Final Report

Internet of Things

Security Penetration Testing

CM3203 - One Semester Individual Project - 40 Credits

School of Computer Science and Informatics

Cardiff University

2021

Author: Maria Carolina Roberts

Supervisor: Dr Yulia Cherdantseva

Moderator: Irena Spasic

Abstract

The Internet of Things (IoT) is increasingly becoming incorporated into everyday life and we could argue that this area of computing is changing our lives for the better. We now have technology that is able to aid us in our everyday, sometimes hectic, lifestyle and make it easier for us to go about our day. However, not many people may know that with this new technology comes its risks and security issues. This project aimed to identify just some of the vulnerabilities that are apparent in IoT devices with the aim of better informing the public about the dangers of IoT weaknesses. Results revealed that issues identified in the IoT device were able to be exploited using a series of tools and were able to cause disruption and possible data breaches. Suggestions for increasing the security of users' devices against these attacks are discussed when evaluating the project.

Acknowledgements

There are many people who have helped me along the way when completing my project journey and I would like to take a moment to thank them.

I would like to give a sincere thanks to Dr Yulia Cherdantseva who has given me a great deal of support, guidance and feedback while completing this project.

In addition, I would like to thank George Theodorakopoulos for his initial ideas on the project title which gave me a strong starting point for the project.

Finally, I would like to give a special thanks to my friends, classmates and family for their constant encouragement and advice.

Table of Contents

List of Figures

List of Tables

Introduction

Aim of the Project

The aim of the project was to conduct an attack on an Internet of Things device in order to find its vulnerabilities and be able to suggest some countermeasures to keep these kinds of devices safe in the future. I have taken the role of a penetration tester who has access to the network, and I have conducted a series of attacks on an IoT device. This process involved me researching previous attacks on IoT devices, stating my approach for the attack, mapping out vulnerabilities that I have identified regarding the devices' known architecture, implementing these attacks based on vulnerabilities found and evaluating my rate of success after I have conducted my research. As well as the main aim, an additional aim of this project was to be able to give an opportunity for students who are interested in cyber security to learn while reading the implementation. For this I have provided a step-by-step guide of my attacks and have also written an environment set up guide in the additional documents of this project.

Scope of the Project

This project will be delivering a step-by-step implementation of an attack on a Belkin WeMo Wi-Fi Smart Plug. Topics covered include APK analysis, locking out users from their smart device account, taking control of a user's smart device, Port Scanning, Deauthentication Attacks, ARP Spoofing and SYN Flood Attacks, which I found are popular attacks amongst IoT devices. I have used VirtualBox as the virtual machine to conduct the attacks and have used Kali Linux as it provides a great range of tools that allowed me to conduct the implementation. Acquiring the firmware of the device was deemed in scope at the beginning of the project, however, since a firmware binary download was not available online, acquiring the firmware via dumping it directly from the device was deemed out of scope.

Structure of the Report

The report is sectioned off into different categories which includes: background research of the Internet of Things, a section specifying the approach that I took to complete the project, implementation of the attacks with a guide on how to conduct each one on Kali Linux if relevant, results and evaluation of my implementation with countermeasures on how to protect the device against the attacks identified and exploited, a section specifying future work that I would like to conduct in order to improve on the findings in this project, an overall conclusion on the project and, finally, a reflection on my learning while completing the project.

Background

Internet of Things

The term ‘‘Internet-of-Things’’ (IoT) is used as an umbrella keyword for covering different aspects of which are related to the extension of the Internet and the Web into real life objects [1]. As of January 2021, there were 4.66 billion active internet users worldwide - 59.5% of the global population. Out of these active users, 92.6% accessed the internet using mobile. [2] Kevin Ashton was who first proposed the concept of IoT in 1999, and he referred to IoT as “uniquely identifiable interoperable connected objects with radio-frequency identification (RFID) technology”. However, the exact definition of IoT is still in the process of forming. There are a number of technologies are involved in IoT, such as wireless sensor networks, intelligent sensing, Radio-Frequency Identification, Near-Field Communication, low energy wireless communications, cloud computing, etc, and has been developed in many ways such as in: smart cities, environmental monitoring, and smart homes and buildings. [3]

The Internet of Things is key in the digital world of connected life. It has a futuristic appeal and can make life easier and more enjoyable for people in a hectic day-to-day routine. For example, the idea of refrigerators monitoring its contents and sending orders directly to the supermarket when something is close to running out or ordering your next meal from the comfort of your bed with a voice or gesture command to intelligent assistants such as Amazon Alexa, Siri or Google Assistant is very appealing to people. With the creation and wide use of smartphones, smart TV, and more smart devices like Amazon Echo and Google Home, these ideas are not just science fiction but are becoming a reality. [4]

In some cases, IoT manufacturers have struggled to implement a security system that is able to keep out attackers and security experts have warned of the potential risk of the large numbers of unsecured IoT devices. In December 2013, a researcher at Proofpoint, an enterprise security firm, discovered the first IoT botnet. They found that more than 25% of the botnet was made up of devices other than computers, including smart TVs, baby monitors, and other household smart appliances. Another example is on October 21, 2016, where many websites including: Netflix, Twitter, Spotify, Reddit and SoundCloud were reported to be inaccessible by users and this was caused by a distributed denial of service attack (DDoS) attack which used a network of user devices from the IoT. [5]

Internet of Things Protocols

IoT devices tend to connect and interact with each other using wireless Radio Frequency (RF) communication and there are many wireless frequencies and protocols which are used in many devices nowadays. Some of the common protocols which are used by IoT devices are Wi-Fi (802.11), ZigBee (802.15.4), Z-Wave, Bluetooth (802.15.1), and Bluetooth Low Energy. [16]

Transport Layer

Transport layer protocols, Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), communicate with each other with the use of port numbers and use this communication to identify applications.

Transmission Control Protocol (TCP)

TCP works with the Internet Protocol (IP), which defines how devices will send packets of data and information to each other. It is connection-oriented which means a connection is maintained until the applications at each end have completed their message exchange. This exchange involves a three-way handshake, where the host who wants to be connected will send an SYN to the target host, the receiver acknowledges it by sending back an SYN, ACK message to the sender and then the sender device will send another ACK message back to the receiver. It is used in order to break application data into packets which networks can deliver, send to and accept packets from the network layer, manage flow control and handle retransmission of dropped packets and acknowledge all packets that arrive for error free communication.

User Datagram Protocol (UDP)

This protocol is primarily used for low-latency and loss-tolerating connections between applications. It is known as connectionless-oriented, and it enables transfer of data before an agreement is provided by the receiver which speeds up communications. This makes it faster than TCP but does not guarantee delivery.

Application Layer

Application layer protocols define how clients and servers, running on different end systems, will pass messages to each other. This passing of messages comes in the form of request messages and response messages.

HTTP

HTTP is used for presenting information that is used for transferring data over a network. There are two main kinds of HTTP messages: requests and responses. HTTP data is sent in plaintext which means all requests and responses can be read by anyone who is monitoring the session. An attacker would be able to read the text in the request and the response and know exactly what information a system is asking for, sending, or receiving.

HTTPS

HTTPS uses TLS (or SSL) in order to encrypt these HTTP requests and responses. TLS involves public key encryption where there is a public key and a private key. The public key will be shared with one device using the server's SSL certificate. When the device would like to establish a connection with a server, the public and private keys are used by both of them to agree on new keys, known as session keys in order to encrypt their communication. This, therefore, makes it harder for attackers to eavesdrop if they are monitoring the session.

SOAP

SOAP (Simple Object Access Protocol) is a messaging protocol specification for exchanging structured information in the implementation of web services in computer networks. It uses XML Information Set for its message format, and relies on application layer protocols, most often HTTP.

UPnP

UPnP (Universal Plug and Play) permits networked devices to discover each other's presence on the network and establish functional network services for data sharing and communication.

Wi-Fi

802.11 b/g/n

802.11: Specifies the set of MAC and Physical Layer protocols in order to achieve Wireless Local Area Network communication.

802.11b: 802.11 was expanded and the 802.11b specification was formed. It supports bandwidth up to 11 Mbps. It uses the same unregulated radio signalling frequency (2.4 GHz) as 802.11. Some positives of 802.11b include: low cost, signal is good and is not easily obstructed. Some negatives of 802.11b include: slow maximum speed and home appliances may interfere on the unregulated frequency band.

802.11g: 802.11g supports bandwidth up to 54 Mbps and utilises the 2.4 GHz frequency. 802.11g is backwards compatible with 802. Some positives of 802.11g include: fast maximum speed, good signal range and not easily obstructed. Some negatives of 802.11g include: more expensive than 802.11b and appliances may interfere on the unregulated signal frequency.

