they are run by IoT manufacturer Belkin from query result of "whois lswf.net" and a study from security company SCIP [10]. These three server names fail our test for manufacturer server (§2.1) because their domains show no information of manufacturer.

Similarly, we extracted 48 device server names from 18 of 21 IoT devices from University of New South Wales (using device traces available on their website http://149.171.189.1/). The remaining 3 of their IoT devices are not detectable with our method because they only visit third-party servers and human-facing manufacturer servers.

Combining our 10 IoT devices with the 18 detectable devices from University of New South Wales gives us 26 IoT devices (merging 2 duplicated devices: Amazon_Echo and TPLink_SmartPlug); together they have 99 distinct device server names.

3.2 IoT Devices in a College Campus

To test our detection method, we begin applying them to network traffic from part of our university campus.

Input Traces: We use passive Internet measurements at the University of Southern California (USC) guest WiFi from 2017-10-06 to 2017-10-11 (6 days). To protect user privacy, packet payloads are not kept and IPs are anonymized by scrambling the last byte of each IP address in a prefix preserving manner. Our study was reviewed by USC's IRB and identified as non-human subject research (USC IRB IIR00002456, 2018-04-19).

Input Server IPs: We collect the IPv4 addresses for the 99 device server names serving our 26 IoT devices across the same 6-day period, from a server at USC as described in §2.1.

Detection Results: We see a total of 35 triggered detections (suggesting at least 35 different devices) from 8 user IPs, as shown in Table 3. Note that "Amazon_*" in Table 3 stands for at least one of Amazon_FireTV and Amazon_Echo. Similarly "Withings_*" stands for at least one of Withings_Scale and Withings_SleepSensor (recall how we handle device types sharing server names in §2.1).

Our first observation is IPs with IoT devices often have several devices, suggesting the use of a network-address translation (NAT) device that uses a single IP address from USC's WiFi.

We also find three IoT user IPs (IP A, B, and C) sharing the exact set of IoT devices. A likely explanation is that there is one user with a set of devices behind NAT that is using a dynamically assigned IP address, and that this

Manufacturer (Sub-brand)	**Full Name**	**Product Type**	**Alias**
Amazon	Amazon Bounty Dash Button	Dash Button	Amazon_DashButton
Amazon	Amazon Echo Dot (2nd Generation)	Smart Speaker	Amazon_Echo
Amazon	Amazon Fire TV Stick	Smart TV Stick	Amazon_FireTV
Amcrest	Amcrest IP2M-841 IP Camera	IP Camera	Amcrest_IPCam
D-Link	D-Link DCS-934L IP Camera	IP Camera	D-Link_IPCam
Foscam	Foscam FI8910W IP Camera	IP Camera	Foscam_IPCam
Belkin (Wemo)	Wemo Mini Smart Plug	Smart Plug	Belkin_SmartPlug
TP-Link	TP-Link HS100 Smart Plug	Smart Plug	TPLink_SmartPlug
Philips (Hue)	Philips Hue A19 Starter Kit	Smart Light Bulb	Philips_SmartLightBulb
TP-Link	TP-Link LB110 Smart Light Bulb	Smart Light Bulb	TPLink_SmartLightBulb

Table 1: Vendors and Models of Our 10 IoT Devices

Sever Candidate	171	(100.00%)	
3rd-Party Servers	96	(56%)	
Manufacturer Servers	75	(44%)	(100.00%)
Human-Facing Mfr Servers	5	(3%)	*(7%)*
Device-Facing Mfr Servers	70	(41%)	*(93%)*

Table 2: Manufacturer Servers Extraction Break-down for Our 10 IoT Devices

address changes three times over our six day study. However we cannot verify this hypothesis (by confirming that these three IP addresses do not overlap in time) because we do not have access to raw captured data and we explicitly do not have access to DHCP to individual mappings to protect user privacy.

These observations show our approach works at a real campus. However, our measurements at USC represent only a small fraction of campus network traffic—we see only guest network WiFi, not wired networks and secure WiFi. These results therefore represent a lower bound for actual IoT deployment at USC campus.

3.3 IoT Devices at an IXP

We also apply detection to partial traffic from an IXP.

