IoT network traffic analysis: opportunities and challenges for forensic investigators?

Tina Wu[a,*], Frank Breitinger[b], Stephen Niemann[b]

[a]*Department of Computer Science*
University of Oxford, Parks Road, Oxford, UK
[b]*Hilti Chair for Data and Application Security*
Institute of Information Systems
University of Liechtenstein, Fürst-Franz-Josef-Strasse, 9490 Vaduz, Liechtenstein

Abstract

As IoT devices become more incorporated into our daily lives, their always on approach makes them an ideal source of evidence. While these devices should use encryption to protect sensitive information, in reality this is not always the case e.g. some expose sensitive data like credentials in cleartext. In this paper, we have conducted an extensive analysis on the communications channels of 32 IoT consumer devices. Our experiments consisted of four main parts; first we carried out a port scan to determine if any ports can be exploited and thus gain remote access. Second, we looked at whether any of the devices used encryption and if not what type of content was exposed. Third, we used the network traffic 'metadata' to identify the destination the data terminated. Finally, we examined the communication between the mobile app and the cloud to see if it can be easily exploited using a proxy server. Our findings show that the majority of devices have remote access unavailable. We found the Shannon entropy test a useful pre-test in identifying unencrypted content. Although many devices encrypted their data, we found several in particular smart cameras would send data in cleartext when they detected motion or during updates. We found the majority of data transverses to the US and stored on Amazon servers with most devices contacting multiple destination. Lastly, we discovered many of the IoT device's mobile apps can be easily exploited using a HTTP Proxy.

Keywords: Internet of Thing, IoT Devices, Network Forensics, Traffic Analysis, IoT investigation, Network traffic

1 Introduction

IoT devices and the potential evidence on them become more and more important when working on criminal cases. For instance, a recent criminal investigation involving data from a suspect's Fitbit device proved useful in a murder investigation. Police were able to use the heart rate data that showed a spike at the time of the alleged crime (Buhecker, 2018). On the other hand, a recent survey by Wu et al. (2019) found that over 50% of practitioners did not feel prepared to handle IoT devices and that there is a shortage of tools for IoT forensics. When IoT devices are involved in a forensic investigation, it is important to make a decision on how to collect evidence, e.g., memory, internal storage or network layer where this work focuses on the latter approach. Typically, this involves examining the network traffic between the devices and systems it communicates with (e.g., cloud, mobile app) looking for unencrypted information (Servida & Casey, 2019; Kayode & Tosun, 2019). Additionally, if the data is encrypted, the 'metadata' from the network traffic can also be helpful, e.g., the country in which the data resides. Often data is stored in multiple countries creating challenges for investigators when various regulations are involved.

*Corresponding author.
Email addresses: tina.wu@cs.ox.ac.uk (Tina Wu), Frank.Breitinger@uni.li (Frank Breitinger), Step.Niemann@posteo.de (Stephen Niemann)
URL: http://www.FBreitinger.de (Frank Breitinger)

While there has been a significant amount of research on IoT devices from various angles, most existing research does not focus on the forensics implications. Therefore, this article will focus on the following four research questions:

RQ1 Does a device expose ports that allow an investigator to connect / access a device?

RQ2 Do IoT devices utilize encryption when sending/receiving information from the cloud and corresponding App?

RQ3 To which countries do the IoT devices and Apps communicate / establish connections (which is an indicator where data resides)?

RQ4 Do IoT device applications (Apps) utilize encryption when sending / receiving information?

To answer these questions, we examined the network traffic of 32 consumer IoT devices (we bought 17 devices; the remaining 15 were part of an existing dataset). Our contribution shows that there are still plenty of problems with IoT devices, e.g., several do not use encryption and therefore traffic can be intercepted. Additionally, we show that data from IoT devices may be spread all over the world which makes it hard to seizure evidence due to jurisdictional challenges. Lastly, we provide a brief prototype implementation that can be used by investigators to analyse network captures.

The rest of the paper is organized as follows: The next section discusses the related work, where we focus on research on

consumer IoT devices, on port scanning, utilising encryption, and using the metadata from the network traffic Subsequently, we outline the methodology and process used to conduct our experiments in Section 3. In Section 3.3 we present the methods used to collect the network traffic followed by the results from the various experiments in Section 4. Section 5 we summarize and evaluate the tool we developed. Finally, we conclude the paper in Section 6.

2 Related work

In this section we review literature related to port scanning, utilising encryption, HTTP proxy and network traffic metadata.

Port scanning. An investigator can scan for open ports on an IoT device to exploit and gain access to the device. Kumar et al. (2019) used a dataset from an internet wide scan that was collected by the consumer security software company Avast. They analysed the dataset for the most common ports open on IoT devices and found device discovery (Universal Plug and Play (UPnP) and mDNS) and device administration (HTTP and HTTPS) ports being the most common. They found that UPnP was prevalent port used by nearly half of the devices. Similarly, Alrawi et al. (2019) manually scanned devices on their network and found 84 open ports running various customised services, SSH, UPnP, HTTP, DNS, Telnet and RTSO. A number of these open ports were found to be vulnerable, these included ports HTTPS/443, SSH/22. More specifically devices running UPnP required no authentication or inbuilt security allowing anyone on the same network control of the device.

Utilising encryption. An essential step when analysing network traffic is to identify if the communication is encrypted or unencrypted. A straightforward approach is considering the port of the communication, however, often ports are abused or randomly chosen. Thus, this can only be an indicator. Consequently, researcher also analyse the payload looking for less random sequences. For instance, it became common to use Shannon (1949) entropy or the chi-square test (Wood et al., 2017).

Similar to our work, Loi et al. (2017) examined the network traffic of 20 consumer smart home devices using the Shannon entropy test to identify if traffic was in cleartext, encoded or encrypted: having a high entropy is a strong indicator for encrypted payload. During their test, they utilized the Shannon entropy as a pre-step that allowed them to ignore encrypted communication and in return identify encoded[1] payload which, if examined manually, would have possibly been missed.

Wood et al. (2017) focused on 4 consumer IoT medical devices to identify cleartext between the device-to-cloud. Therefore, they first filtered for HTTP traffic, then identified unencrypted traffic using the chi-squared method and finally to identify sensitive personal information they used 3 different dictionaries to search for potential medical metadata. Although all the devices used encryption, they found the devices leaked metadata within HTTP GET requests, packet headers and the device conversation IP tables. Overall, they found cleartext data that showed when the device was being used.

Valente & Cardenas (2017) studied the communication of the device-to-cloud of smart toys where they found a vulnerability in one device: The 'Dino', although it encrypted its network traffic it used a weak encryption scheme and the same hard-coded keys to encrypt/decrypt traffic. This allowed the researchers to obtain cleartext information and retrieve the username, cryptography algorithm and encryption key.

The newest study found was by Servida & Casey (2019) who investigated 6 smart home devices and manually examined each device for cleartext traffic. As a result, they found encrypted and cleartext password on 2 devices; they also discovered 2 devices that communicated between the mobile app-to-device in cleartext that revealed authentication details. One of the devices communicated logs in cleartext and this contained information on events triggered, commands sent to the hub and requests made to the cloud.

Network traffic metadata. The main focus of Ren et al. (2019) work explores the privacy implications, this is based on whether the device's location would have an impact on the type of Personal Identifiable Information (PII) exposed. Additionally, they explored the different types of third parties the devices communicated with such as tracking and advertising services.