802.11n: 802.11n utilizes multiple wireless signals and antennas (called MIMO technology) instead of one and introduced specifications providing for up to 300 Mbps of network bandwidth. It is also backward compatible with 802.11b and 802.11g. Some positives of 802.11n include: has the fastest maximum speed, best signal range and is more resistant to signal interference from outside sources. Some negatives of 802.11n include: more expensive than 802.11g and the use of multiple signals may greatly interfere with nearby 802.11b/g-based networks. [13]

Issues Surrounding the Internet of Things

Security and privacy remain huge issues for IoT devices, which introduces a lot of privacy issues for users. Sicari et al. presented research challenges and the current solutions in the field of IoT security focusing on the main security issues which were identified in eight categories: 1) authentication; 2) access control; 3) confidentiality; 4) privacy; 5) trust; 6) secure middleware; 7) mobile security; and 8) policy enforcement. [6]

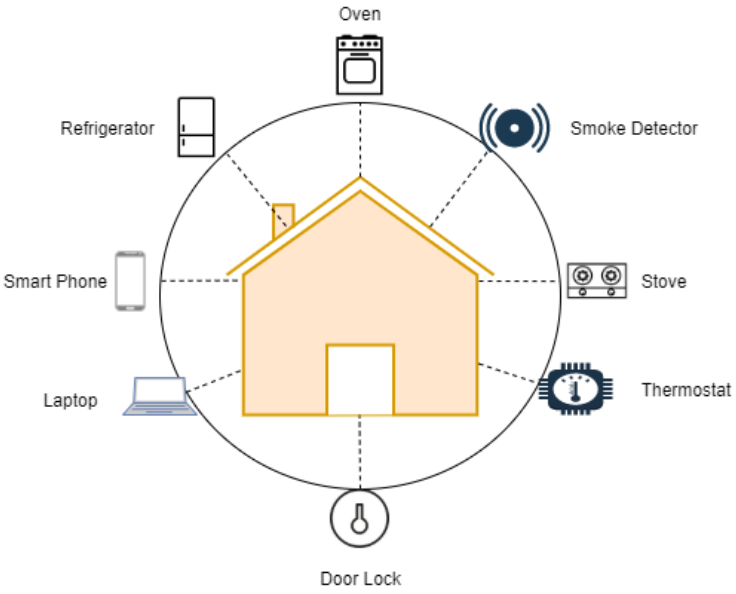


Figure 1: Smart Home with Inter-Linked Devices

Here we can consider a smart home (Figure 1), where all the devices in the home are linked via a router and can be controlled remotely by smartphone or computer. In this example, an experienced attacker may be able to use one of the devices in order to get access to the home door lock as since they are all linked together. This is just a simple example of how IoT vulnerabilities can affect an individual and make them susceptible to attacks, however, there are many other examples of how hackers can go deeper and potentially cause a lot of disruption which will be talked about later. There are many main factors which contribute to hackers being able to access the IoT devices.

IoT Top 10

The Open Web Application Security Project (OWASP) is an online community that produces freely available articles, methodologies, documentation, tools, and technologies in the field of web application security. One of their methodologies include the OWASP Internet of Things Project which is designed to help manufacturers, developers, and consumers to better understand the security issues associated with IoT. According to the OWASP, some weaknesses of IoT devices can be categorised as below:

- **Weak, guessable or hardcoded passwords:** This is the use of easily brute forced, publicly available, or unchangeable credentials, including backdoors in the firmware or client software of the device that grants unauthorized access to the deployed systems.

- **Insecure Network Services:** This is when unneeded or insecure network services running on the device itself, especially those exposed to the internet, compromise the confidentiality, integrity/authenticity, or availability of information or allow unauthorized remote control.

- **Insecure Ecosystem Interfaces:** This is when insecure web, backend API, cloud, or mobile interfaces in the ecosystem outside of the device that allows compromise of the device or its related components. Common issues include a lack of authentication/authorization, lacking or weak encryption, and a lack of input and output filtering.

- **Lack of Secure Update Mechanism:** This is the lack of ability to securely update the device. This includes lack of firmware validation on devices, lack of secure delivery (un-encrypted in transit), lack of anti-rollback mechanisms, and lack of notifications of security changes due to updates.

- **Use of Insecure or Outdated Components:** This is the use of deprecated or insecure software components or libraries that could allow the device to be compromised. This includes insecure customization of operating system platforms, and the use of third-party software or hardware components from a compromised supply chain.

- I**nsufficient Privacy Protection:** This is when the user's personal information is stored on the device or in the ecosystem that is used insecurely, improperly, or without permission.

- **Insecure Data Transfer and Storage:** This is where the lack of encryption or access control of sensitive data anywhere within the ecosystem, including at rest, in transit, or during processing.

- **Lack of Device Management:** This is where the lack of security support on devices deployed in production, including asset management, update management, secure decommissioning, systems monitoring, and response capabilities.

- **Insecure Default Settings:** This is where devices or systems shipped with insecure default settings or lack the ability to make the system more secure by restricting operators from modifying configurations.

- **Lack of Physical Hardening:** This is where there is a lack of physical hardening measures, allowing potential attackers to gain sensitive information that can help in a future remote attack or take local control of the device. [7]

Security Challenges in IoT Architecture

Just like its definition, there is not a universally agreed upon architecture for IoT, however, a widely used format states the general layers of: Perception Layer, Middleware Layer, Network Layer and Application Layer for IoT.

Perception Layer

This layer is in charge of exchanging information for devices (E.g., Zigbee, Sensors, RFID frameworks, GPS). Challenges in security come from things such as embedded sensors in the perception layer and hackers may want to substitute their own codes into the device software. Denial of Service (DoS) attacks, malicious data, tampering, etc. are some of the most common attacks that may occur in the physical layer [8]

Middleware Layer

This layer is where mass data processing and decision making. It produces a very large amount of data which may sometimes be hard to manage which means there is a necessity to filter out malicious data and also gather correct or non-malicious data which becomes hard to do. Attackers will exploit this and may be able to replace data with malicious information and can find out lists of correct data and network information. Therefore, they are able to send invalid or malicious information to the network, leading to a shutdown of the network. [8]

Network Layer

This layer, again, carries a large amount of information which then makes it a target for hackers. Authentication and integrity of the data becomes the main issue at this layer. This layer may be susceptible to replay attacks (where an intruder copies a fragment or key of sent messages), DDoS attacks (flooding the network making it inaccessible), man in the middle (where an intruder eavesdrops communications) and malicious code injections (injecting nodes with malicious code resulting in a control of the network). [8]

Application Layer

This layer includes smart devices which can be very susceptible to attacks depending on the device. Attackers may replace program codes with bugs to hack the device and since the application layer oversees data sharing, it becomes a concern for access control, privacy and leakage of information. Some common attacks in the application layer may include exploiting software vulnerabilities and inhibiting security patches. [8]

Previous Attacks

The Mirai botnet

In the past there have been many attacks on IoT devices using botnets. Botnets are apparent in DDoS attacks where an attacker will temporarily enslave several internet-enabled devices into an arrangement known as botnet [4] and will succeed in overwhelming the network server so that victim users will not be able to access their devices or applications. The Mirai botnet is a very famous example, and it was composed primarily of embedded and IoT devices. In 2016, it overwhelmed and attacked high-profile targets with large impact distributed denial-of-service attacks and this, along with many factors, was largely due to many IoT devices, such as home routers, being installed and rarely patched or updated along with people leaving default credentials on their devices.

Belkin WeMo Smart Plug

McAfee was able to uncover a buffer overflow flaw in the WeMo Insight Smart Plug which may allow an attacker to run their own code on the device and use it to access and attack other devices on the same network as the smart plug. [10] Belkin, therefore, introduced a “GNU Privacy Guard (GPG)-based encrypted firmware distribution mechanism” which was used to stop dangerous firmware injection attacks. This, however, was easy for experienced hackers to overcome as the device was distributing the firmware signing key along with the firmware during the update process over an unencrypted channel. Belkin was also found to be prone to other security issues including bugs like SQL injection and modification of device names to execute arbitrary JavaScript on the user’s smartphone (Android). [11]

Smart Locks

A researcher with the username “Jmaxxset” found security vulnerabilities in the “August Smart Lock” which is a very popular device and was said to be one of the safest smart locks out there. These locks were used by people for their homes or businesses such as Airbnb’s so that guests could check themselves in and out. He found security vulnerabilities such as: guests having the ability to turn themselves into an admin by modifying values in network traffic from user to superuser, firmware not being signed, app functionality to get past Secure Sockets Layer (SSL) pinning (which enabled debug mode), and many more issues. [11]

The Nest Thermostat

Jason Doyle states in “Chapter 1 Internet of Things: A Primer 5” a dangerous vulnerability in Nest products that involved sending a custom-crafted value in the Wi-Fi SSID details via Bluetooth to cause a crash of the device leading to a reboot making it easy for burglars to break into the users’ home during the reboot (a time of 90 seconds) without being caught on the Nest Security Camera. Grant Hernandez, Orlando Arias, Daniel Buentello, and Yier Jin also mention some security vulnerabilities of the Nest Thermostat. Installation of a new malicious firmware on the device was achievable by pressing the button on Nest for 10 seconds to trigger the global reset. At this stage, the device could be made to look for USB media for firmware by communicating with the

sys_boot5 pin. On the USB device, a malicious firmware was present, which the device then used while booting. [11]

Related Work

Zvelo have created a technology for device profiling and threat detection. Using AI-based Behavioural Anomaly Detection it leverages knowledge of botnets, malware and phishing sites, command & control servers, and other identified weakness sources which allow their customers to be alerted to any of their vulnerable devices and informed on the situation. [17]

