Towards a Framework for Testing the Security of IoT Devices Consistently

Gurjan Lally[1], Daniele Sgandurra[2]

[1] Department of Computer Science, Royal Holloway, University of London, UK
gurjan.lally.2015@live.rhul.ac.uk
[2] Information Security Group, Royal Holloway, University of London, UK
daniele.sgandurra@rhul.ac.uk

Abstract: The Internet of Things (IoT) permeates society in many areas, such as automotive, smart-homes, smart-cities, healthcare, and critical infrastructures. Even if the IoT promises economic growth as well as convenience for users, the security (and safety) implications of the IoT are equally significant. In fact, weak security in IoT devices could have dangerous consequences, such as to a car crash, or an intruder entering in our home. As an example, in October 2016, the distributed denial of service attack on Dyn, a company controlling and managing several DNS services, brought down most of America's Internet, and was caused by an IoT botnet (Mirai). This is mainly due to an increasing number of vulnerabilities in IoT devices being discovered on a daily basis, and that are the consequence of poor IoT security practices. To properly address the security and testing of IoT devices, the first step is the description of a threat model. However, few IoT manufactures base their testing on sound threat modelling techniques and comprehensive IoT security guidelines.
For these reasons, in this paper we propose a methodological approach for IoT security testing, which extends the OWASP IoT framework to include threat models to guide the selection of tests used to evaluate IoT attack surfaces and associated vulnerabilities. In addition, the proposed extended framework includes indications on how to actually test a given vulnerability and a set of recommended tools for performing the tests. To this end, we have devised a set of procedures associated with the tests, e.g. accessing device hardware or resetting the device. We also describe a set of tests based on the framework we have performed on IoT devices to test their security. In particular, we have tested the framework on a home router, a relatively cheap baby monitor, and a pricey security system. The methodological testing of the devices reported that the baby monitor showed signs of inadequate security, the router patching any known vulnerabilities as expected from a well-known manufacturer, and the security system quashing any penetration testing attempts.
Keywords: Internet of Things, OWASP, Attack Surfaces, Testing Methodology

1 Introduction

As defined in [15], the Internet of Things (IoT) is "a system of interrelated computing devices, mechanical and digital machines, objects, animals, or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction”. The IoT encompasses a large range of devices (‘things’), among which every-day household electronics, such as dishwashers, fridges, smart cameras, smart watches, smart glasses, smart TVs, and smart light bulbs. Wearable devices can monitor heart rate, steps, and spent calories to name just a few ‘smart’ features introduced by IoT devices. The IoT offers almost limitless possibilities of positive features, finally fulfilling at least some of the alacritous visions of futurists in the mid to late twentieth century[1]. These positives are hard to outweigh, but negatives do exist. Concerns over data collection by product manufacturers and associated privacy, as well as security vulnerabilities in these devices, might well not be enough to completely put people off using IoT devices or care for these issues [11], but these concerns are still prominent.

Perhaps the most nefarious use of IoT devices is the one performed by botnets. One in particular, the Mirai botnet [13], exploits something as simple as default credentials in IoT devices, gaining root access to recruit infected devices to the botnet. This particular botnet made global headlines after causing massive Internet outages, mainly in America. Articles highlighted the root cause being infected IoT devices, stirring skepticism over these devices as a whole. The botnet code itself uses a list of known default credentials for different devices and brute-forces IoT devices over the Telnet protocol. This is a shockingly simple exploitation of IoT devices, yet very common nowadays. The first question is why the Telnet protocol is used so frequently, as the protocol is very well known to be insecure by transmitting data in the clear. Devices are often manufactured with trivial hard coded credentials in firmware, such as username ‘admin’ and password ‘admin’. More complex hard coded credentials still pose a problem, since they could be extracted from the device firmware. For example, the main manufacturer whose devices were targeted by the Mirai botnet simply changed their mechanism of default credentials by assigning a default username and password, stored in a table in the firmware, for each day of the year [18]. Such a fix is clearly not adequate – a method of setting user credentials upon first boot would make more sense (but this would bring other usability issues). Of course, default credentials and firmware extraction/analysis are just examples of vulnerabilities related to attack surfaces. In fact, most of IoT devices for everyday consumers are still not designed with security in mind, with a reported one out of ten devices displaying the prevalent issue of common default credentials alone [12]. Only few manufacturers have started to include security practices in IoT design and development, mainly due to the publicity around the concept of the exploitation of an IoT device potentially having a direct impact on consumers and negative press [16] [17].