HTTP Proxy. While there has been considerable research focused on unencrypted data, there has been limited research on examining the content of encrypted traffic. A study by Kayode & Tosun (2019) intercepted the traffic of 6 smart home mobile apps as they communicated with the cloud. They were able to obtain the user ID, MAC address, home address, username and password from many of these devices. In another study by Chung et al. (2017), they used a proxy server to intercept encrypted traffic to obtain a list of unofficial APIs calls from the Amazon Alexa to extract forensic artefacts from the cloud. However, this was on the condition that user credentials were available.

3 Methodology

The apparatus used for the various experiments is presented in Table 1. For our experiment, the following stepwise procedure was used:

IoT device selection: 17 IoT devices were selected that represent a wide range from different categories. Details are provided in Section 3.1.

Port scanning: After setting up all devices, a port scan was completed to determine which ports are open on the IoT devices. We used Nmap and a quick type of scan for open TCP and UDP ports, using the following command `nmap -sS -sU -p 0-65535 [deviceIP]` (all ports). This step served as an active approach to analyse the devices.

[1] Encoded data is considered different to cleartext data. For instance, utilizing a Base64 encoding vs. sending ASCII characters.

Tools	Description	Utilisation
Raspberry Pi 3	Wireless access point (WAP)	Capture the wireless network traffic to and from the IoT devices
TCPdump	Network traffic collection	Automate capturing network traffic
Wireshark	Packet analyzer	Capture live network traffic
Huawei (ANE-LX1, Android 9)	Mobile device	Control/setup the IoT devices
PCAP Remote	Android app network sniffer for non-rooted mobile device	Capture network traffic from mobile apps
Fiddler	Proxy server to decrypts HTTPS traffic	Observe encrypted traffic in cleartext
Jadx-gui	Analysis APK files	Decompile the IoT mobile application
Network Miner	PCAP analyzer	Search for cleartext within PCAP files
Ent	Entropy test	Work out entropy value

Table 1: Apparatus utilised in the experiments.

Once the port scan was completed, we tried to connect to open ports using appropriate software, e.g., a browser for port 80 / 443, a SSH-client for port 22, and so on.

Network traffic collection: Given that IoT devices often come with supporting apps, e.g., for configuration, capturing all communication channels required various setups which are explained in Section 3.3.

Network traffic analysis: After capturing the data, we analysed the network traffic to determine if encryption had been used and the metadata to identify the location of the data, details are provided in Section 3.4.

Results: The results are then presented in Section 4.

HTTP proxy: We used a HTTP proxy to intercept TLS/SSL connections between the mobile app-to-cloud in order to reveal the cleartext contents of the encrypted traffic. We setup a proxy server, Fiddler [2] and installed the Fiddler CA certificate on a mobile device (Huawei P20 Lite) so that the HTTPs traffic could be decrypted. We examined 13 mobile apps as some can be used to control more than one device, e.g., Amazon Alexa (Section 4.3).

Tool creation: In parallel to analysing the data, a tool was created that helped us to more effectively handle the data. The tool is shared and will enable examiners to automatically extract cleartext data and identify where the data terminated (Section 5).

3.1 IoT device selection

As previously mentioned, 17 IoT devices were selected which were either wired or had wireless connectivity. In detail, we had: 3 smart hubs, several home automation devices (3 smart plugs, 2 smart bulbs), 6 smart cameras and 3 voice assistants. We also connected non-IP devices to the smart hubs that support other communication protocols such as Zigbee, Bluetooth and Zwave devices. Table 3 lists all details (e.g., model names) of the utilised devices. We had four major selection criteria when choosing the devices:

Variety of families: Selected devices had to belong to different device families. Consequently, our selection includes hubs, cameras, switches and smart speaker. Additionally, we ensured that we included different device manufacturers.

Country Purchased devices had to be available on the UK market; however they are often purchasable around the world.

Popularity: For each category of device we searched popular retail outlets, e.g., Amazon, based on price, popularity and average customer rating and reviews.

Compatibility with virtual assistants: When looking into the popularity of devices, we found that users favoured devices compatible with Amazon Alexa or Google Home Mini. Thus, if we had a choice between two particular devices, we chose one that was compatible.

Note, during our research we found two devices with previous security concerns which we also included: The Victure camera, which has over 9000 customer reviews[3], was said to have malware installed on the device and others were concerned regarding the sharing of personal data (Amazon, 2018a). Similarly, a reviewer of the Wansview camera collected network traffic and found their data was sent to China (Amazon, 2018b).

The devices were setup using the manufacturers mobile apps. Additionally, if the IoT device supported Alexa or Google Home Mini, it set up, too. We argue that this is a setup which is found in most households (except maybe for individuals with a computer science / cybersecurity background).

Expanding our test scenario. Additionally, we expanded our experiments by including an existing dataset created by Sivaraman et al. (2018). Originally, the authors used the dataset to classify IoT devices traffic into categories using machine learning, e.g., whether it is a light bulb or a home assistant. They stated their experiments ran between 1st October 2016 and 13th April 2017 and collected traffic from 28 IoT devices (cameras, switches, hubs etc.). However, the only datasets publicly available were between 23rd September 2016 and 12th October 2016[4]. Given the sheer amount of data, we selected 7

[2] https://www.telerik.com/fiddler (last accessed 2020-03-15).

[3] We understand that not all feedback is from genuine reviewers.

[4] https://iotanalytics.unsw.edu.au/iottraces

Category	Device Model	Open TCP ports	Open UDP ports	Vulnerable ports
SH [1]	Samsung SmartThings hub (v2)	8889,8890,39500	123,1900,5353	-
	Phillips Hue Bridge	80,443,8080	1900,5353	80 (HTTP)
	Vera hub	22,53,80,3480,49451	Closed/Filtered	22 (SSH),80 (HTTP)
HA [1]	iBlockcube smart plug	6668	49154	-
	Amazon smart plug	Closed	Filtered	-
	TP-Link switch (HS110)	9999	Filtered	-
	TP-Link bulb (LB100)	9999	Filtered	-
	LE LampUX	6668	49154	-
C [1]	TP-Link cam (KC100)	9999,10443,18443.19443	514	-
	D-Link cam (DCS-932LB)	80,443,8323	Closed/Filtered	80 (HTTP)
	Xiaomi cam	Closed	5353	-
	Yi cam	Closed	Closed	-
	Wansview cam (Q5)	8080, 554	3702, 16680, 17077, 28683, 19332, 19482, 19504, 23004, 33249, 332249, 41081, 41446, 58640	-
	Victure cam (PC530)	8080,554	65000, 67, 782, 1064, 2148, 4672, 6970, 6971, 19047, 20411, 20679,21131, 21354, 23176, 33744, 40724, 46836, 49199, 53037	65000
VA [1]	Amazon Echo (2nd gen)	4070	5353	-
	Amazon Echo (3rd gen)	4070,4071,55442,55443	Filtered	-
	Google Home Mini	8008,8009,8012,443,9000,10001	68,5000,5353	-

[1] Smart Hubs (SH), Home Automation (HA), Cameras (C), Voice Assistants (VA), Smart healthcare and Miscellaneous (M)

Table 2: Identified open TCP and UDP ports on the IoT devices.

days from the dataset (24^{th}/25^{th}/26^{th}/28^{th}/30^{th} September 2016 and 4^{th}/5^{th}/6^{th} October 2016) and excluded devices that overlap with ours. This left us with 15 new devices which are shown in Section 3. As the dataset is not labelled, we were unable to determine idle and interactive state. Remark: this was the only dataset including complete PCAP files. Other datasets from captures were reduced down to features and thus they are primarily relevant for machine learning.

