Can We Classify an IoT Device using TCP Port Scan?

Arunan Sivanathan, Hassan Habibi Gharakheili, and Vijay Sivaraman
School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, Australia.
Emails: {a.sivanathan,h.habibi,vijay}@unsw.edu.au

***Abstract*—The explosion in the number of Internet-of-Things connecting to smart environments has increased the demand for obtaining visibility into these devices among network operators, black-box penetration testers, as well as cyber-attackers. More specifically, enterprise network operators need efficient tools to classify devices in operation for a better maintenance of their network assets, enforce device-specific policies, or quarantine vulnerable devices, thereby reducing the likelihood that they will be compromised. Recent works have advocated passive network monitoring techniques to classify these devices based on their characteristics. However, these techniques require special network infrastructures and powerful data analytics engines to monitor and classify connected devices based on their network behavioral profile. Also, active discovery methods are often limited to the number of devices due to various factors. This paper aims to discover whether an IoT can be classified using an active TCP port scan. First, we propose a technique to determine the type of an IoT device by probing its open TCP ports based on prior knowledge from a range of IoT devices. Then, we evaluate our method by applying to 19 distinct off-the-shelf consumer IoT devices from different vendors. Our preliminary results show that IoT devices can be identified and classified by a very lightweight network TCP scan.**

***Index Terms*—IoT, device classification, port scan**

I. Introduction and Background

THE rapid increase in the Internet of Things (IoT) comprising smart lights, cameras, motion sensors, door locks, thermostats, power switches, medical devices, industrial and household appliances connected to the Internet, has already started a profound effect on our lives. Recent predictions show that the number of connected devices around the world will surpass 20 billion by 2020 [1]. It shows that future enterprises and campus networks will be instrumented with thousands of connected smart devices to facilitate remote monitoring, security, entertainment and efficient management for industries and smart cities. Along with the benefits, the unprecedented scale of IoT has introduced new challenges to network administrators.

Getting the visibility into IoT devices on the network is becoming one of the critical challenges for network administrators. It is crucial to maintain assets, enforce device-specific policies, and refrain access permits to the blacklisted devices from the network. For example, knowing what IoT devices are operating on the network is vital for trouble-shooting. It will help operators block or quarantine known vulnerable device types from the network [2] before they do harm to the network.

The recent focus of researchers and industries on IoT classification has brought various methods to fingerprint and identify devices by observing their network traffic passively. In [3], [4] authors extract the fingerprint of wireless-connected devices from the radio signaling patterns. In [5], authors note that the existence of IoT devices in a network can be identified by observing unique servers communicated through the gateway. Meantime [6]–[9] discuss various attributes that can be extracted from IoT device traffic pattern for identifying and/or classifying IoT devices. Followed by these, in [10] we characterized the IoT traffic based on the statistical attributes such as activity cycles, port numbers, signaling patterns and cipher suites. Also, we proposed a multi-stage machine learning model to classify the devices using flow-level attributes extracted from network switches equipped with special hardware-acceleration (e.g. NetFlow).

While all the above works make important contributions to device classification based on passive monitoring techniques, they require special network middle-box infrastructures, special hardware accelerators, or inspection engines that monitor the network traffic of IoT devices. Also, these solutions require a considerable amount of data to "learn" unique fingerprints for each device. These are highly sophisticated solutions, perfect for continuous monitoring of devices, and possibly discover a new device quickly once they come online.

However, in the scenarios like trouble-shooting or occasional device identification, network administrators may require a simple and lightweight IoT tool that can identify and classify IoT devices with a minimal amount of prior knowledge on various IoT devices. Also, it needs to be highly portable and can be operated without making any changes to network infrastructure. In most cases, administrators of a large-scale network, may not have full visibility into device mapping and infrastructure. Thus, they may want to "actively" probe connected devices using endpoint IP addresses to discover the type of individual devices.

To the best of our knowledge, active device discovery is still in its infancy. In this paper, we examine few possible approaches to actively classify IoT devices.

A. OUI Prefix in MAC address

The vendor information obtained from the Organizationally Unique Identifier (OUI) prefix of the MAC address can be used to classify a device. However, it may not be accurate enough since: (a) IoT device manufacturers typically use NICs

TABLE I
OPEN TCP PORTS FOR IOT DEVICES.