For these reasons, we propose an extension to the OWASP IoT attack surface mappings [14] with the aim of making the security testing of IoT devices more rigorous as to significantly reduce their number of vulnerabilities. In particular, we have extended the OWASP framework to include: (i) a mapping of vulnerabilities to a set of security tests – to facilitate the selection of tests to be performed; (ii) a mapping of tests to potentially useful tools to perform the test – to guide users/manufacturers on

[1] https://www.postscapes.com/internet-of-things-history/

the choice of the tools to perform the test; and (iii) a more detailed threat modelling – to formalize under which assumptions and scenarios the tests are meaningful. The main aim of the proposed framework is to allow IoT designers, as well as manufactures and end-users, to model and risk-assess IoT security in a methodological and comprehensive way. To evaluate the efficacy of the proposed framework, we have used it to guide the tests on three classes of IoT devices.

The paper is structured as follows. In Section 2, we describe related works. In Section 3, we firstly briefly recall the main concepts of the IoT OWASP framework, and we then describe the proposed changes to the IoT OWASP framework, by discussing the extended surface mapping to include IoT security considerations, the tools to use and threat models. In Section 4, we report the results of the tests performed on three classes of IoT devices by following the proposed testing methodology, and we discuss our findings. Finally, in Section 5 we conclude the paper.

2 Related Works

In [1], the authors have analysed the security of Samsung's SmartThings platform and found several security vulnerabilities. Similarly, in [2] the authors have performed functionality extension attacks on IoT devices using smart-lights as a covert Li-Fi communication system to get data from a highly secure office building. The authors were able to read the leaked data from a distance of over 100 meters using cheap equipment. The authors of [3] have designed a feature-distributed malware to perform various malicious activities, such as unlocking smart-locks and disarming security alarms. These results show that traditional web attack techniques, such as cookie stealing, can be turned into sophisticated attacks on IoT devices. Similarly, the authors of [4] examine the security of five commercial home smart-locks, and show that most of these devices suffer from poor design and implementation choices. [5] analyses the IoT vulnerabilities from insecure web/mobile/cloud interfaces, by testing insufficient authentication and authorization, insecure network services, lack of transport encryption and integrity verification, privacy concerns, insufficient security configurability, insecure software/firmware, and poor physical security. In [6] the authors examine the security of different categories of IoT devices to understand their resilience under different threat models, in particular physical access and close proximity of the attacker. However, none of these papers has proposed a framework to enable IoT manufacturers to test the security of IoT devices methodically by using different tools and security assumptions [7] [8], which is the main goal of this paper.

3 Proposed Extended IoT Framework

This section involves the analysis of relevant IoT attack surfaces and vulnerabilities from IoT OWASP framework, and then it proposes an extension to include testing considerations, tools, and threat models for each vulnerability.

3.1 IoT Threat Modelling

A critical aspect of security testing is to evaluate all possible attack surfaces and their associated vulnerabilities, and then select specific attack surfaces to analyze and test. In IoT scenarios, attack surfaces are points of an IoT device at which an attacker can gain access to it to perform a security violation, e.g. by delivering a malicious payload. Vulnerabilities are the means by which an attack can be performed. As an example, **Table 1** lists some of the attack surfaces and associate vulnerabilities from the OWASP IoT Framework [14].

Table 1. IoT OWASP Framework (Excerpt)

Attack Surface	**Vulnerabilities**	
Ecosystem Access Control	• Implicit trust between components • Enrolment security	• Decommissioning system • Lost access procedures
Device Memory	• Cleartext usernames • Cleartext passwords	• Third-party credentials • Encryption keys
Device Physical Interfaces	• Firmware extraction • User CLI • Admin CLI	• Privilege escalation • Reset to insecure state • Removal of storage media
Device Web Interface	• SQL injection • Cross-site scripting • Cross-site Request Forgery • Username enumeration	• Weak passwords • Account lockout • Known default credentials

It appears evident that, while the OWASP framework includes several attack surfaces and sets of vulnerabilities, there are no indications on how to actually test these vulnerabilities, and under which security condition(s). Therefore, the first requirement (R1) of our methodological approach to testing the devices is to identify *suitable* attack surfaces, based on factors such as knowledge of the scenario (i.e., threat model), and whether the attack surface is suitable to the device being tested. For example, prematurely choosing the Device Web Interface attack surface to test would not make sense on a device with no web interface. The second requirement (R2) of our framework is to provide a set of guidelines describing how to test the vulnerabilities along with the list of tools that can be used to perform the test. The aim of these requirements is to make the framework more rigorous and to facilitate the selection of tests to be performed.