3.2 *Port scanning*

To answer RQ1, we carried out port scans to identify open ports which may allow to connect to a devices. The result from the scans is shown in Table 2 and concludes that the majority of devices used well-known and proprietary ports (port range from 0-49151) we found 8 of the 17 devices used 'upper' TCP/UDP ports which range from ports 49152-65535. 3 devices used TLS/SSL (443), 2 had port 80 open which is typically used to run a HTTP and 1 allowed SSH (22, the Vera Plus hub). The root password to access SSH for this device was written on the hub, so root shell access was easy to gain. The Victure cam exposed a large number of TCP (2) and UDP (19) ports. In contrast to the 2 smart cameras Xiaomi and Yi where all ports were closed. This is beneficial from the security prospective but prevents an investigator gaining remote access to the device and acquiring the filesystem using traditional forensic tools. Often proprietary ports were used to communicate with the app. For instance, the TP-Link devices have port 9999 open in order to control the device using the mobile app.

We found the Victure cam has UDP port 65000 open and is commonly used by a specific trojan (Devil v1.3) (Chirillo, 2001). This is consistent with consumers findings from Section 3.1, where they thought malware had been preinstalled on the device. Also, Victure and Wansview cam have port 554 open which is used for Real Time Streaming Protocol (RTSP), although this port can potentially be exploited by sending specially-crafted RTSP packets, this only allows access to the live video stream (Speedguide.net, 2020).

3.3 *Network traffic collection*

In order to collect the network traffic, the devices were turned on and remained active for a duration of 7 days which was separated into two phases:

Idle-phase. The majority of the 7 days, the devices were in idle-phase. That is, the devices were turned on, but we did not perform any interactions on purposes. The network traffic was collected on a Raspberry Pi 3 running `TCPdump`.

Intensive-phase. This phase started after having the devices in idle-phase for roughly 60 minutes and included intensive interaction with all devices to increase the amount of produced traffic. The exact amount of interaction varied depending on the type of device, e.g., duration of voice command, powering on and off the device in short intervals. The type of interactions can be separated into the following three categories:

1. Mobile app and IoT device are on the same network,
2. Mobile app and IoT devices are connected to separate networks (forces utilization of the cloud infrastructure), and
3. Voice commands to trigger Amazon Echo voice assistant.

In order to capture traffic, we used different mechanisms depending on the setup:

Mobile app-to-device/Mobile app-to-cloud: During these scenarios, we used the Android app PCAP Remote[5] which allows the network traffic to be saved in a PCAP file. These were then transferred to a workstation for further analysis.

Device-to-cloud: To intercept the network traffic, a Raspberry Pi 3 was set up as a Wireless Access Point (WAP) which itself was then connected to a switch that allowed port mirroring. All traffic was captured running Wireshark on a connected Windows 10 workstation.

3.4 Network traffic analysis

We manually analysed all captured network traffic using `NetworkMiner`[6] and `Wireshark`. In `NetworkMiner` we used the cleartext dictionary file to carry out a customised search and conducted a string search (in various encoding) using `Wireshark` on the network traffic of each device. We looked for device identifiers (e.g. MAC address, serial numbers) and personal information created during setup (e.g. names, email address, passwords, usernames). As an initial step, the payload was analysed using the `Ent`[7] tool to look for traffic most likely to be unencrypted increasing the chances of finding information. We set the entropy test threshold to 7 (value of the test is between 0-8) to avoid missing unencrypted traffic. We also run the chi-squared test which did not yield any new results.

Analysis of metadata. In addition to the payload, the metadata was utilised where we primarily focused on the location of the connected cloud services. Therefore, a python script was developed to extract the destination IP address, host field and the number of bytes sent to that server from each IoT device, as each IoT device can contact many different servers / destinations. We used the destination IP address to identify the location using GeoIP[8] database and the host address using WHOIS data to identify the IoT cloud infrastructure. Details about the tool are provided in Section 5. When possible, we identify the type of cloud provider for an IoT device (e.g., Amazon AWS, Azure etc.). Lastly, we examined the existing dataset whose testbed was based in Australia in order to identify the final destination of the data.

4 Results

4.1 Utilization of encryption

This section addresses RQ2 wherefore we examined 32 IoT devices and found the majority had secure communication channels especially when the mobile app communicated with the cloud. We examined the unencrypted network traffic for any evidence potentially useful to an investigation. Although the majority of the devices encrypted their content unexpectedly, some devices would send in cleartext video of the detected motion and unique identifiable data during updates. Note, these devices mostly use encrypted channels but performed some actions using unencrypted channels.

Details. The majority of the devices used secure protocols TLS/SSL. However, we found 9 devices transmitting with no encryption (HTTP) or partially encrypted (TLS/HTTP) with the cloud or mobile app. 7 devices used no encryption between the device-to-cloud and 3 devices used no encryption between the mobile app-to-device. We found LE LampUX lightbulb and the iBlockcube plug used Internet Protocol Device Control (IPDC), which is unusual as this is typically used for Voice Over IP (VoIP). A reason for this might be that VoIP protocols are always prioritised by the router reducing latency in the device.

To identify unencrypted communication, we started with connections that had an average entropy score of 7 or below: Vera plus hub, LIFX bulb , D-Link camera, Samsung camera†, Insteon camera†, Victure cam, Wansview cam, Withings scale†, Withings monitor† and Withings sleep sensor†, the results of the entropy test are shown in 3. While the entropy test correctly detected the devices that used unencrypted traffic, the LiFX lightbulb† used encrypted protocols (TLS/SSL). It was discovered in previous research that the reason for the low entropy of the LiFX lightbulb† was because it encoded and not in a human-readable format (Loi et al., 2017).

When analysing the remaining traffic, we also identified the Xiaomi and D-Link cameras which are not using encryption but had higher entropy scores: the Xiaomi camera communicated to the cloud using no encryption and the D-Link camera communicated to the mobile app in cleartext. Both showed high entropy scores as these devices utilize video compression (Casino et al., 2019). We found that when the Xiaomi camera detected motion, the unencrypted video, MAC address and timestamp were sent in cleartext through a HTTP PUT request packet, a snippet of this is shown in Figure 1. The access key ID and signature are also present in the header, this can provide an investigator access to the AWS account. We also found that when the mobile app for the D-Link camera was activated, cleartext was present between the device and the mobile app during live streaming where partial JPEG images were present in the HTTP header.

The devices that communicated with the mobile app in cleartext included 4 devices; Vera hub, D-Link cam, Victure cam and Wansview cam. The only interesting finding was from the Victure cam: a HTTP POST requests which included the API access token and key.

Next, we examined the 7 devices that communicated with the cloud in cleartext; 2 of these devices were smart cameras. The first is the Samsung camera† that sent unencrypted HTTP POST

[5] https://play.google.com/store/apps/details?id=com.egorovandreyrm.pcapremote&hl=en_GB (last accessed 2020-03-15).

[6] https://www.netresec.com/?page=NetworkMiner

[7] https://github.com/rsmith-nl/ent (last accessed 2020-03-15).

[8] https://dev.maxmind.com/geoip/geoip2/geolite2/ (last accessed 2020-03-15).

† Marked IoT devices are from the expanded test scenario.