An interesting piece of related work is that of Peshraw Abdalla who found significant vulnerabilities in the security components of Intelligent Onvif YY HD, which is a wireless IP camera. He found many different types of vulnerabilities including the occurrence of default credentials, weak device identifier default number which made the device easy to find by attackers, sensitive information transferred without using encryption methods and a vulnerable android application which had a lot of weaknesses. This is particularly dangerous as it shows how easy it is to find vulnerable IoT devices and hack them with very little knowledge, showing that they should be investigated further to warn people of the dangers and raise awareness. [18]

Telefónica have recently created a new unit which comprises cloud, security and IoT/Big Data businesses. This had led to the creation of their own IoT Threat Detection solution which is used to detect anomalies in traffic to and from IoT devices and provides an “agentless solution”. It allows protection from layer 3 to 7 of the 7-layer OSI Model and detection techniques include, but are not limited to, automatic learning, heuristics and signatures, together with the specific intelligence of IoT. [19]

Approach

Penetration Testing

IoT penetration testing involves the theory of ethical hacking in order to assess security vulnerabilities in devices and help give suggestions to make them more secure in the future. For my investigation I have be followed a typical penetration testing methodology involving undertaking the scope, attack surface mapping, vulnerability assessment and exploitation and finally documentation and reporting.

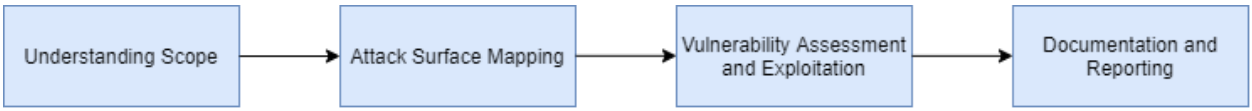


Figure 2: Penetration testing methodology

Legislation

It is important to do your own due diligence before conducting a penetration test in order to ensure no laws are being broken in the process. In the UK, for most tests, these laws include the following:

- UK Computer Misuse Act 1990
- UK Data Protection Act 1998
- Human Rights Act 1998
- Police and Justice Act 2006 [12]

In order to conduct my penetration test, I had to get consent from the individuals whose network I am on and, in this case, it was my own therefore I asked the individuals whom I live with to give consent to my practices. I have also informed them that my testing period will be from the hours of 7am to 11pm until the 14th of May and I had access to all credentials required to perform my tasks.

Penetration Testing Approaches

Testing may occur by the manufacturer, third party consulting firms, security teams, security researchers and these parties may be given different ranges of information to start with when they are performing the assessment. In relation to IoT devices, sometimes this can include the entire IoT system as well as infrastructure or it may just contain a subset of the IoT system as, in occasions, this can be cheaper or may involve less experience.

Black Box

In the black box approach, the assessments are carried out at a low cost. For this approach, the investigator has no prior knowledge of the infrastructure, the device or the technology involved. The penetration tester will follow the approach of an attacker with little experience. The penetration tester may be from third-party consulting firms, a security researcher etc. [16]

White Box

In the white box approach, the investigation will involve sharing full access to system and network information with the penetration tester. This approach is often the one that gives the best results, with the amount of knowledge the penetration tester knows having a correlation with better results they find. This approach tends to be most costly, however, ensures an in-depth investigation of the device's security. [16]

Grey Box

In the grey box approach, the tester has limited or partial knowledge about the system and network. Most of the time they have more knowledge than individuals at the organisation they are working for. This can often take the form of login credentials. This approach can strike a balance between depth and efficiency and can show what someone can do when being connected to the network. [16]

For my investigation, although I relate in some aspect to all the approaches, I think that the most similar approach to my situation is the “White Box” approach as I have full access to the systems and devices that I have conducted the tests on.

Penetration Test Network Information

Home Network Design

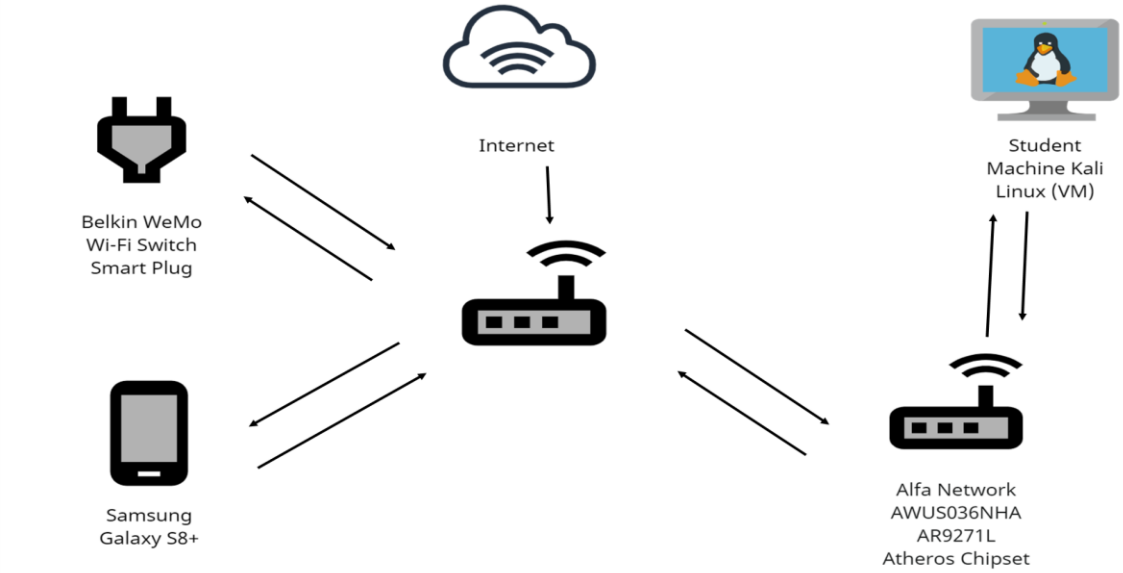


Figure 3: Home Network Design

Sniffing Network Design

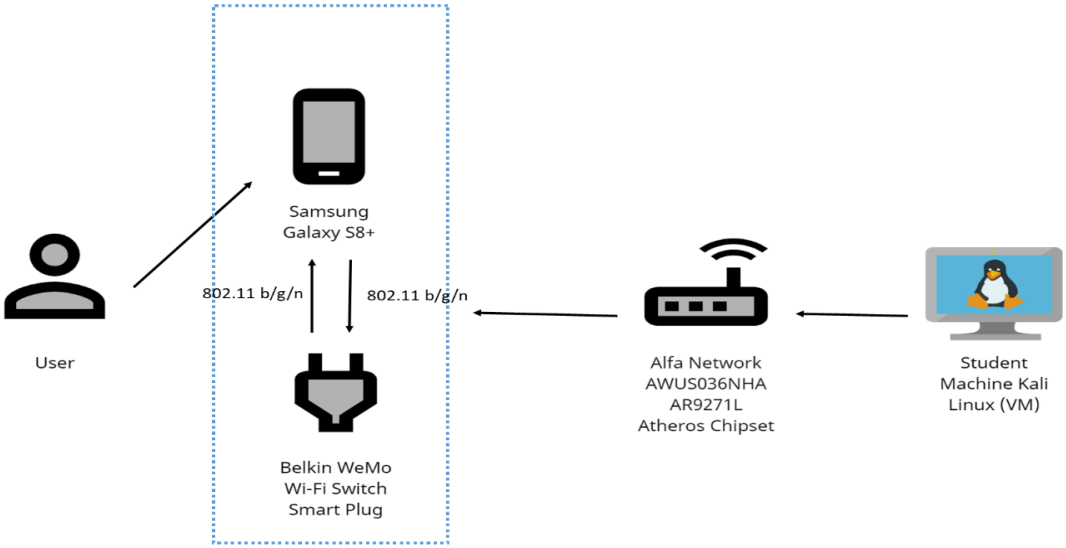


Figure 4: Penetration Testing Network Design

Network Design Features:

- Belkin WeMo Wi-Fi Switch Smart Plug F7C027
- Samsung Galaxy S8+ (Android)
- Student Machine (Laptop) running Kali Linux Virtual Machine
- Sky Home Router
- Alfa Network AWUS036NHA AR9271L Atheros Chipset (network adapter)
- 802.11 b/g/n protocol
- Star Topology

I have provided a detailed set up guide of the Virtual Machine and Atheros adapter in the additional information of my individual project submission.

Architecture Description

Samsung Galaxy S8+

I have used the Samsung Galaxy S8+ in order to access the WeMo App (Android). This will be used to communicate with the WeMo Smart Plug via Wi-Fi. The device works with Amazon Alexa and Google Assistant.

Belkin WeMo Wi-Fi Smart Plug

The WeMo Smart Plug supports Wi-Fi: 2.4GHz 802.11N 1x1, Access Point (AP) and Access Point Client (APCli) Modes. The Security Modes Supported are WEP, WPA, WPA2. It needs, in my case, an Android running v2.3 or higher and a Wireless Router (2.4GHz, 802.11, G or N compatible) with Broadband Internet connection - in my case this is a Sky Broadband Hub. This is the device that I have attacked for my investigation. This is because it has had previous issues identified with other similar models in the past and I wanted to see if I could still exploit it even after a couple of firmware updates since then.