3.2 Extended Attack Surface Mapping

To consider these two requirements, the extended framework includes three additional mappings for each vulnerability, which are:

(i) *IoT Security Considerations*, i.e. guidelines on how to perform the testing;
(ii) *Methodologies and Tools*, i.e. a list of suggested tools and methodologies to perform the tests;
(iii) *Threat Models*, i.e. a description of the security assumptions, in particular in terms of attacker's access;

The first requirement (R1) is satisfied by including in the framework *IoT Security Considerations* and *Methodologies and Testing tools*. The first extension describes effectively means of testing, that is, the ways in which to test a given vulnerability if it seems possible. The second extension, *Methodologies and Tools*, refers to various pieces of software or hardware, as well as methodologies, that can be used to perform the testing process. This piece of information is useful to testers, since it works in concurrency with the Security Considerations to extend upon what will be needed to actually test the vulnerability. The second requirement (R2) is satisfied by extending the framework to include *Threat Models*. In fact, some vulnerabilities may require hardware access, and some may be exploited remotely. The extension *Threat Models* defines the security assumptions, e.g. proximity of attacker to perform the attack, so that manufactures can decide whether to perform the testing or not based on the device deployment scenarios. For instance, in case a device is not deployed in open fields, all the tests related to physical attacks could be omitted.

We have included these threat models in the extended framework:

- **Physical access**, i.e. an attacker that can tamper with the hardware;
- **Close access**, i.e. an attacker that is in close proximity of the IoT device (e.g., RFID access), without having physical access;
- **Network access**, in an attacker able to get access to the same network of the IoT device (e.g., WiFi);
- **Remote access**, i.e. an attacker able to connect remotely to the device;
- **Application access**, i.e. an attacker able to connect remotely to the application (typically running on a smartphone) that controls the IoT device;
- **Router/Hub access**, i.e. an attacker able to connect remotely to the home router or hub mediating/controlling the IoT device;
- **Cloud access**, i.e. an attacker able to connect to the Cloud backend only.

Each of these accesses is associated with a set of attacks that an attacker can perform, and a stronger assumption (e.g., physical access), which also includes the attacks associated with a weaker assumption (e.g., network access), while the vice-versa is not true. The symbols we have used to visually represent these threat models are shown in **Table 2**.

Table 2. Symbols used in the Framework to Represent the Threat Models

Physical Access	Close Access	Network Access	Remote Access	App Access	Gateway Access	Cloud Access
📹	((•))	📶	🌐	📱	📶	☁

Table 3 reports the proposed extended framework by showing a representative subset of the vulnerabilities and attack areas for which we have provided an extension, namely *Device Physical Interfaces*, *Device Web Interface*, *Device Firmware*, *Device Network Services*, and *Local Data Storage*[2].

Table 3. Proposed Extended OWASP IoT Attack Surface Areas.

Device Physical Interfaces	Device Web Interface	Device Firmware	Device Network Services	Local Data Storage

Vulnerability	IoT Security Considerations (How to Test)	Methodologies and Tools	Threat Model
Firmware extraction	Gain access to serial ports from hardware and dump firmware	(physical) UART to USB cables	📹
User/admin CLI	Use potential serial ports to access the CLI	(physical) UART to USB into computer	📹
Privilege escalation	Potential buffer overflow attacks, command injections	Metasploit framework	📹
Reset to insecure state	Reset button? Analyse network activity thereafter	Wireshark	📹
Known default credentials	Check internet for known default password lists for a given device	Browsing internet	🌐
SQL injection	Give malicious input in some form on the web interface	SQL ninja/Metasploit SQLi tools	🌐
Weak passwords	Guess passwords or bruteforce	Brutus bruteforcer	🌐
Account lockout	Test whether account locks by constantly logging in incorrectly	Brutus or manually	🌐
Username enumeration	Bruteforce default usernames until some message indicates its existence	Brutus or any Metasploit bruteforcer	🌐

[2] For the sake of conciseness, the Table shown here briefly summarizes "IoT Security Considerations" and "Methodologies and Tools".

Hardcoded credentials	Extract the firmware and manually analyse it for any credentials	Binwalk firmware extraction and analysis tool	🌐
Sensitive information disclosure	Same as above, but analyse for sensitive information instead	Binwalk firmware extraction and analysis tool	🌐
Sensitive URL disclosure	Same as above, but look for any sensitive URLs	Binwalk firmware extraction and analysis tool	🌐
Firmware version display and/or last update date	Displayed in some UI? Transmitted over network?	Wireshark	🌐
Information disclosure	Intercept packets to gain unintended information	Wireshark	🌐
Administrative command line	Check an open port (Telnet or SSH). Guess default credentials, bruteforce or extract firmware as above	Fing network service identifier to identify Telnet/SSH. Firmware tools as above	🌐
Injection	Inject code into network messages/user input, perform dot traversal	Metasploit tools	🌐
Denial of service	Flood a device with network traffic/packets/broken packets	Ettercap	🌐
Vulnerable UDP services	Sending spoofed IP UDP packets or even DoS attacks on open UDP ports	Ettercap DoS	🌐
Unencrypted services	Read unencrypted data with a packet sniffer	Wireshark	🌐
Unencrypted data	View data perhaps stored by applications accompanying the device	Dex2jar for APK to JAR	📱
Update sent without encryption	Look for streams of unencrypted packets related to an update	Wireshark	📶 🌐
No Manual update mechanism	Look for ways by which it is not possible to manually update	Web interfaces	🌐