		Device-to-cloud			Mobile app-to-cloud			Mobile app-to-device		
Category	**Device Model**	**Entropy**	**Cleartext**	**Protocol**	**Entropy**	**Cleartext**	**Protocol**	**Entropy**	**Cleartext**	**Protocol**
SH [1]	Samsung SmartThings hub (v2)(wired)	7.78	✗	TLSv1.2	-	✗	TLSv1.2	-	✗	TLS1.2
	Phillips Hue Bridge(wired) [3,4]	7.72	✓	HTTP/TLSv1.2	7.99	✗	TLSv1.2	7.81	✗	TLS1.2
	Vera plus hub(wired)[2]	7.87	✗	TLSv1.2	6.24	✗	HTTP/TLSv1.2	6.54	✓	HTTP
HA [1]	iBlockcube smart plug	7.74	✗	TLSv1.2	7.22	✗	TLSv1.2	7.45	✗	IPDC
	Amazon smart plug	7.80	✗	TLSv1.2	-	✗	TLSv1.2	7.54	✗	-
	TP-Link plug(HS110)	7.74	✗	TLSv1.2	7.74	✗	TLSv1.2	7.56	✗	TLSv1.2
	TP-Link bulb(LB100)	7.72	✗	TLSv1.2	7.20	✗	TLSv1.2	7.23	✗	TLSv1.2
	LE LampUX	7.87	✗	TLSv1.2	7.28	✗	TLSv1.2	7.91	✗	IPDC
	LiFX lightbulbs†	6.12	✓	TLSv1	-	✗	-	-	✗	-
	iHome†	7.59	✗	TLSv1.2	-	✗	-	-	✗	-
	Nest Protect smoke alarm†	7.67	✗	TLSv1.2	-	✗	-	-	✗	-
C [1]	TP-Link camera(KC100)	7.99	✗	TLSv1.2	7.97	✗	TLSv1.2	7.81	✗	TLSv1.2
	D-Link camera(DCS-932LB)(wired)	7.82	✗	TLSv1	7.78	✗	TLSv1.2	6.40	✓	HTTP
	Xiaomi camera	7.91	✓	HTTP	7.91	✗	TLSv1.2	-	✗	-
	Yi camera	7.92	✗	TLSv1.2	7.59	✗	TLSv1.2	-	✗	-
	Netatmo Welcome camera†	7.20	✗	TLSv1.2	-	✗	-	-	✗	-
	TP-Link Day Night Cloud camera†	7.41	✗	TLSv1.2	-	✗	-	-	✗	-
	Samsung SmartCam camera†	6.05	✓	HTTP/TLSv1.2	-	✗	-	-	✗	-
	Nest Dropcam†	7.78	✗	TLSv1.2	-	✗	-	-	✗	-
	Insteon camera(wired)†	6.57	✓	HTTP/TLSv1.2	-	✗	-	-	✗	-
	Victure cam (PC530)	6.06	✗	-	6.45	✗	HTTP	5.74	✗	HTTP
	Wansview cam (Q5)	7.86	✓	TLSv1.2	5.86	✗	HTTP/TLSv1.2	6.68	✗	HTTP
VA [1]	Amazon Echo(2nd gen)	7.99	✗	TLSv1.2	-	✗	TLSv1.2	-	✗	-
	Amazon Echo(3rd gen)	7.97	✗	TLSv1.2	-	✗	TLSv1.2	-	✗	-
	Google Home Mini	7.99	✗	TLSv1.3	7.72	✗	TLSv1.2	7.91	✗	TLSv1.2
S [1]	Withings smart scale†	5.69	✓	HTTP	-	✗	-	-	✗	-
	Withings smart baby monitor†	5.69	✓	HTTP	-	✗	-	-	✗	-
	Withings aura smart sleep sensor†	6.17	✓	HTTP/TLSv1.2	-	✗	-	-	✗	-
	Blipcare blood pressure meter†	7.48	✗	TLSv1	-	✗	-	-	✗	-
M [1]	Netatmo weather station†	7.40	✗	-	-	✗	-	-	✗	-
	Triby speaker†	7.59	✗	TLSv1.2	-	✗	-	-	✗	-
	PIX-STAR photo-frame†	7.48	✗	TLSv1.2	-	✗	-	-	✗	-

[1] Smart Hubs (SH), Home Automation (HA), Cameras (C), Voice Assistants (VA), Smart healthcare and Miscellaneous (M)
[2] On the Vera hub we connected the Aeotec Door/Window sensor Gen5 (ZW120-C), this device uses Z-wave. In order to generate data as none of the other devices were compatible with this hub
[3] On the Phillip Hue Bridge we connected a Phillips lightstrip that uses Zigbee
† Devices from the existing dataset

Table 3: The 32 IoT devices used in the experiments, the highlighted entropy values correspond to devices that did not use encryption

```
PUT /ipc009-video-storage
    /2019/12/17/6272883219/283973403\_195222622.
    mp4\?
Expires=1576585342000&
GalaxyAccessKeyId=5371742894246&
Signature=GLNOHtIhIRqzzFAWRNTOs7rAjZk= HTTP/1.1
Host: awsde0.fds.api.xiaomi.com
```

Figure 1: Snippet taken from the unencrypted HTTP PUT request from the Xiaomi camera.

requests to the cloud, which exposed unique identifiers including, MAC address, username, serial number, timestamp and user specific device name (e.g. 'smarthomeunsw'). The second camera an Insteon† also displayed cleartext information such as port numbers, MAC address, public IP address and unique ID. The remaining 3 devices were smart health care devices that required personal data such as height, weight, postcode/zipcode etc. All this data is sensitive and helpful not just in identifying the user but also their physical characteristics. The 3 devices manufactured by Withings† (Sleep, baby monitor and scales) all sent cleartext data through HTTP POST requests. Although the Withings baby monitor and sleep† did not contain any sensitive cleartext information, the Withings smart scales† displayed considerable amount of user information, e.g., weight, height, as shown in Figure 2. Although the API for the scales had depreciated, to understand the data we used the following source[9].

[9] https://blog.chris007.de/hacking-the-withings-wifi-body-

Communication Channel	Device Model	Entropy	Protocol	Cleartext data
Device-to-cloud	LiFX lightbulbs	6.12	TLSv1	Loi et al. (2017)
	Samsung SmartCam	6.05	HTTP/TLSv1	MAC address, username, serial number, timestamp and user specified device name
	Insteon Camera(wired)	6.57	HTTP	Port numbers, MAC address, public IP address, unique ID
	Withings smart scale	4.68	HTTP	Weight,height,age, sex etc.
	Triby speakers	7.59	TLSv1.2	Username, serial number, MAC address
	Xiaomi camera	7.91	HTTP	URL and timestamp of captured motion, MAC address
Mobile app-to-device	D-Link camera	6.40	HTTP	JPEG images
	Victure camera	6.45	HTTP	API access key, access token

Table 4: Summary of findings when looking at unencrypted communication.

```
"id":11020336, #Numeric ID of user
"sn":"SMA", #Username of user
"wt":51.701, #Weight(kgs)
"ht":1.68, #Height(meters)
"agt":30.1, #Age in years
"sx":0, #Sex of user(0=male, 1=female)
"cr":1472696125, #Unix timestamp when account
    created
"lg":"fr_FR", #Language French
"utc":1474829944 #Unix timestamp of last
    measurement (Unix)
```

Figure 2: Snippet taken from the cleartext HTTP POST request of the Withings smart scale.

An unexpected finding was when the Triby speaker[†] communicated with the cloud during an update, the HTTP GET request is displayed in cleartext and included information such as MAC address, username and serial number as shown in Figure 3. We did not find any sensitive cleartext data on the Phillip Hue Bridge or Vera Plus hub. However, these provide a central gateway to connect other devices so when new sensors / devices are connected in the future, we expect cleartext data to be identified. This is especially true for the Phillips Hue Bridge which only uses partial encryption between the device-to-cloud communication channel.

```
GET /update/triby/update.pup?mac=18:B7:9E
    :02:20:44&machine=triby&board_name=TRIBY_V0&
    revision=5&sub_revision=4271&version=triby
    -10.48.1&version_build=52.3a.e0&up=691259&
    hwcap=1:0&feature_tag=&board_flags=c:1,ci:0,
    vl:0,cv:0,p:1 HTTP/1.1
Host: developer.invoxia.com
```

Figure 3: Snippet taken from the unencrypted HTTP GET request during an update of the Triby speaker.