Alfa Network AWUS036NHA AR9271L Atheros Chipset

ALFA AWUS036NHA provides 2.4GHz 150Mbps Wi-Fi data transfer speeds. It comes with the latest 802.11ac network standards with hardware-based Wi-Fi optimization (wireless types 802.11n, 802.11bgn, 802.11b, 802.11b/g, 802.11g), plus an external high gain antenna. This has been used in my investigation in order to achieve monitoring mode on the Virtual Machine which I needed to be able to do some attacks on the device.

Lenovo Yoga 530

The Lenovo Yoga 530 running Windows 10 has been used to host a Virtual Machine using Oracle VM VirtualBox. This has been used to download Kali Linux on my Windows machine and was where I conducted most of my attacks.

Software Tools:

Kali Linux

Kali Linux is an open-source, Debian-based Linux distribution where you can perform various information security tasks, such as Penetration Testing, Security Research, Computer Forensics and Reverse Engineering. [14] Kali Linux is maintained and funded by Offensive Security. Many individuals use this platform as it is perfect for ethical hacking, which is what I did in my case, and some individuals may use it for exploiting security vulnerabilities for malicious use.

I have chosen to download Kali Linux on my Windows machine using a Virtual Machine called VirtualBox. This is a free desktop visualization platform supported on Windows, macOS and Linux and it comes with more functionality than other platforms, such as VMware, for the price. VirtualBox User Interface is very simple and user friendly. Settings are split into Machine Tools and Global Tools and with Machine Tools used for creating, modifying, starting, stopping and deleting virtual machines.

Kali Linux comes with, and can download, a large range of tools. The tools which I have used in my investigation have been listed below:

Aircrack-ng

Aircrack-ng is a network software package which consists of detector modules, packet sniffer modules, WEP and WPA/WPA2 Passphrase cracker modules and analysis tools for 802.11 wireless Local Area Networks.

Arpspoof

Arpspoof allows users to intercept packets on a switched Local Area Network. It is useful when redirecting packets from a victim device, which will be my smart plug, on the Local Area Network which is intended for another network host, my home sky router, on the Local Area Network by forging ARP replies.

Nmap

Nmap is a tool that allows hosts and services on a computer network to be discovered by sending packets and analysing responses. Some features which I have used in my investigation include probing computer networks which include host discovery and service plus operating system detection.

Wireshark

Wireshark is a popular network protocol analyser which can be used to inspect and sniff network packets that are transmitted over the network capturing from modules such as Ethernet, Bluetooth, Wireless (IEEE.802.11), etc.

Tcpdump

Tcpdump is a network packet analyser running on a command line interface. Users are able to see TCP, IP and other packets of information that are being transmitted or received over a network of which the host machine is connected to. This data collected can be stored in a file for further analysis or it can run openly on the command line terminal.

Metasploit

Metasploit is a computer security project that provides information about security vulnerabilities which can also aid in penetration testing practices by providing a range of tools users can use to attack a victim device, one of these tools of which is synflood which I have used to conduct a SYN Flood Attack on the smart plug device.

Cupp

Cupp stands for Common User Passwords Profiler and is one of many tools that allow for dictionary creation in order to aid in cracking passphrases such as WPA/WPA2-PSK by using systems to collate personal information given by an attacker about a victim and make passwords out of them.

MobSF

Mobile Security Framework (MobSF) is an automated, all-in-one mobile application pen-testing, malware analysis and security assessment framework that is capable of performing static and dynamic analysis of APK's. [21] Unlike many other malware analysis tools, such as ImmuniWeb® MobileSuite and White Hat Security, this tool is free and easy to use.

Angry IP Scanner

Angry IP Scanner is used for scanning IP addresses with the goal of finding alive hosts and gathering interesting information about each of these identified hosts.

Gping

Gping is a tool that allows a visual aid of ping values when targeting a device by putting the information collected as a graphical representation of the data.

Hardware Analysis

To better understand the device and identify the possible attack surfaces, it is important to look at the device hardware both internally and externally.

External Analysis

Figure 5: Front, Back and Restore Button of Belkin WeMo Smart Plug

Input Ports

- Power Button
    - Used to turn switch on and off
    - Solid Blue LED indicates on
- Restore Button
    - Used to restore to Factory Default Settings. This will erase all settings.

Output Ports

- Flashing LED indicator
    - Flashing Blue & Amber: First setup - Awaiting instructions
    - Solid Blue: Everything is OK and setup
    - Off: Normal (good)
    - Solid Amber: Poor connection
    - Flashing Amber: No connection
    - Flashing Blue: Starting up

[9]

Product Information

- WeMo ID: WeMo.Switch.2B7
- Model: F7C027uk
- Device MAC Address: 58EF68979A11

Other Information

- BS 1363 AC Power Plug
- Tri-angle Screws

Internal Analysis

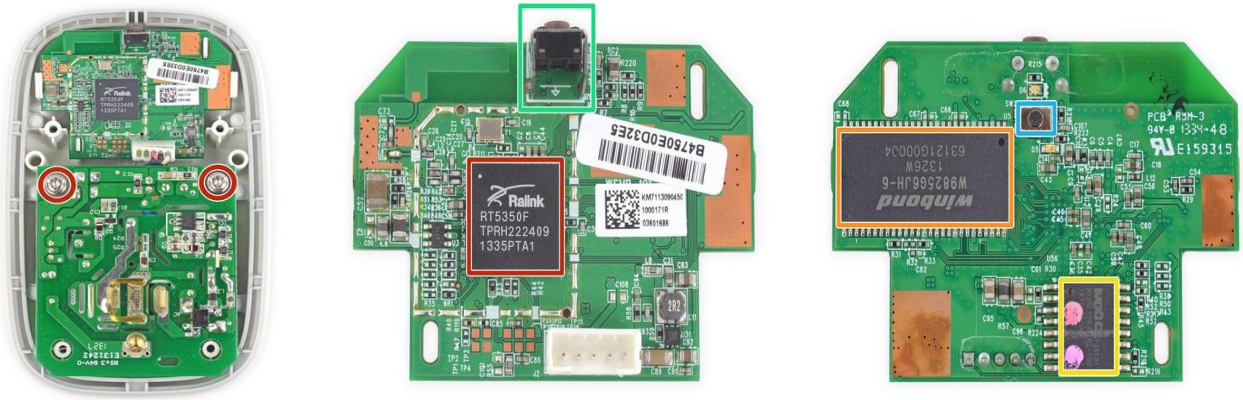


Figure 6: Internal Inspection of Device (Motherboard) [15]

Motherboard Components [15]

- Ralink RT5350 Wi-Fi SoC
- Winbond W9825G6JH-6 256 Mb 166 MHz SDRAM
- Macronix MX25L12835EMI-10G 128 Mb NOR Flash Memory
- Momentary push button switches for factory reset
- Momentary push button switches for power switch

IoT Threat Modelling

An attack surface refers to ways in which a device (including IoT devices) can be compromised. This may be done wirelessly or via hardware and software. The more attack surfaces a device has, the higher the likelihood that I may be able to compromise it. For every attack surface that I discover, it will have an associated risk, likelihood, and impact. The idea is to map out all of the device's features to their associated technical dependencies so that we can identify threat use cases using methods such as STRIDE which was developed by Microsoft. This method is used to help with identifying weaknesses in IoT devices. [16]

The STRIDE method for threat use cases stands for the following:

Threat	Property Violated	Definition
Spoofing	Authentication	Attackers will impersonate something or someone else.
Tampering	Integrity	Attackers will modify data or code without authorisation.
Repudiation	Non-repudiation	Attackers will claim not to have performed an action. This may be malicious activity.
Information Disclosure	Confidentiality	Attackers will expose information to someone who is not authorised to see it.
Denial of Service	Availability	Attackers will make a service unavailable for intended users by bringing down a system.
Elevation of privilege	Authorisation	Attackers will gain administrator level access by upgrading from user to administrator without proper authorisation.

The STRIDE method needs to be rated using a rating system. There are some common rating systems which include CVSS, which consists of three metric groups: Base, Temporal, and Environmental with 14 scoring areas. Another rating system is DREAD, which where risks rate from 1-3. 1 indicating a low risk, 2 indicating a medium risk, and 3 indicating a high risk.

The DREAD rating system stands for the following:

- **Damage potential:** How great is the damage if exploited?
- **Reproducibility:** How easy is it to reproduce the attack?
- **Exploitability:** How easy is it to attack?
- **Affected users:** Roughly how many users are affected?
- **Discoverability:** How easy is it to find the vulnerability? [20]

Rating	High (3)	Medium (2)	Low (1)
Damage Potential	Able to overthrow security and become admin, getting full trust to run as administrator and take over the system.	Able to leak sensitive information	Able to leak sensitive information (non-trivial)
Reproducibility	The attack is always reproducible (regardless of time)	The attack can be reproduced	Even with information about the vulnerabilities of the system, it is still hard to reproduce the attack
Exploitability	Someone with minimal skill would be able to complete the attack and execute the exploit.	A skilled attacker would be able to complete the attack repeatedly.	A very skilled attacker with a lot of experience can complete the attack with in-depth knowledge.
Affected Users	All users, default configurations, all devices	Will affect some users, some devices, custom configurations.	Will affect a small percentage of users and/or devices through an obscure feature on the system.
Discoverability	The attack can be easily found with it being a published attack and/or vulnerability.	Affects a seldom-used feature where an attacker would need to be very creative in order to find its vulnerabilities.	The attack is vague and obscure meaning that it is unlikely to be exploited.