As already recalled, by selecting a specific *Threat Model*, the proposed framework shows the vulnerabilities to test belonging to the uncovered attack surfaces. Then, the associated *IoT Security Considerations* report a set of guidelines on how to perform the testing. For example, in the *Physical Access* threat model, the *Device Physical Interfaces Attack Surface* is enabled. Here, a possible vulnerability is *Firmware Extraction*, and one of the main *Security Consideration* is that access to serial ports from hardware must be tested, and a set of *Methodologies and Tools* to obtain the firmware are described.

In the following section we describe how we have used the extended framework to assess the security of three categories of IoT devices.

4 Testing with the Extended Framework

This section covers the actual testing and describes the results obtained following the extended IoT framework. In detail, we have tested the devices on a router, on a baby monitor, and on a security system. For the sake of responsible disclosure, we omit the details of the manufacturers and IoT models of any of these devices, and any details which could lead to any harmful outcome.

A preliminary phase of the proposed methodological approach taken to the testing is that of *reconnaissance*, which involves learning about and gathering information on the target before the test. In our tests, for each of the three devices, the following pieces information were gathered: chipsets used, known vulnerabilities (to verify their existence or absence), information on how open the devices, as well as other various specifics that would aid the testing. In one case, to speed up the testing, firmware for one of the devices was obtained by emailing the manufacturer and having them send the actual binary.

4.1 Selecting Suitable Vulnerabilities from the Extended Framework

After the reconnaissance phase, vulnerabilities to be tested for each device are carefully selected based on the threat model and on whether a specific vulnerability would be suitable for that device. The list of vulnerabilities considered in our tests is shown in **Table 4**, and will be detailed in the following.

Table 4. Tested vulnerabilities for each device. **R** denotes Router, **B** denotes Baby Monitor camera, **S** denotes the Security System.

Attack Surface	**Vulnerability**	**R**	**B**	**S**
Device Network Services	Information disclosure	X	X	X
	Administrative command line	X	X	X
	Denial of Service		X	X
	Unencrypted services		X	X
Device Web Interface	Account Lockout	X		X
	Weak Passwords	X		
	SQL Injection	X		
	Default known credentials	X		
Update Mechanism	Update sent without encryption	X		
Device Physical Interface	User/Admin CLI		X	
	Firmware Extraction		X	
Device Firmware	Hardcoded credentials		X	X
	Firmware Version display	X	X	X
	Firmware Last update date	X	X	X