Input Traces: We use the FRGPContinuousFlowData dataset [14] abbreviated here as *FRGP*, collected by Colorado State University (CSU) from 2015-05-10 to 2015-05-19 (10 days). This dataset consists of anonymized Internet traffic flow records in Argus format from the Front-Range Gigapop (www.frgp.net) connecting customers of that regional network (including 18 universities and 14 large organizations in Colorado) with two commercial ISPs: Century Link and Comcast. Data is provided as Argus-format flow records, with anonymized IP addresses.

Input Server IPs: Since we do not have IPs for our 99 device server names in 2015, we draw upon *historic* DNS data from Farsight Security ([11]). We choose to use Farsight DNS datasets collected from an extended two year period (June 2014 to April 2016, centered by the 10-day interval of FRGP data) to get more device server IPs.

Our server IP data is incomplete, with IPs for 51 of our 99 device server names (giving us at most 11 detectable IoT device). We expect it to be incomplete both because Farsight's data collection is passive, so some server name resolutions may simply not occur in their observations, and because some of these IoT devices may not have been deployed at this time.

We verify server IPs from 2-year Farsight DNS data are likely applicable to the 10-day FRGP data by confirming IPs for our 99 device servers are either stable over time or rotating within stable pools. We collect server IP history for our 99 device server names from 2017-10-20 to 2017-12-28 and confirm that none of them really changes IP in this 2-months period: more than half of them (58 out of 99) rotate IPs within a pool of IPs while the rest (41 out of 99) keep using the same IP mappings. We use Farsight because it collected at roughly the right time period for our network data, and it is collected from many locations.

Detection Results: We see a total of 60 detections from 59 IoT User IPs. However only two types of IoT devices are detected: Withings_SmartScale (58 detections) and PIXSTAR_PhotoFrame (2 detections).

We believe we see so few detections in part due to an incomplete list of server IPs: the DNS data only covers IPs for 51 of our 99 device server names, and even for the 51 covered server names, the DNS data potentially miss server IPs. We miss IPs because for device-server names that rotate within a pool of IPs, passive DNS may miss some. Our IXP also connects academic and government institutions, so it may see fewer household-targeted IoT devices.

We confirm that incomplete server IPs contributes to under-detection by inspecting IoT device candidates in this detection. A IoT device candidate is an IoT device that we think might exist because we identify packet exchanges with at least one of its device server names. As shown in Table 4, for almost all (96%, 555) of the 575 IoT device candidates,

IP A & B & C	IP D	IP E	IP F	IP G & H
Belkin_SmartPlug	Samsung_IPCam	Samsung_IPCam	HP_Printer	Amazon_*
Samsung_IPCam	HP_Printer	HP_Printer	Withings_Scale	
HP_Printer	LiFX_SmartLightBulb	LiFX_SmartLightBulb	Amazon_*	
Netatmo_WeatherStation	Amazon_*	Amazon_*		
LiFX_SmartLightBulb	Withings_*			
Amazon_*				
Withings_*				

Table 3: IoT Devices In USC Campus

IoT Device Candidates	575	(100%)	
Candidates w 1 Identified Dev Sver	555	(96%)	(100%)
Detected IoT Devices	60	(10%)	(11%)
Candidates w 2 Identified Dev Svers	16	(3%)	
Candidates w 3 Identified Dev Svers	4	(1%)	

Table 4: Breakdown for FRGP Data Detection's IoT Device Candidates

we only identify one of their device server names (potentially due to we only know part of 99 device servers' IPs). And the reason we can detect Withings_SmartScale and PIX-STAR_PhotoFrame is these two devices have only one known device server name and seeing that one device server name is enough to trigger detection.

Redo Detection with More Server IPs: We show that a more complete list of server IPs can reveal more types of IoT devices by re-doing detection to the first half day's FRGP data (collected from 7am to 7pm on 2015-05-10) with IPv4 addresses we collect for our 99 device server names from 2017-10-12 to 2018-02-23 at USC (using the method from §2.1.) We believe these server IPs, despite collecting 2 years after FRGP data, are still applicable to some degree, knowing IPs for our 99 device server names are very stable.

The detection results show more types of device detected: besides Withings_SmartScale (7 detections) and PIX-STAR_PhotoFrame (1 detection), we also find Belkin_SmartPlug (2 detections), LiFX_LightBulb (1 detection) and TPLink_Smartplug (1 detection, which could be a TPLink_LightBulb or a TPLink_IPCam because these three devices share the same device server name and are not distinguishable for our method). (We also detect 1 Withings_ *, standing for at least one of Withings_Scale and Withings_SleepSensor.)