At the end of the DREAD analysis there is a risk rating for the vulnerabilities which are as follows:

Risk Rating	DREAD Score	Comments
High	12 to 15	This is a severe risk vulnerability which should be considered and inspected in a short time period.
Medium	8 to 11	This is a moderate risk vulnerability which should be considered once critical and severe vulnerabilities have been addressed first.
Low	5 to 7	This is a low-risk vulnerability which does not pose a very large risk to the Information Technology infrastructure.

Threat Model Diagram

Here I have carried out a threat model diagram in order to visualise assets of my smart home environment. This will also help to visualise and identify the key areas for attack, the methods I would use to attack the smart plug and its impact.

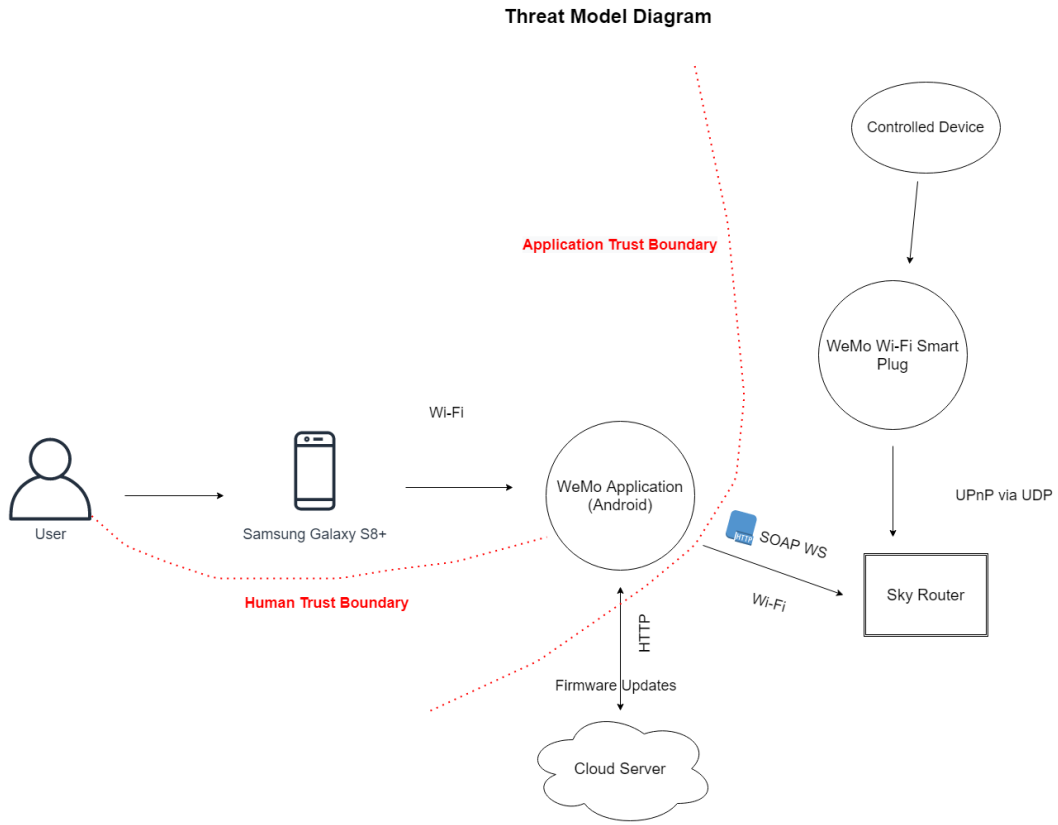


Figure 7: User Case Threat Model Diagram

Architecture Description

WeMo Application (Android)

A simple mobile application that lets you control your devices via the user interface it provides. There are up to 8 modes depending on which device you have which are: turn device on/off, schedule, auto-off timer, long press, motion detector, away mode, motion alerts and usage alerts. The smart plug that I am using has the 4 modes: turn device on/off, schedule, auto-off timer and away mode. The application requires you to discover your WeMo device with your mobile device and the mobile application and connect to a home network.

Samsung Galaxy S8+

A mobile device which I will be using to control the device. This connects to the WeMo Application via a Wi-Fi router that has “802.11 g or n” wireless communication and, in my case, needs to be an Android running v2.3 or higher (alternatively an iPhone running iOS v5 or higher).

Protocol Implementation

Communication between the WeMo Application and the WeMo smart Wi-Fi plug is built upon UPnP (Universal Plug and Play), several protocols including User Datagram Protocol (UDP), Simple Object Access Protocol (SOAP), Hyper-text Transfer Protocol (HTTP) and Simple Service Discovery Protocol (SSDP). They are used for various tasks separately but together implement the functionality required for device discovery, description, control, and eventing. UPnP control and eventing messages can only occur after the mobile app has obtained the UPnP description of a WeMo device. [23] The device will use HTTP when communicating with the cloud server.

Cloud Server

The cloud server is used when there is no Wi-Fi connection between the application and the smart plug and, therefore, they connect via the cloud server. This is also where the device will get its firmware updates when they are released.

Trust Boundaries

Depicted by red dashed lines in Figure 7, trust boundaries involve the boundary between the responsibilities of the customer (human trust boundary) and the responsibilities of the cloud provider (application trust boundary).

STRIDE Threats

Following on from the threat model diagram, I will be using the STRIDE method, as explained previously, in order to identify threats of the nature of the device's architecture.

Threat	Analysis
Spoofing	System could be examined for spoofing where an attacker will successfully identify as the user by falsifying data and gain the ability to control the device without the user knowing.
Tampering	System could be examined for tampering where an attacker will review messaging communications between the device and its applications. This also provides the opportunity to tamper with the device's firmware in order to control the device.
Repudiation	System could be examined for repudiation where an attacker will perform illegal operations without having to log in. This may also be an opportunity to disable login and tracing functionalities.
Information Disclosure	System could be examined for information disclosure where an attacker may identify clear text communications (as the device does use HTTP communication instead of HTTPS) by using sniffing functions.
Denial of Service	System could be examined for denial of service where an attacker could attempt to exploit the "forgotten password" feature on the login page in order to lock a user out of the mobile application. Explore possible denial of service attacks making the user unable to use the device.
Elevation of privilege	System could be examined for elevation privilege where an attacker can find a way to test whether they can gain administrative privileges or not on the application so they can control the smart plug.

Identifying Threats

Threat #1

Threat Description	An attacker may remotely take over the WeMo Wi-Fi smart plug and control the plug without the user knowing
Threat Target	WeMo smart plug user
Attack Techniques	Installing the WeMo application and attempting to connect to the plug using their own mobile device when in close range to it

Threat #2

Threat Description	An attacker may lock user out making them unable to access and use the application
Threat Target	WeMo smart plug user
Attack Techniques	Attempting to try and lock the account and lock the user out by attempting multiple incorrect passwords attempts or using the “forgot password” feature on the mobile application

Threat #3

Threat Description	An attacker may try to access and exploit the APK source code by analysing its contents
Threat Target	WeMo Mobile Application
Attack Techniques	An attacker may use an application to decompile the APK source code and use it to find vulnerabilities of the smart plug and mobile application

Threat #4

Threat Description	An attacker may attempt to find open ports and exploit these if they have existing vulnerabilities available
Threat Target	WeMo Smart Plug
Attack Techniques	This may be done by using tools such as nmap to scan ports associated with the smart plug and see whether there are any open ports. This is called port scanning.

Threat #5

Threat Description	An attacker may sniff sensitive information within the packets and communication between the smart plug and the network
Threat Target	WeMo Smart Plug
Attack Techniques	This may be done via a man-in-the-middle attack which can be done in several ways using tools such as Wireshark, arpspoof, ettercap etc

Threat #6

Threat Description	An attacker may be able to boot the device off the network by overflowing the smart plug with packets of information and making it unusable
Threat Target	WeMo Smart Plug
Attack Techniques	This may be done via a deauthentication attack which can be used to send multiple packets to the smart plug device using tools such as aireplay-ng

Threat #7

Threat Description	An attacker may be able to capture and crack the WPA/WPA2 wireless passwords and use these to access sensitive information on the network
Threat Target	Network
Attack Techniques	This may be done using a deauthentication attack and sniffing tools in order to boot the smart plug off the network and wait for it to reconnect. Once the device has reconnected to the network, the WPA handshake may be able to get captured by sniffing the network and later cracked using a series of tools.