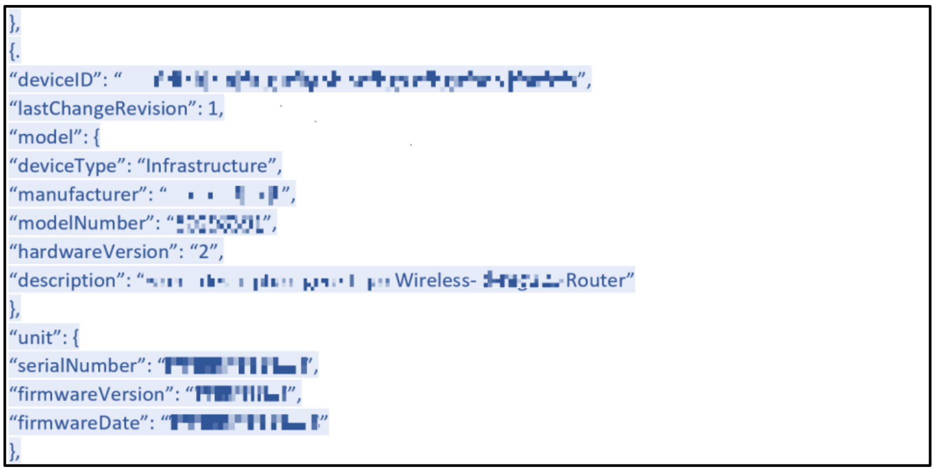
4.2 Home Router Testing

In the considered testbed, the home router is one that has previously been tested by security professionals, and a well-known reported vulnerability with this specific device is the use of default credentials, with the username and password being ‘admin’. Therefore, we first started with security testing of credentials. The expectation would be for this vulnerability to have been fixed, despite the issue still being listed on a frequently update website of default credentials. Following the list of guidelines in the extended framework, we performed the tests through the automatic testing of credentials on the web interface, and easily-guessed credentials were not being an issue anymore. We then tested possible vulnerabilities to the firmware and update mechanism. Upon actually setting up the device and connecting to the router, a prompt was made by a webpage to run a firmware update in order for the device to work. However, this was easily skipped past by closing the webpage. One could then proceed to use the device normally, potentially postponing the firmware update indefinitely, as a common user would, resulting in any possible vulnerability patches in the update being left unapplied. However, to access the router login page, the firmware had to be updated. Even so, it should not be allowed to be possible to skip a crucial firmware update and use the device. An incoming update to the router was intercepted, using Wireshark[3]. The contents were encrypted, meaning that the device is secure against the Update Sent Without Encryption vulnerability, listed under the Update Mechanism attack surface.

We then tested for an administrative command line as outlined by the Device Network Services attack surface mappings. As per the Security Considerations for the vulnerability, open service ports were searched for to potentially gain access to a command line interface (CLI). The aim is to establish an SSH or Telnet connection. To do this, Fing service scanner[4] and the Netstat[5] command was used. The router had a lot of miscellaneous ports open, but none which could potentially hold the route to a CLI. It turns out that it could actually be chosen to enable remote connection to the router on the router settings page by enabling SSH. Had there been an accessible administrative command line available over the network, this would have been a further venue of possible exploitation to test. The Web interface was tested further with malicious input into the username and password fields. Following the recommendations reported on the *Tools* column, we used SQL ninja[6] to perform an SQL injection, which did not return any successful result. We discovered that the router obtained credentials from a cloud service and that the manufacturer has taken measures against SQL injections – noted as the most important web application security risk [15]. By this same method of testing suggested by the framework, we also discovered that the web interface employs account lockout. In addition, weak (default) passwords could not be tested for, since the user has to set the password on the admin page. The final test was for Information Disclosure vulnerability. This test returned positive as some

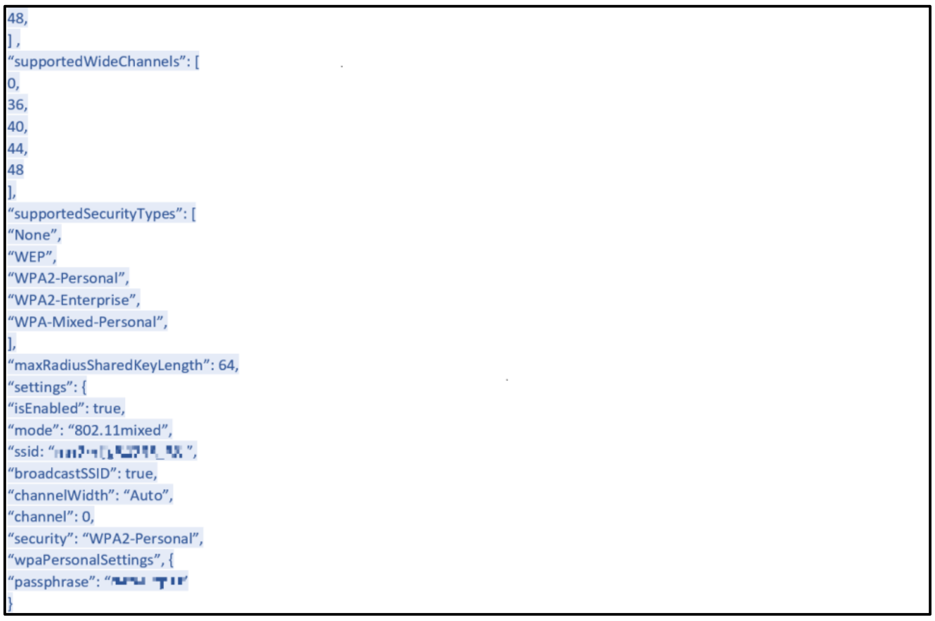
[3] https://www.wireshark.org/
[4] https://www.fing.io/
[5] https://linux.die.net/man/8/netstat
[6] http://sqlninja.sourceforge.net/

sensitive information was disclosed. In particular, by using Wireshark[7] to analyze packets, upon loading the device web interface, it was observed that details of the device were sent over HTTP in the clear (see **Fig. 1**). This includes a display of the current firmware version and its last update, as well as the serial number. This information is only sent when loading the web interface.



```
},
{.
"deviceID": " [redacted]",
"lastChangeRevision": 1,
"model": {
"deviceType": "Infrastructure",
"manufacturer": " [redacted]",
"modelNumber": "[redacted]",
"hardwareVersion": "2",
"description": "[redacted] Wireless- [redacted] Router"
},
"unit": {
"serialNumber": "[redacted]",
"firmwareVersion": "[redacted]",
"firmwareDate": "[redacted]"
},
```