We conclude that with our IP-based detection method, it is hard to detect IoT devices in the past because server IPs change over time and commercial historical DNS dataset has limited coverage.

4 VALIDATION

We validate the correctness and completeness of our IP-based IoT detection by controlled experiments. We set up our experiment by placing our 10 IoT devices (Table 1) and 15 non-IoT devices on a wireless LAN behind a home router. We run tcpdump at the wireless LAN interface to observe all traffic from the LAN to the Internet.

We run our experiments for 5 days to simulate 3 possible cases in real-world IoT measurement. On Day 1 to 2, we do not interact with IoT devices at all. Therefore first 2 days' data simulates observations of unused devices and contains only background traffic from the devices, not user-driven traffic. On day 3 to 4, we trigger the device-specific functionality of each of the 10 devices (viewing the camera, purchasing items with Amazon Dash, etc.). Thus if we take the first 4 days' data together it shows what we will see in an extended observation where devices are used. On day 5, we reboot each device, looking how a restart affects device traffic.

Our detection algorithm uses the same set of device server names that we describe in §3.1. We collect IPv4 addresses for these device server names (by issuing DNS queries every 10 minutes) during the same 5-day period at the same location as our controlled experiments.

Detection During Inactive Days: We begin with detection using the first 2 days of data when the devices are inactive. We detect more than half of the devices, 6 true positives out of 10 devices; we miss the remaining 4 devices: Amazon_DashButton, Foscam_IPCam, Amcrest_IPCam, and Amazon_Echo (4 false negative). We see no false positives. (all 15 no-IoT devices has been detected as non-IoT.) This result shows that short measurements will miss some inactive devices, but that background traffic from even unused devices is enough to detect more than half.

Detection to Data With Device Activity: We next consider the first four days of data, including both inactive periods and active use of the devices, with results shown in Table 5. When observations include device interactions, we find all devices.

We also see one false positive: a laptop is falsely classified as Foscam IP camera. We used the laptop to configure the

All Device Present	25	(100%)	
Correctness	24	(96%)	*(100%)*
True Positive	10	(40%)	*(41.67%)*
True Negative	14	(56%)	*(58.33%)*
Incorrectness	1	(4%)	*(100%)*
False Positive	1	(4%)	*(100%)*
False Negative	0	(0%)	*(0%)*

Table 5: Detection Result to First 4 Day's Data

device and change the device's dynamic DNS setting. As part of this configuration, the laptop appears to have contact ddns.myfoscam.org, a device-facing server name. Since the Foscam IP Camera has only one device server name, this overlap is sufficient to detect the laptop as a camera. This example shows that IoT devices that use only a few device server names are liable to false positive.

Applying Detection to All Data: When we apply detection to the complete dataset, including inactivity, active use, and reboots, we see the same results as without reboots. We conclude that user device interactions is sufficient for IoT detection; we do not need to ensure observations last long enough to include reboots.

5 RELATED WORK

Several prior groups considered detection of IoT devices.

Traffic analysis: IoTScanner detects LAN-side devices by passive measurement within the LAN [12]. They intercept wireless signals (WiFi, Bluetooth and Zigbee packets) and identify IoT devices by packets' MAC addresses. They depend on MAC addresses to identify devices, so their work requires LAN access and cannot generalize to detection from Internet-wide traffic. In comparison, our method applies to whatever parts of the Internet are visible in available network traffic, and we are able to categorize devices based on what device servers they visit.

Work from the University of New South Wales characterize traffic of 21 IoT devices by traffic metadata, including sleeping time, average packet size , and number of DNS requests sent [13]. They provide a thorough characterization of their 21 devices' traffic and briefly discuss identifying IoT devices by traffic metadata. Our work has some important differences. We use a different detection signal: packet exchanges with particular device servers, rather than their types of metadata. Our method can detect IoT devices behind NAT from aggregated traffic generated by all NATed devices, enabling us to detect real-world IoT devices.

IPv4 scanners: Shodan is a search engine that provide information (mainly service banners, the textual information describing services on a device) on Internet-connected devices (including IoT devices). Shodan actively crawls all IPv4 addresses on a small set of ports to detect devices using text (like "IP camera") in service banners and other device-specific information. As a result, Shodan covers a subset of IoT devices that use public IP addresses (or use private IP with port mapping to a public address) and have publicly identifiable information.