Threat #8

Threat Description	An attacker may install malicious firmware to the WeMo smart plug
Threat Target	WeMo smart plug firmware
Attack Techniques	This may be done by sideloading dangerous and malicious content to the smart plugs' firmware during a firmware update. This may let attackers gain access to the system and sensitive

	information.

Threat #9

Threat Description	An attacker may be able to get administrative privileges and access to filesystems, resulting in them being able to attack the Local Area Network (LAN)
Threat Target	WeMo smart plug firmware
Attack Techniques	This access may be achieved via SSH or Telnet where an attacker may discover a buffer overflow to access filesystem contents and utilize post exploitation tools. There is also a possibility that the malicious actor may locate a known bug in libraries used by the smart plug.

Threat #10

Threat Description	An attacker may be able to access local files, information and resources on a mobile device
Threat Target	WeMo Mobile Application
Attack Techniques	This may be possible if a hacker is able to discover flaws in API communications that expose a WebView to a JavaScript bridge for access to local files, information and resources. An SQL injection attack may be used for SQLite calls locally on the mobile device to attach a database and create a file which has access to local resources.

Threat #11

Threat Description	An attacker may be able to disrupt the use of the smart plug by overloading the device ports with SYN packets.
Threat Target	WeMo Smart Plug
Attack Techniques	This may be possible by use of a SYN Flood Attack which aims to overload the device in a denial-of-service attack using ports that the attacker may find using a port scan known as a SYN Stealth Scan.

Threat Ratings

Earlier in the report, I introduced the DREAD rating system which I will now be using to rate the threats that I listed above by their likelihood of occurrence and their potential impact if they were to occur. Other methods of rating threats are available; however, I will be using the DREAD rating system as I find it easiest to understand and follow.

Final risks are ranked and calculated using the following ratings for reference:

Risk Rating	**Result**
High	12 to 15
Medium	8 to 11
Low	5 to 7

Threat #1

An attacker may remotely take over the WeMo Wi-Fi smart plug and control the plug without the user knowing	
Item	**Score**
Damage Potential	1
Reproducibility	2
Exploitability	2
Affected Users	1
Discoverability	1
Overall Risk Rating: Low	**7**

Threat #2

An attacker may lock user out making them unable to access and use the application	
Item	**Score**
Damage Potential	1
Reproducibility	1

Exploitability	2
Affected Users	1
Discoverability	1
Overall Risk Rating: Low	**6**

Threat #3

An attacker may try to access and exploit the APK source code by analysing its contents	
Item	**Score**
Damage Potential	3
Reproducibility	3
Exploitability	2
Affected Users	2
Discoverability	2
Overall Risk Rating: High	**12**

Threat #4

An attacker may attempt to find open ports and exploit these if they have existing vulnerabilities available	
Item	**Score**
Damage Potential	2
Reproducibility	2
Exploitability	3
Affected Users	2
Discoverability	2
Overall Risk Rating: Medium	**11**

Threat #5

An attacker may sniff sensitive information within the packets and communication between the smart plug and the network	
Item	**Score**
Damage Potential	3
Reproducibility	2
Exploitability	2
Affected Users	2
Discoverability	2
Overall Risk Rating: Medium	**11**

Threat #6

An attacker may be able to boot the device off the network by overflowing the smart plug with packets of information and making it unusable	
Item	**Score**
Damage Potential	2
Reproducibility	2
Exploitability	3
Affected Users	1
Discoverability	1
Overall Risk Rating: Medium	**9**

Threat #7

An attacker may be able to capture and crack the WPA/WPA2 wireless passwords and use these to access sensitive information on the network	
Item	**Score**
Damage Potential	3
Reproducibility	3

Exploitability	2
Affected Users	2
Discoverability	2
Overall Risk Rating: High	**12**

Threat #8

An attacker may install malicious firmware to the WeMo smart plug	
Item	**Score**
Damage Potential	3
Reproducibility	1
Exploitability	1
Affected Users	2
Discoverability	1
Overall Risk Rating: Medium	**8**

Threat #9

An attacker may be able to get administrative privileges and access to filesystems, resulting in them being able to attack the Local Area Network (LAN)	
Item	**Score**
Damage Potential	3
Reproducibility	1
Exploitability	1
Affected Users	2
Discoverability	1
Overall Risk Rating: Medium	**8**

Threat #10

An attacker may be able to access local files, information and resources on a mobile device	
Item	**Score**
Damage Potential	2
Reproducibility	1
Exploitability	2
Affected Users	1
Discoverability	2
Overall Risk Rating: Medium	**8**

Threat #11

An attacker may be able to disrupt the use of the smart plug by overloading the device ports with SYN packets	
Item	**Score**
Damage Potential	1
Reproducibility	2
Exploitability	2
Affected Users	1
Discoverability	1
Overall Risk Rating: Low	**7**

Implementation and Results

Exploiting Android Application Package (APK)

Android Package refers to the package file format that is used by the Android OS, and a number of other Android-based OS for distribution and installation of mobile applications, games and middleware. By exploiting the APK I can see if I can see any weaknesses and sensitive information by assessing and accessing the application source code.

Using Evozi APK Downloader I was able to download the APK of the WeMo application easily and very quickly. This gave me the latest version of the application which matched with the version that I have on my mobile device.

Next, I needed to decompile the source code and scan it for vulnerabilities and information. In order to do this, I used a Java programming language decompiler called “Jadx” which provides an interface to extract source code from class files.

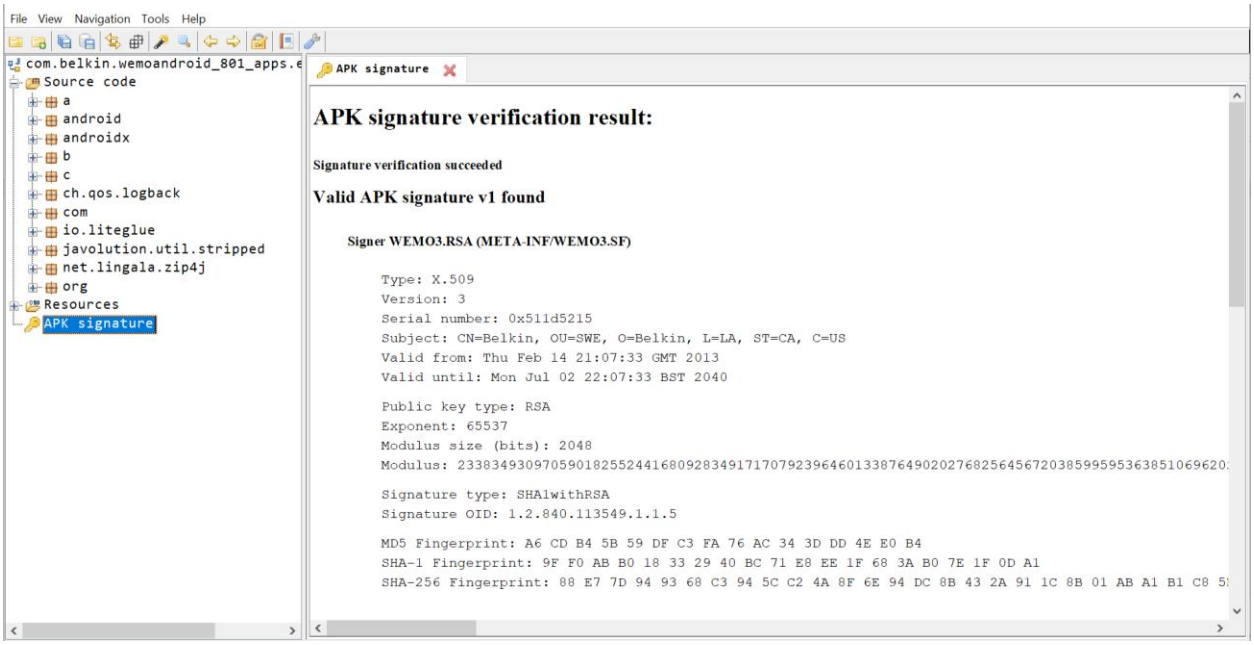


Figure 8: WeMo Application Source Code Using Jadx

After successfully decompiling the source code, I realised that I needed a tool to help me understand and make use of the large amount of code presented to me and for this I chose MobSF which I have described above.

I used a Docker container in order to use the application, running it using a local server on port 8000 to view the user interface as seen in Figure 9. To access the site, you must open port 8000 manually.