Previous research by Loi et al. (2017) studied 20 IoT devices and found 5 that communicated in cleartext, 3 of these devices were smart cameras. This corresponds with our findings where we found 6 of the 8 devices that sent cleartext were smart cameras. A possible reason for this could be that smart cameras have more features and services when compared to other devices such as smart plugs. For instance, TP-Link plug had 1 TCP port open while the TP-Link camera had 4 TCP ports open.

4.2 *Network traffic metadata*

In the following we focus on RQ3 and identify the destinations that the data transverses. Figure 4 shows the flow of traffic to the top 10 countries with the height of the bands corresponding to the number of bytes sent by each IoT device. Overall, the data for 26 of the 32 devices terminated in the US as shown in Table 5. This is unexpected as the closest Amazon data centre to our testbed (UK) is in Ireland (WikiLeaks, 2018). While trying to establish a reason for our data being sent to the US instead of Ireland, we found that unlike Google and Facebook that provide approximate locations to their data centres, Amazon do not advertise this freely.

Country	Number of Devices
United States	26
France	14
Germany	14
United Kingdom	13
Ireland	11
Netherlands	8
Australia	7
Singapore	5
Denmark	5
Finland	4
India	4
Japan	3
Mexico	3
Romania	3
China	3
Brazil	2
Russia	2

Table 5: The number of devices connected to each country

This coincides with previously report results. For instance, Tilley (2017) reported that data from their Ring Doorbell was routed to servers in China. We found only 3 devices; Insteon,

scale-2/ (last accessed 2020-03-15).

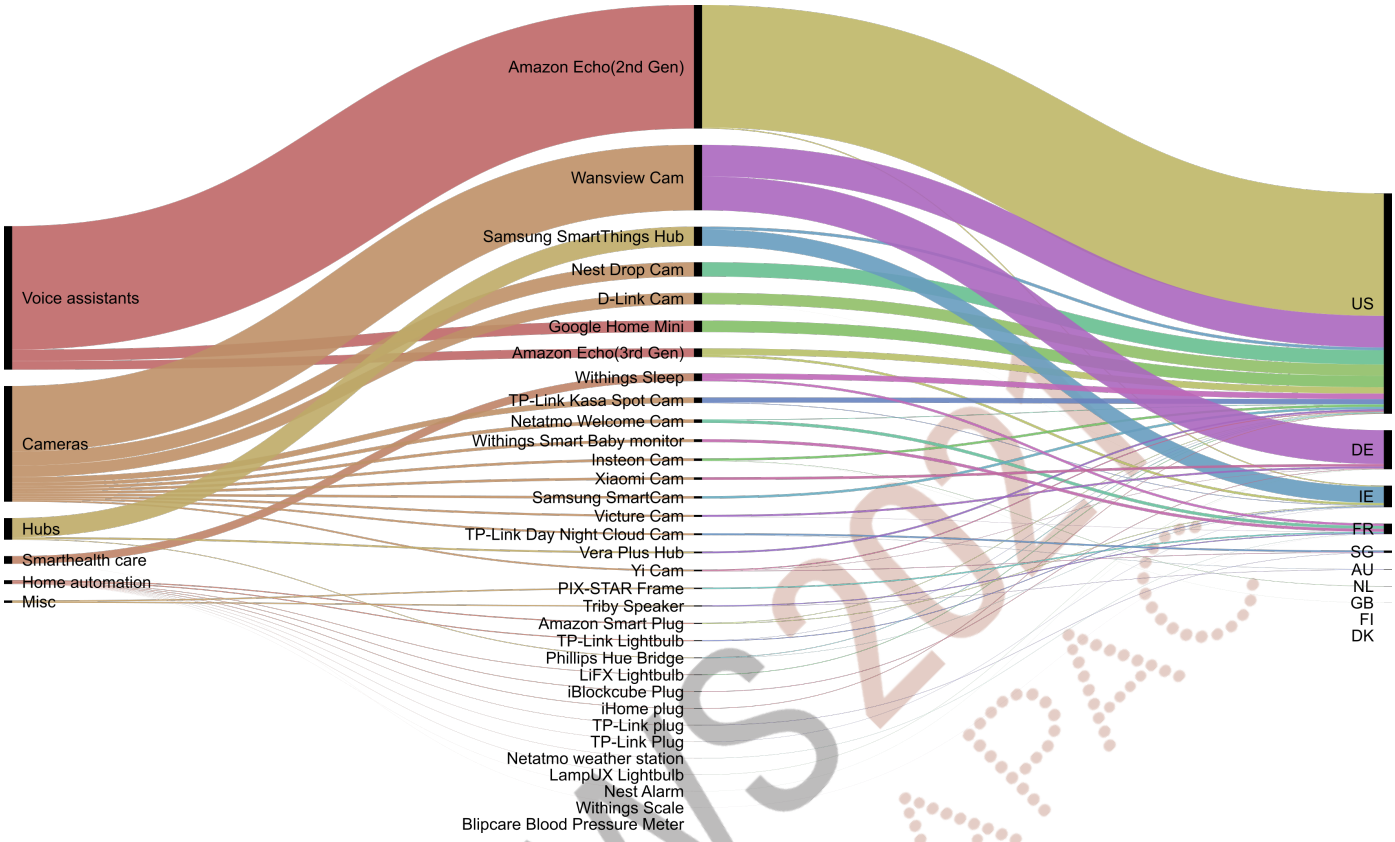

Figure 4: The top 10 data destinations grouped by the number of bytes transmitted by each device to their final destination.

Wansview and Xiaomi cameras that sent their data to Alibaba servers based in China. In Section 3.1 the consumer raised the concern that their data was sent to China, this was confirmed in our findings regarding the Wansview camera.

Furthermore, we found 75% (24) of the IoT devices sent data to multiple destinations, the remaining devices sent data to a single destination. From an investigative point of view, this is interesting as multi destinations and jurisdictions can potentially cause delays in gaining access and securing the data. It will also require communication between countries with different legal systems.

Next, we examined the devices that contacted the most destinations the results are shown in Appendix A, this was the LiFX lightbulb (28) followed by the TP-Link Bulb, TP-Link plug, Vera Plus Hub (all had 9) and the Insteon cam (7). When compared to other devices, such as smart hubs and cameras lightbulbs, they have limited features, yet surprisingly they were the devices that contacted the most destinations.

To gain a better understanding of the type of cloud infrastructure data was sent to, this part of the analysis focused on the most frequent type used: we found 21 of the 32 devices contacted a server belonging to Amazon. The reliance on the Amazon servers to store data could be an issue for investigators. This was demonstrated in a recent case involving Amazon Alexa where Amazon refused to hand over evidence relating to a criminal investigation and denied the request in absent of a valid or binding legal demand. This could also be an issue for future cases when dealing with Amazon (Cuthbertson, 2018).

4.3 HTTP Proxy

In this section we used a proxy server to examine the encrypted contents of the network traffic (see RQ4). We found several of the mobile apps allowed a proxy connection while the remaining implemented certificate pinning. We found a case where a device would unexpectedly take snapshots and there was no setting to control this behaviour. On the same mobile app, we found the username and password were sent Base64 encoded. As this mobile app controls several devices, this means we were able to gain access to them all.