Devices	TCP ports
Amazon Echo	4070, 4071, 55442, 55443
August Doorbell	554, 8554, 19531
Belkin Cam	80, 81, 443, 9964, 49153
Belkin Motion	53, 49152
Belkin Switch	53, 49155
Dlink Cam	21, 23, 5001, 5004, 16119
Google Chromecast	8008, 8009, 9000
Google Home	8008, 8009, 9000, 10001
HP Printer	80, 443, 631, 3910, 3911, 8080, 9100, 9220, 53048
Hue bulb	80, 8080
iHome Plug	80
Netatmo Cam	80, 5555
Samsung Cam	80, 443, 554, 943, 4520, 49152
Smart Things	23, 39500
TPLink Cam	80, 554, 8080
TPLink Switch	80, 9999
Triby Speaker	80, 5080, 44395
Vivitar power	6668
Whithings Sleep	22, 7685, 7888

supplied by third-party vendors, and hence the OUI prefix of the MAC address may not convey any information about the IoT device; (b) it is impossible to distinguish the different device types from same vendor, even though the OUI provides correct details of the vendor.

B. DHCP hostname

Few devices use `hostname` field during the DHCP negotiation to advertise their names to the DHCP server (e.g. Samsung SmartThings device announces itself as *"SmartThings"* by the hostname field). This can be used to identify some devices in the network. However, there are a number of issues with the DHCP hostname: (a) many IoT devices do not set the hostname option in their DHCP requests – indeed we found that about half the IoT devices we studied do not reveal their hostnames; (b) even if the IoT device exposes its hostname it may not always be meaningful (e.g. iHome PowerPlug uses *"hap-29D71D"* for its hostname) and lastly; (c) some devices (e.g. HP printer) allow users to customize the hostnames to an arbitrary value.

C. Service Discovery Protocols

Several service discovery protocols (e.g. SSDP, Bonjour, Alljoyn, and IoTivity) are used to advertise the services offered by the devices to their peers [11]. Most of these protocols are designed to respond to discovery/search packets generated in the multicast address space. It is a perfect solution to scout devices on the network along with the services offered by them. However, only a limited number of devices have adopted the discovery protocols to identify themselves so far. Another downside of using the service discovery protocols is that in typical network configurations, multicast discovery packets may not reach devices in other subnets. Thus, it may allow discovery of devices only within the subnet where the scanner resides unless the multicast forwarding is enabled.

D. Parsing Service Banners

Some devices respond with a banner to request for a service exposed by them. Sometimes, these banners may contain

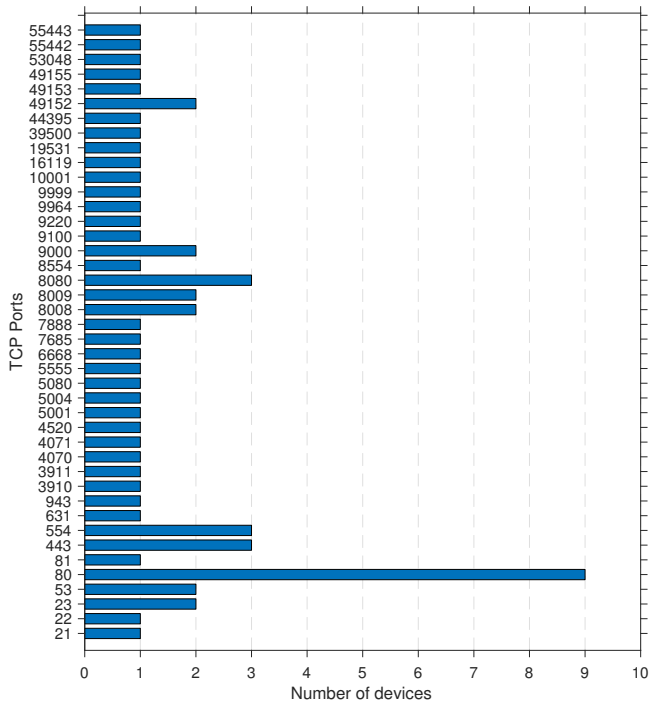

Fig. 1. Histogram of TCP ports across IoT devices.

details about the device, helping detect and classify them. For example HTTP service may reveal details such as server version, type of the host operating system, or hostname. Traditionally this technique has be used by network administrators to map their inventory of the systems on their network. Also, popular IoT device search engines such as `Shodan` [12], [13] use bots to parse these banners and index devices like webcams, SCADA systems, and network routers. Meantime, as these information can be used by cyber intruders to narrow down some applicable exploits, some manufacturers started restricting the details shown in the service banners. In addition to that, it will become hard to obtain a banner when a device does not expose services on standard ports (e.g. running a web server on a custom port number instead of using the standard port 80).