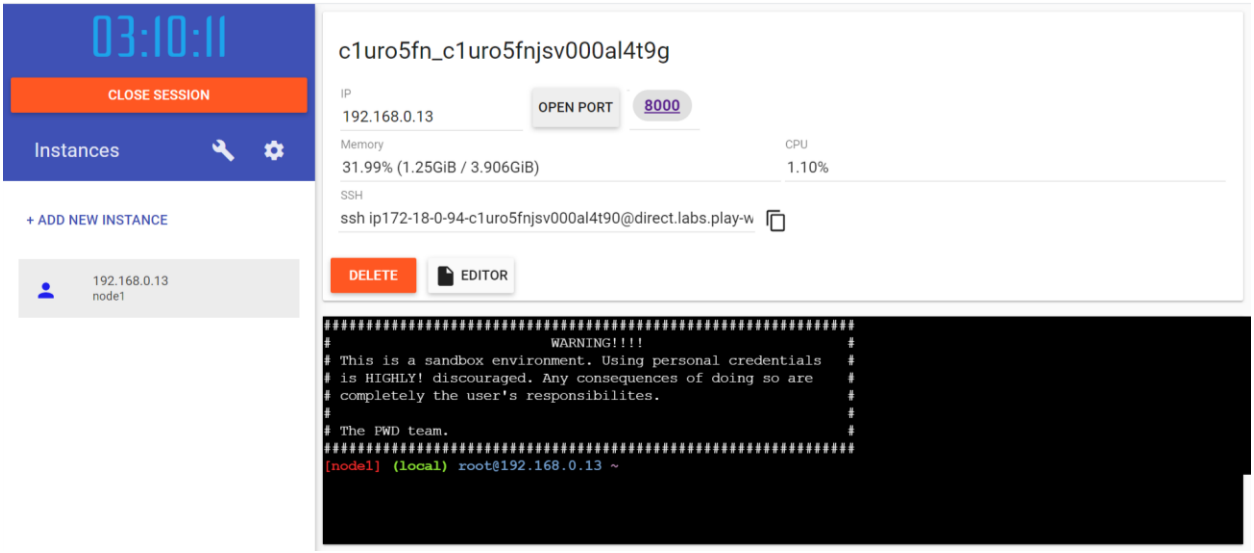


Figure 9: Docker MobSF Instance

I selected the APK file that I had downloaded from Evozi APK Downloader and analysed it using MobSF. This took a couple of minutes due to the large amount of source code present. Presented was a user-friendly interface where I was able to see application permissions, android API, browsable activities, security analysis, malware analysis, reconnaissance and components.

In the “Information” section of the application MobSF gives an average CVSS rating, a security score and a trackers detection score. As seen in Figure 10 below, the WeMo application has a CVSS base score of 6.3 and has an associated severity rating of Medium severity and an app security score of 10/100 (critical risk).

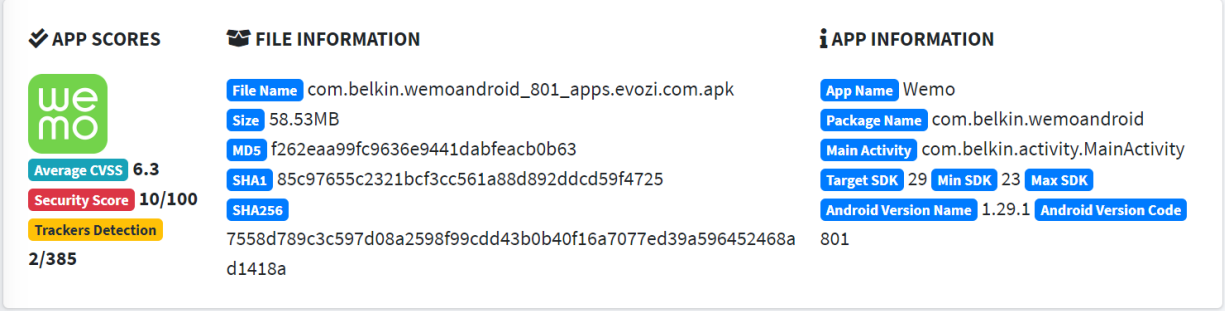


Figure 10: MobSF App Score, File Information and App Information

MobSF offers a security analysis breakdown of network security, manifest analysis, code analysis, binary analysis, NIAP analysis and file analysis and helps to identify poor coding practices and vulnerabilities with the application features. In the tables below I have included a summarised breakdown of important vulnerabilities given in these sections.

Network Security

No	Scope	Severity	Description
1	*	High	Base config is insecurely configured to permit clear text traffic to all domains.

Code Analysis

No	Issue	Severity	Standards
1	App uses SQLite Database and executes raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also sensitive information should be encrypted and written to the database.	High	CVSS V2: 5.9 (medium) CWE: CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') OWASP Top 10: M7: Client Code Quality
2	IP Address disclosure	Warning	CVSS V2: 4.3 (medium) CWE: CWE-200 Information Exposure OWASP MASVS: MSTG-CODE-2
3	The App uses an insecure Random Number Generator.	High	CVSS V2: 7.5 (high) CWE: CWE-330 Use of Insufficiently Random Values OWASP Top 10: M5: Insufficient Cryptography OWASP MASVS: MSTG-CRYPTO-6
4	Files may contain hardcoded sensitive information like usernames, passwords, keys etc.	High	CVSS V2: 7.4 (high) CWE: CWE-312 Cleartext Storage of Sensitive Information OWASP Top 10: M9: Reverse Engineering OWASP MASVS: MSTG-STORAGE-14
5	MD5 is a weak hash known to have hash collisions.	High	CVSS V2: 7.4 (high) CWE: CWE-327 Use of a Broken

			or Risky Cryptographic Algorithm OWASP Top 10: M5: Insufficient Cryptography OWASP MASVS: MSTG-CRYPTO-4
6	SHA-1 is a weak hash known to have hash collisions.	High	CVSS V2: 5.9 (medium) CWE: CWE-327 Use of a Broken or Risky Cryptographic Algorithm OWASP Top 10: M5: Insufficient Cryptography OWASP MASVS: MSTG-CRYPTO-4

Included within the reconnaissance analysis section, there is also a “hardcoded secrets” section which may be of use to an attacker.

Possible Secrets
"alt_secret" : "3f1b7bd6-8e75-11e9-bc42-526af7764f64"
"client_secret" : "6zrihnpo4qofbptvrsmg3tk87yobibqu"
"firebase_database_url" : "https://productionwemoandroidpn.firebaseio.com"
"google_api_key" : "AIzaSyAUtxUo9BeddH2cOgZEQA_pkhFmRd5w1AI"
"google_crash_reporting_api_key" : "AIzaSyAUtxUo9BeddH2cOgZEQA_pkhFmRd5w1AI"
"key_push_warmtime_over" : "Warming time is over"

Lock out user

The purpose of this attack was to start from the beginning of the mobile application journey and try to lock out the user via the login functionality that is required when you first use the WeMo mobile application. The attack involved typing in the wrong username and/or password and examining what actions the server took due to the large amount of incorrect login attempts. The WeMo application login page states that you need to put in an email address and a password in order to login and therefore the attack requires the hacker knowing the user's email address. This would allow me to see if I could attempt a brute-force attack on the login page or lock out the user account via a password limit per account.

Firstly, I typed in a correct username and password which, as expected, logged me into the application with no issues. Next, I entered an incorrect password with the correct email and the response of “Email password combination is not correct” showed up in red on the screen. I was given the opportunity to re-enter the password. I attempted to type in an incorrect password more than 100 times and did not get any warning of password limit attempts, nor was I locked out of the application. I tried the same thing but now with the email input and got the same result. The consequences of not putting a limit on username and password attempts could be very dangerous, especially when trying a brute-force attack on the password and in the case where attackers know the email of the user.

The same procedure was attempted with the “forgotten password” functionality, where multiple attempts saying that the user has forgotten the password were made in order to see if this would lock out the user account. However, again, after 100 “forgotten password” attempts the email was flooded with “reset password” emails but the user’s account was not monitored or locked out.

Control the Device

The purpose of this attack was to see if a secondary user was able to gain access and be able to control the WeMo smart plug. This would be a situation where the attacker would be close enough to the plug to establish a connection. For this attack I used a second mobile device (iPhone 11) that belonged to a family member who was connected to the same network as me. I downloaded the WeMo application on the iPhone and attempted to connect to the plug.

The application prompts the user to sign in and so I attempted to create a new account and connect to the plug through Wi-Fi connection. I followed the step-by-step process on the application in order to set up the plug and I found that in order to connect, I needed to perform a factory reset and boot the Samsung Galaxy S8+ running the application off the device and then connect using the new secondary iPhone meaning that the primary user would know if there was an attempt to control the device in this way.

Next, I decided to investigate further and see if I could connect to the smart plug on two mobile devices using the same account. I logged into the same account on the iPhone that I was using on the primary Samsung device and found that I could control the device on both mobile devices without having to set up the plug or type in the network SSID and password (which would be required if I were to set up the device using separate accounts). The Samsung Galaxy S8+ also did not receive a notification that another user was added to the account.

Although the attack was unsuccessful, due to the large amount of login attempts that we were able to do in the previous attack, it may be possible that someone the primary user knows who has malicious intent could try and log into the same account using a brute-force attack or just information that they already know and control the device.

Port Scanning

Port scanning is used to probe a server or host for open ports. Such applications may be used by attackers to target their victims and identify network services running on a host and exploit its vulnerabilities. Malicious client applications (ex. scripts, bots, malware) often exploit code found in the server software that let them get unauthorized access on the remote machine. This is one of the reasons why testing all ports is vital to achieving an in-depth security verification.