Details. We observed the decrypted communication between the mobile app-to-cloud and found 7 of the 13 apps allowed proxy connections. The remaining apps did not complete a connection with their servers as they had certificate pinning implemented, which involves the coupling of a host's trusted credentials to its identity (e.g. an X.509 certificate or public key) (Onwuzurike & De Cristofaro, 2015). We decompiled the apps and searched for keywords such as 'x509' and 'checkServerTrusted' and found further evidence of certificate pinning.

The apps iLiving-iBlock and LampUX allowed a proxy connection and normal device use, e.g., turn on/off, although we did not find any sensitive data transmitted by these devices. We found that when we opened the YI cam app, it would send a list

of URLs which contained the motion captured, username, user ID and API key. The Kasa app controlled the TP-Link devices (lightbulb, switch and camera), but only the camera routed useful data through the proxy. There was an unusual activity of the TP-Link camera when the app was opened: it took a snapshot which included a timestamp and URL link to the JPEG snapshot (note we were not able to control / disable this functionality). From the same mobile app, we captured HTTP GET request packets exposing the basic authentication field that contained the username and password to login for the device, which was encoded in Base64, which is shown in Figure 5. Obtaining the password for one account could lead an investigator to access other IoT device accounts. Especially as Wang et al. (2018) found that 52% of users reused their passwords for many online services.

```
POST /data/LINKIE.json HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Authorization: Basic XXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXX
User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; ANE
    -LX1 Build/HUAWEIANE-L21)
Host: 192.168.1.151:10443
```

Figure 5: Snippet taken from the intercepted encrypted HTTPS POST request of the TP-link cam. The values have been obfuscated for confidentiality reasons.

5 Tool creation and Evaluation: `IoT Network Analyzer`

Although our results can be found using separate open source tools, it would require an investigator considerable amount of time to manually extract the data. Consequently, we implemented part of the process in a tool called `IoT Network Analyzer`, constructed in Python to automate the process. The tool is currently a working prototype and has the following four main features:

Entropy calculation: In order to identify sessions that are in clear text, the Shannon entropy is utilised and calculated over all packets within a session. This calculation is only done on the data portion of the segment; header fields such as ports or IPs of the segment are ignored.

Location of data: The source and destination IP address are extracted to make an assumption of the geolocation. This was accomplished using IP Stack[10] which is a "powerful, real-time IP to geolocation API".

Usage of secure ports: First a list of ports is created with the total number of occurrences for each port. This list is then checked against the pre-defined list of 22 secure ports (Appendix B) which can be modified if needed. All this information is then mapped with their corresponding source and destination IP address.

Cleartext extraction: Lastly, our tool tries to extract cleartext.

The tool also has the following additional features:

IP addresses and connections overview: A list of the source and destination IP address and their respective number of packets sent.

Packets overview: It analyzes the structure of the different packets and outputs a list of the number of packets including; the total, raw layer, UTF8, byte, ARP, DHCP, DNS, and finally the total amount of packets processed.

Command line interface (CLI): The tool has an interactive CLI based on the cmd2[11] library. This allows an interactive usage of the tool, data extraction and output capabilities.

The tool is available at: https://github.com/Dyvels/IoTAnalyzer

5.1 *Tool assessment*

For the evaluation we tested the tool's four main features and the performance of the tool, utilising 5 PCAP files that we knew had cleartext data and another 5 PCAP files with no cleartext data.

Entropy calculation. To evaluate the entropy part, we compared our results to a similar tool called `Ent` and they provided identical results. Our tool has the benefit of automatically calculating the entropy based on the payload whereas `Ent` requires manual filtering of the IP addresses and extraction of the payloads. This means using our tool an investigator is able to calculate the entropy on the various communication channels at the same time.

Location of data. As we use the external IPStack service for detecting the location, we briefly compared these results with two similar services: IP2Location-Lite[12] and Geolite2 (MaxMind)[13]. These have been suggested by Gharaibeh et al. (2017) as they have an accuracy of 80% at country-level. The results are shown in Table 6 and show little difference between the databases, meaning any of these databases would be equally suitable. Note: a method to assess the accuracy of IPStack would be to compare it against a "ground truth" database that has true geographical location for each IP address, however such a database does not exist (Gharaibeh et al., 2017).

Usage of secure ports. To test the accuracy of this feature we compared the results to those of a similar tool T-shark. We analysed the 10 PCAP files using T-shark to filter for the ports and then compared these results to our tool, with both tool showing identical results.

[10] https://ipstack.com/product

[11] https://github.com/python-cmd2/cmd2

[12] https://www.ip2location.com/demo

[13] https://dev.maxmind.com/geoip/geoip2/geolite2/

Devices	IPStack (IoT Network Analyzer)	GeoIP	IP2location
Withings smart scale	France	France	France
Samsung smartcam	Australia, New Zealand, US	Australia, Cambodia, Germany, Luxembourg, UK, US	Australia, Japan, New Zealand, UK, US
Vera hub	US	US	US
DLink cam	Ireland	Ireland	Ireland
Insteon cam(wired)	Australia, China, Ireland, Japan, Netherlands, UK,US,	Germany, Ireland, Singapore, UK, US	China, France, Germany, Taiwan, UK, US
TPlink cam	Germany, Ireland, Singapore, UK, US	France, Germany, Ireland, Singapore, US	China, France, Germany, UK, US
Amazon Echo	Ireland, UK, US	Ireland, UK, US,	Ireland, UK, US
Google Mini	Mexico, US	Mexico, US	UK, US
Phillips Hue	Germany, Ireland, Singapore, Spain, UK, US	Germany, Ireland, Singapore, Spain, UK, US	Germany, Ireland, UK, US
Xiaomi cam	China, France, Germany, Netherlands, Singapore, Taiwan, UK	China, France, Germany, Ireland, Japan, Netherlands, Singapore, South Korea, Taiwan, UK, US	China, France, Germany, Netherlands, Singapore, Taiwan, UK

Table 6: Countries identified using IoT Network Analyzer and results from two other geolocation databases

Cleartext extraction. To identify cleartext traffic, we manually examined each PCAP file in `NetworkMiner`[14] followed by a comparison using our tool. NetworkMiner and our tool identified the same PCAP files containing cleartext information.

Performance evaluation. The performance test (with respect to processing speed, Central Processing Unit (CPU) and memory usage) was conducted on a system running Windows 10 on a Intel(R) Core(TM) CPU i5-4670K with 32GB RAM. The results from processing the PCAP files of sizes from 500 - 75,000 kilobytes (kbs) are shown in Figure 6, and shows that it takes around 1 second to 6 minutes. Table 7 shows that the tool is not resource intensive as it consumes around 60% CPU and has small memory footprint.

Filesize (kB)	CPU usage (%)	Memory usage (%)
500	35	0.38
2000	40	0.73
10,000	52	1.16
50,000	59	7.76
75,000	65	8.67

Table 7: Results from the performance evaluation

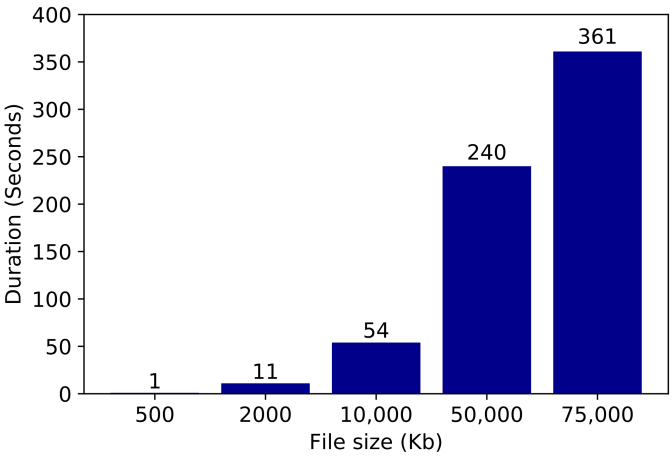

Figure 6: Different sizes of PCAP files and the time required to process

6 Conclusion

In this paper we analysed 32 different IoT devices with respect to their network settings to better understand implications for forensic practitioners. While we found several problems with existing devices, our results show improvements compared to previous studies. For instance, Loi et al. (2017) found that access through Telnet and SSH was prevalent; Cimpanu (2020) reported about a leaked list containing over half a million credentials that allow to access IoT devices via telnet dated October-November 2019 (the age of the devices is unknown).