The methods mentioned above have their own pros and cons. Each method is able to identify different sets of devices. Still, there are some devices that cannot be seen under any of these discovery techniques. Therefore, we have come up with a technique to classify IoT devices by focusing on open TCP ports on the device side. This paper aims to find the feasibility of this approach.

The contributions of this papers are:

- First, we propose a technique to determine the type of an IoT by probing its open TCP ports based on the prior knowledge we gathered from a range of IoT devices. We use a hierarchical port scanning approach to perform an optimal scanning without creating unnecessary congestion on the network.
- Then, we evaluate our method by applying it to 19 distinct off-the-shelf consumer IoT devices from different vendors. The set of devices we used includes cameras,

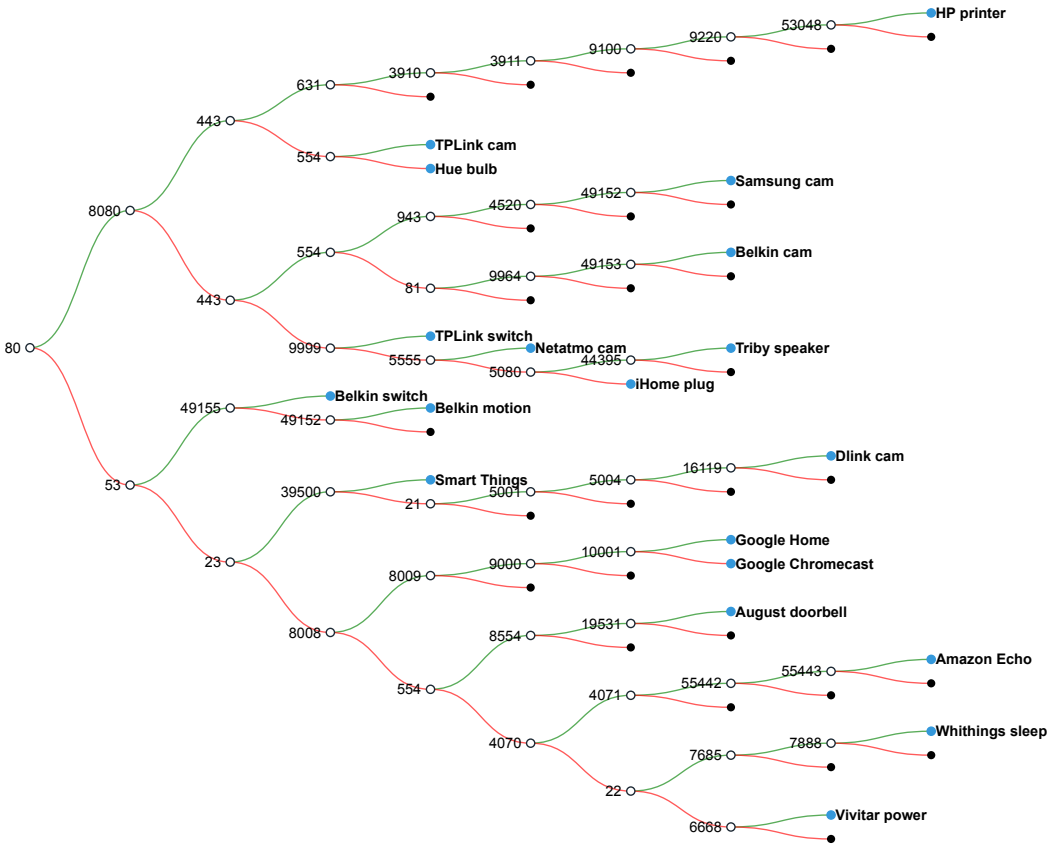

Fig. 2. Hierarchal TCP scan for classifying IoT devices.

light bulbs, power plugs, voice assistants, and home appliances.