Fig. 1. Cleartext Device Details



```
48,
] ,
"supportedWideChannels": [
0,
36,
40,
44,
48
],
"supportedSecurityTypes": [
"None",
"WEP",
"WPA2-Personal",
"WPA2-Enterprise",
"WPA-Mixed-Personal",
],
"maxRadiusSharedKeyLength": 64,
"settings": {
"isEnabled": true,
"mode": "802.11mixed",
"ssid: "[redacted] ",
"broadcastSSID": true,
"channelWidth": "Auto",
"channel": 0,
"security": "WPA2-Personal",
"wpaPersonalSettings", {
"passphrase": "[redacted]
}
```

Fig. 2. Cleartext Credentials Details

[7] https://www.wireshark.org/

It is a more threating outcome when the user actually logs into the web interface as all user information and device information, including the device's password, is transmitted in clear text (see **Fig. 2**). Hence, adversaries could simply just read the HTTP packet and have access to sensitive information, such as password – which can also be reused across other accounts of the same user.

4.3 Baby Monitor Testing

The Baby monitor we have tested is a low-cost baby monitor, designed to send a video stream to an application on a mobile phone with the camera registered to it. It does not have a web link available to view the video, but only a mobile application.

The entire initial setup phase of the device and its controlling application had secure network communications, which involved user details input to the application, entering wireless network details, and configuring the two devices such that the mobile application displayed the camera's live video feed. Any information disclosure here and subsequently through any future communications was not evident during the testing. All communications were encrypted, and no details were transmitted in the clear. To test possible vulnerabilities here, following the framework guidelines, Wireshark was used as well as an Android application titled 'Packet Capture'[8], which records packets in and out of a smartphone for a set period of time. Packets were then individually analyzed, in particular around 90 packets during the camera initialization and 20-50 for each following analysis of general camera usage.

As per the extended framework, Fing network service scanner[4] was then used to scan for any open services with the hope of finding a service that would offer access to some administrative CLI. One service detected of particular interest was Telnet, which is dangerous, since it transmits data in clear-text. The camera had an open Telnet port indeed, requiring credentials to connect to, and so a dictionary attack could be performed. It seemed appropriate to gather some further understanding of the device to do this. Since the camera's firmware was not readily available anywhere on the Internet, we followed the framework guidelines on how to exploit the APK of the camera's mobile application. In fact, the APK includes the compiled version of the Java source code for a mobile application, which still embeds some strings from the original source code. To this end, as per the framework guidelines, we have analyzed the APK with automatic tools to search for words such as 'default', 'Telnet', 'pass', 'user', as well as any variations of these words including capital letters. There were lots of instances of these words in the code but after following the program control flow of these instances, we discovered that no code referred to the Telnet login. There was an imported class used, 'Telnet login', and a default Apache library class but both classes were not actually used in the product. One interesting finding, however, was that the actual camera was manufactured by one of manufacturers that produced several vulnerable devices targeted by the Mirai botnet for default credentials. The framework includes guidelines on to perform a dictionary attack, which we did on the Telnet login of the camera with the Mirai word list, as well as any other common

[8] https://play.google.com/store/apps/details?id=app.greyshirts.sslcapture

logins by using some MSFconsole[9] commands. However, the tests were unsuccessful and username enumeration was not possible. The connection was also dropped after several attempts and 60 seconds. Similarly, brute-forcing the Telnet port would take a very long time without meaningful results.

We then considered the security of the device's firmware. The firmware version and update-date was clearly listed on the application. To access the firmware by referencing our framework, it made sense to pursue the Device Physical Interface Attack Surface. IoT Security considerations were taken into account for the Firmware Extraction vulnerability, and it was quickly learned that hardware would need to be dismantled to access useful serial ports. Additionally, as per the recommended Tools to test for the vulnerability, UART to USB cables were required. After emailing the manufacturer and obtaining the cables, it was learned that, to access the camera's hardware, the device would need to be broken open with the risk of damaging the hardware to the point of it not working again. The device was opened, with the goal of finding a UART serial port (see **Fig. 3**), which might have offered access to an administrative CLI as well as the firmware for the device if connected to.