The devices that sent data in cleartext were smart cameras and smart healthcare devices. Smart healthcare devices exposed personal data potentially useful for an investigator to identify features of a person of interest. Another finding was when the Xiaomi camera detected motion, it would send an unencrypted packet containing not only the captured video but also the credentials to an AWS server. As the number of IoT devices are increasing, investigators will become overwhelmed with devices to analyse, therefore the entropy test will be as a useful tool in triaging devices that have unencrypted traffic.

One of the forensic challenges discussed in existing work is the storage of IoT data in multiple locations which then leads to different jurisdiction (Yaqoob et al., 2019; Hegarty et al., 2014). In our findings, the majority of data was sent to the US, however, there were also plenty of other countries (see Section 4.2). Note, this was despite our testbed being based in the UK and the 2^{nd} dataset collected in Australia. There would be less legal complexity if the data was stored in data centres within the country of origin. These legal difficulties were highlighted in a case involving Microsoft where they refused to comply with

[14] https://www.netresec.com/?page=NetworkMiner

a US search warrant because the data was not stored in the US but in the EU (Carswell, 2016). Additionally, we found 77% of the devices' data was sent to multiple destinations and 64% of the IoT devices contacted a server belonging to Amazon.

From examining the encrypted content, we found an unexpected event where the Kasa app for the TP link devices would take snapshots which could not be controlled. On the same mobile app, we found the username and password used a weak HTTP based authentication that could easily be decoded. Several other mobile apps accepted proxy connections, however, these were smart plugs and did not have any sensitive data.

References

Alrawi, O., Lever, C., Antonakakis, M., & Monrose, F. (2019). SoK: Security Evaluation of Home-Based IoT Deployments. *Proceedings - IEEE Symposium on Security and Privacy*, *2019-May*, 1362–1380. doi:10.1109/SP.2019.00013.

Amazon (2018a). Amazon. https://www.amazon.co.uk/product-reviews/B07CB14GTB/ref=acr_dp_hist_1?ie=UTF8&filterByStar=one_star&reviewerType=all_reviews#reviews-filter-bar.

Amazon (2018b). Amazon. https://www.amazon.co.uk/product-reviews/B07QLR94S1/ref=cm_cr_getr_d_paging_btm_next_4?ie=UTF8&filterByStar=one_star&reviewerType=all_reviews&pageNumber=4#reviews-filter-bar.

Buhecker, R. (2018). FitBit and Wearables Proving to be a Valuable Tool in Forensics. https://www.secureforensics.com/blog/fitbit-and-wearables-proving-to-be-a-valuable-tool-in-forensics.

Carswell, S. (2016). Microsoft warns of risks to Irish operation in US search warrant case. https://www.irishtimes.com/business/microsoft-warns-of-risks-to-irish-operation-in-us-search-warrant-case-1.2548718.

Casino, F., Choo, K. K. R., & Patsakis, C. (2019). HEDGE: Efficient Traffic Classification of Encrypted and Compressed Packets. *IEEE Transactions on Information Forensics and Security*, *14*, 2916–2926. doi:10.1109/TIFS.2019.2911156. arXiv:1905.11873.

Chirillo, J. (2001). *Hack Attacks Revealed: A Complete Reference with Custom Security Hacking Toolkit*. USA: John Wiley & Sons, Inc.

Chung, H., Park, J., & Lee, S. (2017). Digital forensic approaches for amazon alexa ecosystem. *Digital Investigation*, *22*, S15 – S25. URL: http://www.sciencedirect.com/science/article/pii/S1742287617301974. doi:https://doi.org/10.1016/j.diin.2017.06.010.

Cimpanu, C. (2020). Hacker leaks passwords for more than 500,000 servers, routers, and IoT devices. https://www.zdnet.com/article/hacker-leaks-passwords-for-more-than-500000-servers-routers-and-iot-devices/.

Cuthbertson, A. (2018). Amazon ordered to give Alexa evidence in double murder case. https://www.independent.co.uk/life-style/gadgets-and-tech/news/amazon-echo-alexa-evidence-murder-case-a8633551.html.

Gharaibeh, M., Zhang, H., Shah, A., Ensafi, R., Huffaker, B., & Papadopoulos, C. (2017). A look at router geolocation in public and commercial databases. *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, *Part F131937*, 463–469. doi:10.1145/3131365.3131380.

Hegarty, R., Lamb, D. J., & Attwood, A. (2014). Digital evidence challenges in the internet of things. In *INC*.

Kayode, O., & Tosun, A. S. (2019). Analysis of IoT Traffic using HTTP Proxy. *IEEE International Conference on Communications*, *2019-May*, 1–7. doi:10.1109/ICC.2019.8761601.

Kumar, D., Shen, K., Case, B., Garg, D., Alperovich, G., Kuznetsov, D., Kuznetsov, D., Gupta, R., & Durumeric, Z. (2019). All things considered: An analysis of iot devices on home networks. In *Proceedings of the 28th USENIX Conference on Security Symposium* SEC'19 (p. 1169–1185). USA: USENIX Association.

Loi, F., Sivanathan, A., Gharakheili, H. H., Radford, A., & Sivaraman, V. (2017). Systematically evaluating security and privacy for consumer IoT devices. *IoT S and P 2017 - Proceedings of the 2017 Workshop on Internet of Things Security and Privacy, co-located with CCS 2017*, (pp. 1–6). doi:10.1145/3139937.3139938.

Onwuzurike, L., & De Cristofaro, E. (2015). Short: Danger is my middle name - Experimenting with SSL vulnerabilities in android apps. *Proceedings of the 8th ACM Conference on Security and Privacy in Wireless and Mobile Networks, WiSec 2015*, . doi:10.1145/2766498.2766522.

Ren, J., Dubois, D. J., Choffnes, D., Mandalari, A. M., Kolcun, R., & Haddadi, H. (2019). Information exposure from consumer iot devices: A multidimensional, network-informed measurement approach. In *Proceedings of the Internet Measurement Conference* IMC '19 (p. 267–279). New York, NY, USA: Association for Computing Machinery. URL: https://doi.org/10.1145/3355369.3355577. doi:10.1145/3355369.3355577.

Servida, F., & Casey, E. (2019). Iot forensic challenges and opportunities for digital traces. *Digital Investigation*, *28*, S22–S29. URL: https://doi.org/10.1016/j.diin.2019.01.012. doi:10.1016/j.diin.2019.01.012.

Shannon, C. E. (1949). Communication theory of secrecy systems. *The Bell System Technical Journal*, *28*, 656–715. doi:10.1002/j.1538-7305.1949.tb00928.x.

Sivaraman, V., Loi, F., Habibi Gharakheili, H., Sivanathan, A., Vishwanath, A., Wijenayake, C., & Radford, A. (2018). Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics. *IEEE Transactions on Mobile Computing*, . doi:10.1109/tmc.2018.2866249.