II. Device Discovery using Port Scans

Since most of the IoT devices are manufactured for specific (and limited) tasks, they expose different sets of services operating on port numbers specific to the device chosen by their manufacturer [14]. Unlike traditional non-IoT devices, it has less likelihood to be customized by users in typical scenarios. We found unique combinations of open ports operational on various devices in our previous work [15], which evaluates the security of IoT devices. This inspired us to generate the inventory of TCP port signatures to distinguish a range of devices.

A. Device Signature of Open Ports

We now explain two methods for fingerprinting of IoT devices considering open TCP ports. TCP open ports can be probed using two different modes: 1) SYN stealth (also known as half-open) scan, and 2) TCP connect scan.

1) SYN Stealth Scan: It is the quickest scan method since it probes the ports just by the SYN packet instead of attempting to establish a full TCP connection. A SYN Stealth scanner labels a port as ***open*** when she receives an **SYN/ACK** packet as a response for an SYN request targeted to a port, and as ***closed*** when she receives a **RST** (reset) from the device. Also, she notes a port as ***filtered*** if no response is received from the remote device after several attempts. The filtered ports are mostly occurring because of host-based firewalls that block reply packets.

2) TCP Connect Scan: In this method the scanner attempts to establish a successful TCP connection on each port. TCP connect scanner decides a port as ***open*** when the connection is established successfully after complete three-way handshakes. Otherwise, she marks it as ***closed***.

TCP connect mode is more reliable than the SYN stealth scan since all the ports reported by TCP connect scanner should be mapped to a service which is ready to make a connection. Thus, we use TCP connect mode to scan our devices. For this work, we use `Nmap` to scan the ports of each device in our testbed.

Table I shows the list of open ports on IoT devices (i.e. signature) identified by scan over entire port range (i.e. 1 to 65535). Altogether, we found 42 unique open TCP ports form a collection of 19 devices. Although we see the different combination of open ports for each device, we find some similarities specially for devices from the same vendor (e.g. both Google Chromecast and Google Home use ports 8008, 8009, and 9000). Fig. 1 shows the histogram of TCP ports across IoT devices. It can be seen that TCP port 80 (i.e. HTTP) is the dominant port number that is seen in 9 (out of 19) devices, followed by TCP ports 443, 554 and 8080. Also, we see several unique ports used by only one device.

B. Device Identification using Hierarchical Port Scanning

Now, we propose an efficient mechanism to identify the unknown device types based on the open TCP port fingerprints we extracted earlier. Rather than scanning entire port range in devices and mapping with the TCP open port fingerprints, we propose to scan the ports one after the other in a hierarchical manner.

We used an automated script to build a hierarchical tree from the fingerprints of the devices as shown in Fig. 2. This tree depicts the consecutive probing order based on the status of previous port whether it is open (indicated by green links) or closed (indicated by red links). Initially, the script chooses to probe the most used port number first, which is port 80 in our case. If the port is open, it eliminates the devices that keep port 80 as closed. Then it attempts the second most frequently used port number (in our case it is port 8080). In a similar way, we eliminate the devices until we confirm the fingerprint of a device(shown by blue nodes). Here, the black nodes indicate that there are no known fingerprints for further probing.

III. Evaluation

In this section, we evaluate the efficiency and benefits of our proposed hierarchical scan.

A. Required Number of Probes

In order to identify a given device by probing the entire port range (1-65535) would be a highly time-consuming task. Also, intensive port scan offer a heavy load to the network by probing packets, thus impacting the network performance. Instead of scanning the entire port range, probing the collection of ports, used by at least one device, can reduce the number of ports to be probed per device. In our case, still, it requires 42 port probes per device. The number of ports needed to be checked per device may increase dramatically with the number of devices in our fingerprint inventory.

Fig. 2 shows that hierarchical tree based scanning approach needs only 6 ports on average to classify and confirm 19 IoT devices. Also, we note that many devices can be classified within a few numbers of probes without checking all ports identified by fingerprinting. However, we recommend confirming the full fingerprint rather than pruning trees to decrease the false positive alarms. For example, probing first 3 port (80, 8080, 443) is enough to classify the HP printer from other devices. However, probing the rest 6 ports will confirm that, it is a real HP printer, rather than falsely labeling a device that is not included in the fingerprint inventory. This shows that our methodology can classify the IoT devices with the minimal amount of probings. Also, this tree can be further optimized by balancing the branches or manually prioritizing the devices that are in high quantity.