Mirai is a malware that infect and take over IoT devices for launching cyber attacks [1]. Mirai detects IoT devices by port scanning IPv4 addresses with telnet (ports 23 and 2323), then breaks in using a list of 62 common IoT devices credentials. As with Shodan, Mirai thus covers a subset of IoT devices exposed on the public Internet.

Comparing to Shodan and Mira, our method covers IoT devices using both public and private IP addresses, because we use passive measurements to look for signals that work even on traffic from devices behind NATs. Our detection covers all IoT devices that exchanges packets with device servers during operation.

We also preserve user privacy better as we expose no information about individual IoT users (we only extract IPs but IPs are anonymized). Shodan and Mirai, in comparison, expose user IP and other information like user geolocation or device credential.

6 FUTURE WORK

Although the IP-based IoT detection we describe here is complete, it points to future work that might expand IoT detection. Our current algorithm requires historical DNS data to identify server IP addresses in archived traces. We are working on a new detection method that directly work with server's DNS domain names, without converting them to IP addresses. We are also looking at approaches to learn new server names and IP addresses as part of the detection process.

7 CONCLUSION

To understand the risks of insecure IoT devices requires learning about the location, distribution, and growth of IoT devices. To address this need, we developed an IP-based IoT detection method, seeded with information of 10 IoT devices from 7 vendors. We test our new approach with controlled experiments, then we apply it to 6-days of Internet traffic from a college campus, and also to 5 days of traffic from an IXP.

Acknowledgments: We thank Arunan Sivanathan at University of New South Wales for sharing their IoT device data with us [13]. We thank Paul Vixie for providing historical DNS data from Farsight. This material is based on research sponsored by Air Force Research Laboratory under agreements FA8750-17-2-0096 and FA8750-17-2-0280, and by the Department of Homeland Security (DHS) Science and Technology Directorate, Cyber Security Division (DHS S&T/CSD) via contract number HHSP233201600010C. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

REFERENCES

[1] Anna-Senpai. Mirai Malware Source Code. https://github.com/jgamblin/Mirai-Source-Code.

[2] Dyn. Dyn analysis summary of Friday October 21 attack. http://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/.

[3] Gartner. The Internet of Things units installed base from 2014 to 2020. https://www.statista.com/statistics/370350/internet-of-things-installed-base-by-category/.

[4] Guo, H., and Heidemann, J. IoT traces from 10 device we purchased. https://ant.isi.edu/datasets/iot/.

[5] Krebs, B. KrebsOnSecurity hit with record DDoS. https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/.

[6] Kurkowski, J. Python domain extraction library tldextract. https://pypi.python.org/pypi/tldextract.

[7] Loshin, P. Details emerging on Dyn DNS DDoS attack, Mirai IoT botnet. blog http://searchsecurity.techtarget.com/news/450401962/Details-emerging-on-Dyn-DNS-DDoS-attack-Mirai-IoT-botnet, Oct. 2016.

[8] Mozilla. Public suffix list from Mozilla foundation. https://www.publicsuffix.org/.

[9] OVH. OVH news - the DDoS that didn't break the camel's VAC. https://www.ovh.com/us/news/articles/a2367.the-ddos-that-didnt-break-the-camels-vac.

[10] SCIP. Belkin Wemo switch communications analysis. https://www.scip.ch/en/?labs.20160218.

[11] Security, F. Passive DNS historical Internet database: Farsight DNSDB. https://www.farsightsecurity.com/solutions/dnsdb/.

[12] Siby, S., Maiti, R. R., and Tippenhauer, N. O. IoTscanner: Detecting privacy threats in IoT neighborhoods. In *Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security* (New York, NY, USA, 2017), IoTPTS '17, ACM, pp. 23–30.

[13] Sivanathan, A., Sherratt, D., Gharakheili, H. H., Radford, A., Wijenayake, C., Vishwanath, A., and Sivaraman, V. Characterizing and classifying IoT traffic in smart cities and campuses. In *Proceedings of the IEEE Infocom Workshop on Smart Cities and Urban Computing* (May 2017), pp. 559–564.

[14] USC/LANDER. FRGP (www.frgp.net) Continuous Flow Dataset, traces taken 2015-05-10 to 2015-05-19. provided by the USC/LANDER project (http://www.isi.edu/ant/lander).