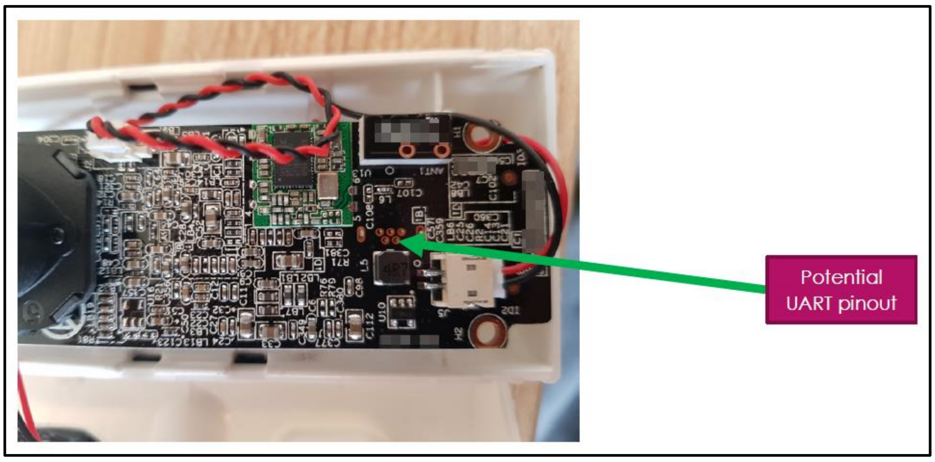


Fig. 3. Potential UART pinout on Baby Monitor camera

Upon analysis of the hardware, we found a candidate for a UART pinout, which we tested with a voltmeter to read the voltages given off by each port, so that could be detected whether it was a UART pinout based on the voltage given off by each port and their voltages relative to each-other. The voltages were far too close together to be UART pins – it was expected that there would be a fixed amount of voltage between them, relative to each-other but this was not evident.

The final test on this device was to understand its resilience against DoS attacks. After referencing the framework, attempts were made to take down the camera by flooding it with a stream of packets, as per the Security Considerations for the Denial

[9] https://www.offensive-security.com/metasploit-unleashed/msfconsole/

of Service vulnerability. To do this, Ettercap[10], a program shipped with Kali Linux[11], was used as listed in the extended framework. By specifying the IP address and the type of attack to carry out, Ettercap proceeded to transmit to the device an overwhelming number packets. The device physically heated up and its light flickered red indicating an error. The video stream then slowly flickered out through packet loss and the camera turned itself off. Hence, the DoS attack was performed successfully.

4.4 Security System Testing

The final IoT device that we tested is a reputable security system, complete with motion detection, Cloud-stored video recordings, two-way camera communication and support for multiple cameras. The camera streams to a website which requires a user login to connect to the camera. The device web interface was regarded as secure: SQL injections failed and account lockouts were enabled. Additionally, weak passwords were not permitted for the device, and all the efforts at analyzing the web interface were unsuccessful. Regarding the Device Network Services Attack Surface, any network communications were encrypted, and so no information was retrieved. Similarly, there were no publicly accessible ports on the device. Our preliminary tests showed that this device was designed with security in mind, considering the selected attack surfaces-vulnerabilities pair.

4.5 Results of the Testing

The results of the tests are shown in **Table 5**, which reports, for each device, the vulnerabilities tested, and whether or not the vulnerability was realized. Here, ‘Y’ denotes realization, ‘N’ denotes that the vulnerability was not realized, ‘-’ denotes untested, and ‘?’ indicates there could potentially be an issue with more tests.

Table 5. Results of the Tests using the Proposed Framework.

Device	**Vulnerability**	**Vulnerability exploited?**
Router	Information disclosure	**Y**
	Account lockout	N
	Weak passwords	N
	SQL injection	N
	Default known credentials	N
	Administrative command line	N
	Update sent without encryption	-
	Firmware version display/update date	**Y**
Baby Monitor	Information disclosure	N
	Administrative command line	?
	Denial of service	**Y**
	Unencrypted services	N
	Firmware extraction	N
	User/admin CLI	?

[10] https://www.ettercap-project.org/
[11] https://www.kali.org/

	Hardcoded credentials	?
	Firmware version display/update date	**Y**
Secure System	Information disclosure	N
	Administrative command line	?
	Denial of services	-
	Hardcoded credentials	N
	Unencrypted services	N
	Firmware version display / last update date	-
	Account lockout	N
	Weak passwords	N
	SQL injection	N
	Known default credentials	N

By following the framework guidelines, we discovered that the results of the tests were different than expected. For example, it was not expected that the router would have basic security issues such as instances of clear text communication. Additionally, it was not expected that the baby monitor camera would be as secure as it was, despite few existing issues. We expected to find some information disclosure or problems with network communications at least, but there was nothing in that sense. An expected result, however, was the security system device's firm security against the attack vectors chosen. Therefore, it appeared that the device is properly designed from a security perspective – at least, considering the five selected attack surfaces.

5 Conclusion

IoT security is still an issue today, in particular as several insecure devices are still mass produced, with little attention from manufacturers on device security. However, with IoT regulation groups and boards aimed at ensuring that security standards are kept – e.g. the EU IoT Council[12] – we probably should see a swing in a positive direction for the security of IoT devices. Existing frameworks to test IoT devices simply are not mature enough for testers. There definitely needs to be an extension to it both in terms of what is included, e.g. the extension of tools/attack scope, as well as more vulnerabilities being added, such as relay/replay attacks. IoT security frameworks also need constant updating: for example, the OWASP attack surfaces mapping page was last updated in 2015, which is not in line with how IoT has developed since then.