B. Including UDP ports

Scanning the UDP ports are generally slower (and more complex) compared to TCP scans. The challenge with UDP port scanning is UDP protocol does not exchange any handshaking packets to initiate the connection. Due to this reason, most of the time opened UDP ports does not respond to probing packet which crafted with an empty or arbitrary payload. It makes difficult to identify the open ports accurately. Meantime, closed UDP ports respond with ICMP port unreachable error message. However, many devices limit ICMP port unreachable error messages in the rate of one per second in by default. It makes the scanning process extremely slow. Thus on this work, we limited port fingerprinting to TCP ports only

IV. Conclusion

Getting the visibility into IoTs among the thousands of smart devices connected to an enterprise or campus-scale network is becoming a tedious task for network operators. Existing frameworks for IoT device detections mostly depend on passive traffic monitoring techniques and/or require additional special infrastructures. Also, active device discovery methods are applicable for a very limited set of devices. In this work, we have analyzed the feasibility of an active device classification technique using a hierarchical port scanning method. The preliminary results indicate that this technique can be used to classify the devices with the minimal amount of probings.

References

[1] IEEE Spectrum. (Last accessed July 2017.) Popular Internet of Things fore of 50 billion devices by 2020 Is outdated. https://goo.gl/6wSUkk.

[2] Cisco, "Cisco 2017 Midyear Cybersecurity Report," Tech. Rep., 2017.

[3] V. Srinivasan *et al.*, "Protecting your daily in-home activity information from a wireless snooping attack," *Proc. UbiComp*, p. 202, 2008.

[4] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A. R. Sadeghi, and A. S. Uluagac, "Peek-a-Boo: I see your smart home activities, even encrypted!" Tech. Rep., 2018. [Online]. Available: http://arxiv.org/abs/1808.02741

[5] H. Guo and J. Heidemann, "IP-Based IoT Device Detection," in *Proc. ACM workshop on IoT Security and Privacy (IoT S&P)*, Budapest, Hungary, Aug 2018.

[6] M. Miettinen, S. Marchal, I. Hafeez, T. Frassetto, N. Asokan, A. R. Sadeghi, and S. Tarkoma, "IoT Sentinel Demo: Automated Device-Type Identification for Security Enforcement in IoT," Tech. Rep., 2017.

[7] Y. Meidan *et al.*, "Profiliot: A machine learning approach for iot device identification based on network traffic analysis," in *Proc. Symposium on Applied Computing*, Marrakech, Morocco, 2017, pp. 506–509.

[8] Y. Meidan, M. Bohadana, A. Shabtai, M. Ochoa, N. Tippenhauer, J. D. Guarnizo, and Y. Elovici, "Detection of Unauthorized IoT Devices Using Machine Learning Techniques," *arXiv*, 2017. [Online]. Available: http://arxiv.org/abs/1709.04647

[9] Y. Amar *et al.*, "An Analysis of Home IoT Network Traffic and Behaviour," 2018. [Online]. Available: http://arxiv.org/abs/1803.05368

[10] A. Sivanathan *et al.*, "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," *IEEE Transactions on Mobile Computing*, 2018.

[11] Z. Song, A. A. Cárdenas, and R. Masuoka, "Semantic Middleware for the Internet of Things," *2010 Internet of Things, IoT 2010*, 2010.

[12] "Shodan." [Online]. Available: https://www.shodan.io/

[13] J. Guarnizo, A. Tambe, S. S. Bhunia, M. Ochoa, N. Tippenhauer, A. Shabtai, and Y. Elovici, "SIPHON: Towards Scalable High-Interaction Physical Honeypots," 2017.

[14] A. Hamza *et al.*, "Clear as MUD: Generating, Validating and Applying IoT Behaviorial Profiles," in *Proc. ACM workshop on IoT Security and Privacy (IoT S&P)*, Budapest, Hungary, Aug 2018.

[15] F. Loi *et al.*, "Systematically Evaluating Security and Privacy for Consumer IoT Devices," in *Proc. ACM workshop on IoT Security and Privacy (IoT S&P)*, Texas, USA, Nov 2017.