Speedguide.net (2020). Port 554 Details. https://www.speedguide.net/port.php?port=554.

Tilley, A. (2017). This Smart Doorbell Was Accidentally Sending Data To China, Until People Started Freaking Out. https://www.forbes.com/sites/aarontilley/2017/03/22/this-smart-doorbell-was-accidentally-sending-data-to-china-until-people-started-freaking-out/#612e957d5984.

Valente, J., & Cardenas, A. A. (2017). Security & privacy in smart toys. *IoT S and P 2017 - Proceedings of the 2017 Workshop on Internet of Things Security and Privacy, co-located with CCS 2017*, (pp. 19–24). doi:10.1145/3139937.3139947.

Wang, C., Jan, S. T., Hu, H., Bossart, D., & Wang, G. (2018). The next domino to fall: Empirical analysis of user passwords across online services. *CODASPY 2018 - Proceedings of the 8th ACM Conference on Data and Application Security and Privacy*, *2018-January*, 196–203. doi:10.1145/3176258.3176332.

WikiLeaks (2018). Map of Amazon's Data Centers. https://wikileaks.org/amazon-atlas/map/.

Wood, D., Apthorpe, N., & Feamster, N. (2017). Cleartext data transmissions in consumer iot medical devices. In *IoT S and P 2017 - Proceedings of the 2017 Workshop on Internet of Things Security and Privacy, co-located with CCS 2017* IoT S and P 2017 - Proceedings of the 2017 Workshop on Internet of Things Security and Privacy, co-located with CCS 2017 (pp. 7–12). Association for Computing Machinery, Inc. doi:10.1145/3139937.3139939.

Wu, T., Breitinger, F., & Baggili, I. (2019). Iot ignorance is digital forensics research bliss: A survey to understand iot forensics definitions, challenges and future research directions. In *Proceedings of the 14th International Conference on Availability, Reliability and Security* (pp. 1–15).

Yaqoob, I., Hashem, I. A. T., Ahmed, A., Kazmi, S. A., & Hong, C. S. (2019). Internet of things forensics: Recent advances, taxonomy, requirements, and open challenges. *Future Generation Computer Systems*, *92*, 265 – 275. URL: http://www.sciencedirect.com/science/article/pii/S0167739X18315644. doi:https://doi.org/10.1016/j.future.2018.09.058.

Appendix A The IoT devices that contacted multiple countries

	Countries																																		
Devices	AT	AU	BG	BR	BY	CA	CH	CN	DE	DK	FI	FR	GB	HK	HU	ID	IE	IN	IS	JP	KH	KR	LU	MX	NL	NO	NZ	PL	RO	RU	SG	SK	TW	UA	US	XA	Total
Amazon Echo (3^{rd})	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	2
Amazon Echo(2^{nd})	○	○	○	○	○	○	○	○	○	○	○	○	◉	◉	○	○	◉	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉	○	4
Amazon Smart Plug	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉	○	2
D-Link camera	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉	○	2
Google Home Mini	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉	○	○	○	○	○	○	○	○	○	○	◉	○	2
Insteon Cam	○	○	○	○	○	○	○	◉	○	○	○	○	◉	○	○	○	◉	○	○	◉	○	○	○	○	◉	○	○	○	○	○	◉	○	○	○	◉	○	7
LiFX lightbulb	◉	◉	◉	◉	◉	○	◉	○	◉	◉	○	◉	◉	○	◉	◉	◉	○	◉	◉	○	○	○	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	◉	28
Phillips Hue Bridge	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉	○	○	○	○	○	○	◉	○	○	○	○	○	○	○	○	○	○	◉	○	3
Samsung Smart Camera	○	◉	○	○	○	○	○	○	◉	○	○	○	◉	○	○	○	○	○	○	○	◉	○	◉	○	○	○	○	○	○	○	○	○	○	○	◉	○	6
Samsung SmartThings Hub	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉	○	2
TP-Link Bulb	○	○	○	○	○	○	○	○	◉	◉	◉	◉	◉	○	○	○	◉	◉	○	○	○	○	○	○	◉	○	○	○	○	○	○	○	○	○	◉	○	9
TP-Link Day Night Cloud camera	○	◉	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉	○	○	○	◉	○	3
TP-Link Kasa Spot Cam	○	○	○	○	○	○	○	○	○	○	○	◉	◉	○	○	○	◉	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉	○	4
TP-Link plug	○	○	○	○	○	○	○	○	◉	○	○	◉	◉	○	○	○	○	○	○	○	○	○	○	○	◉	○	○	○	○	○	◉	○	○	○	◉	○	6
TP-Link Switch	○	○	○	○	○	○	○	○	◉	◉	◉	◉	◉	○	○	○	◉	◉	○	○	○	○	○	○	◉	○	○	○	○	○	○	○	○	○	◉	○	9
Triby Speaker	○	◉	○	○	○	○	○	○	○	○	◉	◉	◉	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉	○	○	○	○	○	◉	○	6
Vera Plus Hub	○	○	○	○	○	○	○	○	◉	◉	◉	◉	◉	○	○	○	○	◉	○	○	○	○	○	○	◉	○	○	○	◉	○	○	○	○	○	◉	○	9
Withings Sleep	○	○	○	○	○	○	○	○	◉	○	○	◉	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉	○	3
Withings Smart Baby monitor	○	◉	○	○	○	◉	○	○	○	○	○	◉	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	3
Xiaomi Cam	○	○	○	○	○	○	○	◉	◉	○	○	○	○	○	○	○	○	○	○	○	○	◉	○	○	◉	○	○	○	○	○	○	○	○	○	◉	○	5
Yi Cam	○	○	○	○	○	○	○	○	◉	○	○	○	◉	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉	○	○	○	◉	○	4
Wansview Cam	○	○	○	◉	○	○	◉	◉	◉	○	○	○	◉	○	○	◉	○	○	○	○	◉	○	○	○	○	○	○	◉	○	○	○	○	◉	○	◉	○	10
Victure Cam	○	○	○	○	○	○	○	○	◉	○	○	◉	◉	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	◉	○	4

Appendix B List of secure ports

Port	Description
500	IPSec (VPN tunneling)
443	Transport Layer Security (TLS)
22	Secure Shell (SSH)
8883	Secure MQTT
6679	IRC over SSL (Secure Internet Relay Chat)
6697	IRC over SSL (Secure Internet Relay Chat)
9090	Webwasher, Secure Web, McAfee Web Gateway – Default Proxy Port, Manage Engine Applications Manager
9091	Openfire Administration Console (SSL Secured)
19999	DNP – Secure (Distributed Network Protocol – Secure), a secure version of the protocol used in SCADA systems, communication used for the RTU's and IED's
1884	Internet Distance Map Svc
5684	Constrained Application Protocol (CoAP)
1311	Dell Open Manage HTTPS
1920	IBM Tivoli Monitoring Console (HTTPS)
4712	McAfee Web Gateway 7 – Voreingestellter GUI Port HTTPS
5001	Synology HTTPS WebUI (DSM)
8243	HTTPS for Apache Synapse
8444	PCsync HTTP
8531	WSUS HTTPS Standardport[66]
14943	Trend Micro ServerProtect for Linux (SPLX) 3.0 web console can be accessed using HTTPS (Hypertext Transfer Protocol over SSL/TLS)
21012	AMLFilter, AMLFilter Inc. amlf-engine-01 HTTPS Standardport
21022	AMLFilter, AMLFilter Inc. amlf-engine-02 HTTPS Standardport