This paper shows that the proposed extension to the OWASP attack surface to vulnerabilities mappings is useful from a testing perspective. In particular, the testing methodology we have proposed adds further structure to the process of identifying and exploiting any vulnerability in IoT devices and would be useful to actually add to the OWASP IoT attack surface mappings. A future extension to our proposed framework, which we are currently working on, is the addendum of extended mappings for the remaining attack surfaces listed in the original OWASP framework.

[12] https://www.theinternetofthings.eu/

Acknowledgment.
This work was partially supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 779391 (FutureTPM).

References

1. Fernandes, Earlence, Jaeyeon Jung, and Atul Prakash. "Security analysis of emerging smart home applications." In 2016 IEEE Symposium on Security and Privacy (SP), pp. 636-654. IEEE, 2016.
2. Ronen, Eyal, and Adi Shamir. "Extended functionality attacks on IoT devices: The case of smart lights." In Security and Privacy (EuroS&P), 2016 IEEE European Symposium on, pp. 3-12. IEEE, 2016.
3. Min, Byungho, and Vijay Varadharajan. "Design and evaluation of feature distributed malware attacks against the Internet of Things (IoT)." In Engineering of Complex Computer Systems (ICECCS), 2015 20th International Conference on, pp. 80-89. IEEE, 2015.
4. Ho, Grant, Derek Leung, Pratyush Mishra, Ashkan Hosseini, Dawn Song, and David Wagner. "Smart locks: Lessons for securing commodity internet of things devices." In Proceedings of the 11th ACM on Asia conference on computer and communications security, pp. 461-472. ACM, 2016.
5. Bertino, Elisa, and Nayeem Islam. "Botnets and internet of things security." Computer 2 (2017): 76-79.
6. Xu, He, Daniele Sgandurra, Keith Mayes, Peng Li, and Ruchuan Wang. "Analysing the Resilience of the Internet of Things Against Physical and Proximity Attacks." In International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage, pp. 291-301. Springer, Cham, 2017.
7. Sgandurra, Daniele, and Emil Lupu. "Evolution of attacks, threat models, and solutions for virtualized systems." ACM Computing Surveys (CSUR) 48, no. 3 (2016): 46.
8. Sgandurra, Daniele, Erisa Karafili, and Emil Lupu. "Formalizing Threat Models for Virtualized Systems." In IFIP Annual Conference on Data and Applications Security and Privacy, pp. 251-267. Springer, Cham, 2016.
9. Rouse, Margaret: "Prevent Enterprise IoT Security Challenges". (2018). Available at: http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT
10. Dave, E. "How the next evolution of the Internet is changing everything." The Internet of Things (2011). Available at: https://www.cisco.com/c/dam/en_us/about/ac79/docs/inov/IoT_IBSG_0411FINAL.pdf
11. Rouffineau, Thibaut: “Consumers are terrible at updating their connected devices.” (2016). Available at: https://blog.ubuntu.com/2016/12/15/research-consumers-are-terrible-at-updating-their-connected-devices
12. Shipulin, Kirill, Positive Technologies: “Practical ways to misuse a router.” (2017). Available at: http://blog.ptsecurity.com/2017/06/practical-ways-to-misuse-router.html
13. Antonakakis, Manos, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric et al. "Understanding the mirai botnet." In *USENIX Security Symposium*, pp. 1092-1110. 2017.
14. OWASP, “IoT Attack Surface Areas.” (2015). Available at: https://www.owasp.org/index.php/IoT_Attack_Surface_areas

15. OWASP, “Top 10 2017: The Ten Most Critical Web Application Security Risks.” *Sl: The OWASP Foundation* (2013).
16. Trendall, Sam, PublicTechnology.net: “Labour MP: If a device is called ‘smart’ – don’t buy it.” (2018). Available at: https://publictechnology.net/articles/news/labour-mp-if-device-called-%E2%80%98smart%E2%80%99-%E2%80%93-don%E2%80%99t-buy-it
17. Ranger, Steve, ZDNet: “Internet of Things: Finding a way out of the security nightmare.” (2016). Available at: https://www.zdnet.com/article/internet-of-things-finding-a-way-out-of-the-security-nightmare/
18. Paul, The Security Ledger: “Mirai Redux: A Year’s Worth of DVR Passwords Published Online.” (2017). Available at: https://securityledger.com/2017/01/mirai-redux-a-years-worth-of-dvr-passwords-published-online/