Kali Linux has a great port scanning application known as “nmap”. The result of a port scan can be generalized into one of three categories: Open or Accepted (Host sent a reply indicating that a service is listening on the port), Closed or Denied or Not Listening (Host sent a reply indicating that connections will be denied to the port), Filtered, Dropped or Blocked (No reply from the host). [22]

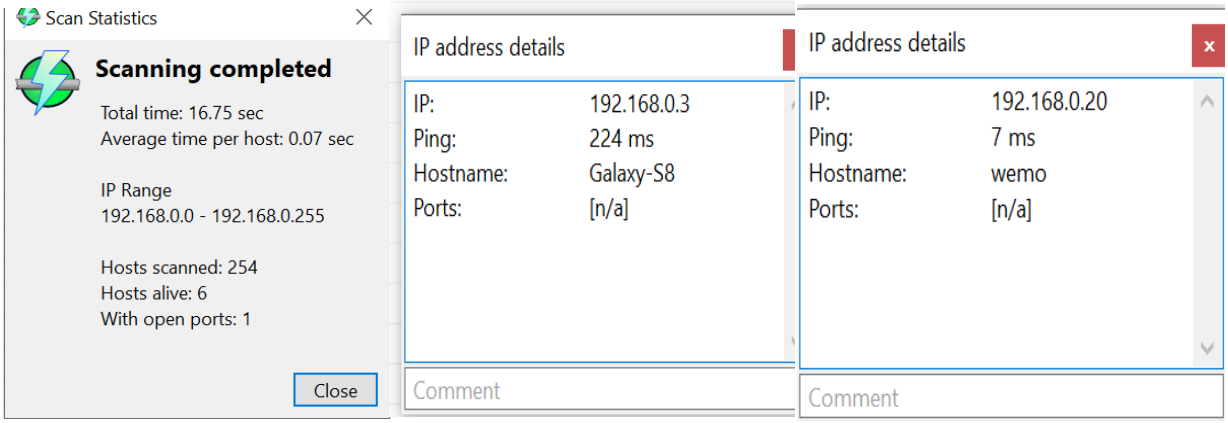


Figure 11: Identifying IP Addresses Using Angry IP Scanner

In order to find the device IP address, I used the Angry IP Scanner application on my machine. I was able to find the IP address of my mobile phone device and the WeMo device by spotting their names on the scanned list. To double check this I also used “arp -a” on my windows command line and matched the IP address to the known MAC address of the smart plug device.

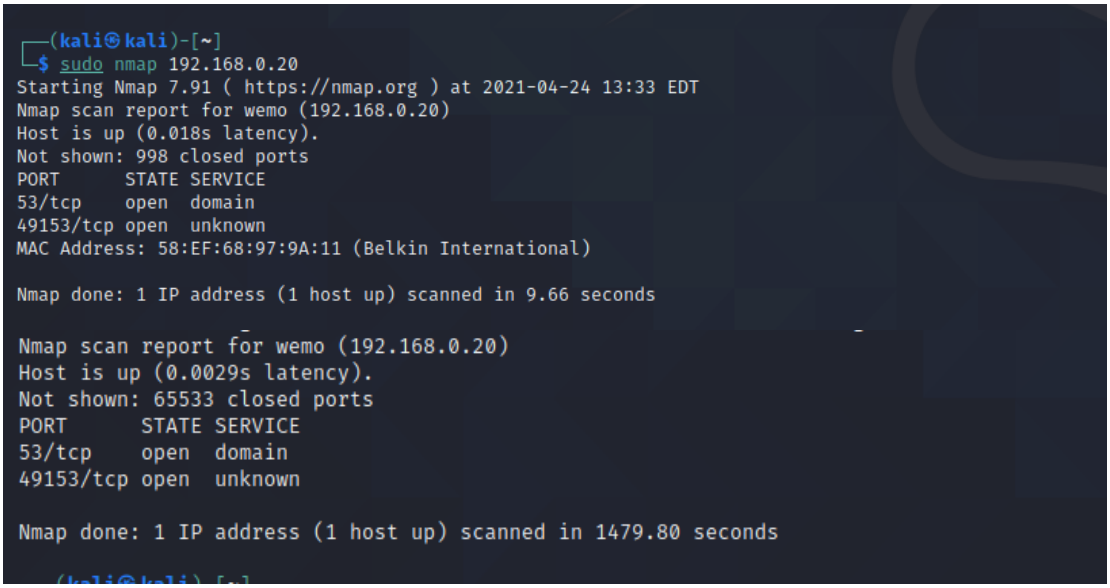


Figure 12: Port Scan using Nmap

I scanned the single IP address for the WeMo smart plug device using “nmap 192.168.0.20” in order to scan UDP and TCP ports. I also used nmap -p- 192.168.0.20 to create a scan report and scan all 65535 ports. There were two open ports - 53 (Domain Name Service (DNS)) and 49153 (Unknown Service). The other ports would have had reliable filters and therefore they were unable to be picked up by nmap. I was also able to confirm the IP address matching to the device as its MAC address showed up in the scan which matches as said on the device. These open ports could be a gateway for attackers to conduct malicious attacks as we will see later in the report.

Deauthentication Attack

A deauthentication attack is a type of denial-of-service (DoS) attack against a user and a Wi-Fi wireless access point. A DoS attack involves an attacker making a network or device unavailable to its intended users by overwhelming and temporarily or indefinitely disrupting services of a host that is connected to the Internet.

I first had to put my wireless adapter (Alfa Network AWUS036NHA AR9271L Atheros Chipset) into monitor mode. A guide on how to set up the wireless network adapter is submitted in the project additional documents. In order to put the wireless adapter into monitor mode I used the command line in the Virtual Machine and typed in a set of commands. The first command was “sudo iwconfig” which showed the parameters of the network interface which are specific to the wireless operation (E.g., interface name, frequency, SSID) and this showed “lo” and “eth0” with no wireless extensions and “wlan0” with a wireless extension. Wlan0 was the network that I changed to monitor mode.

All processes that could affect the “airmon-ng” command must be identified and killed by using the command “sudo airmon-ng check kill” otherwise you tend to come across issues with functionality. Finally, “sudo airmon-ng start wlan0” is used to put the network adapter into monitor mode.

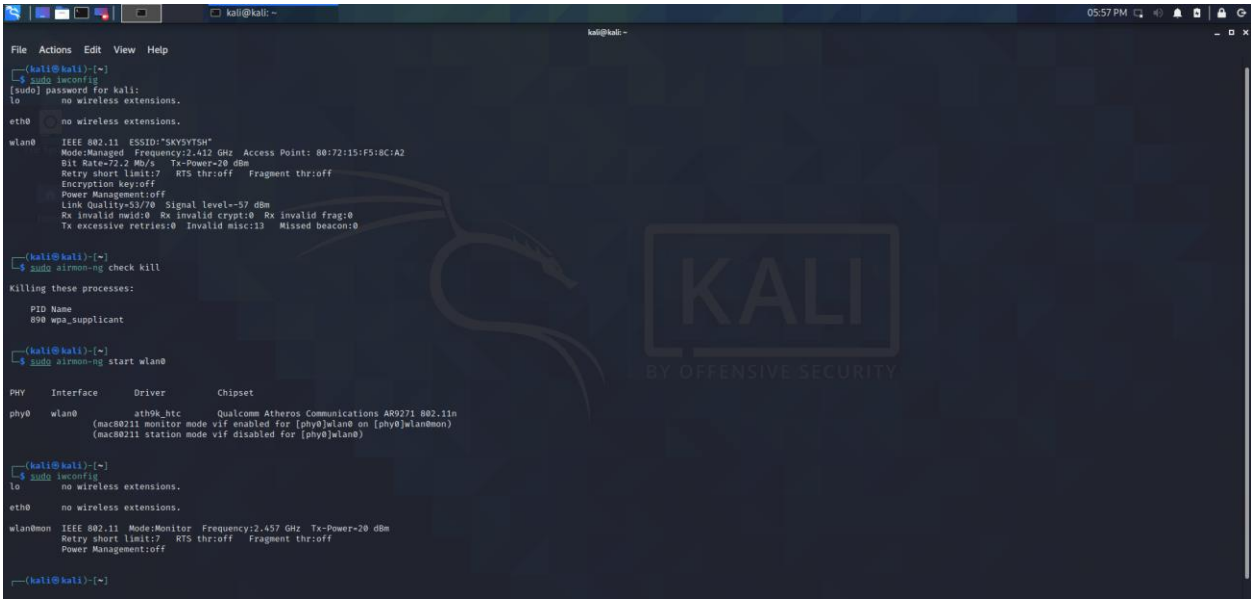


Figure 13: Monitor Mode Setup

In order to check this worked correctly we can use “sudo iwconfig” once again with my network card name “wlan0” now being called “wlan0mon” with changes in its wireless extension description as seen in Figure 13. In most cases the network card name will change to mon0 but in my case it changed to wlan0mon.

The next step in the deauthentication attack was to use a series of commands for the attack. This is used to generate rogue wireless traffic and inject frames in order to perform a denial-of-service attack. Firstly, I was able to get my routers Access Point address from the “sudo iwconfig” command seen in Figure 13 which is “80:72:15:F5:8C:A2”. For the next step, the command “airodump-ng wlan0mon” in order to scan all wireless networks in range. I was able to see my target network access point during the scan as seen in the red coloured box in Figure 14.

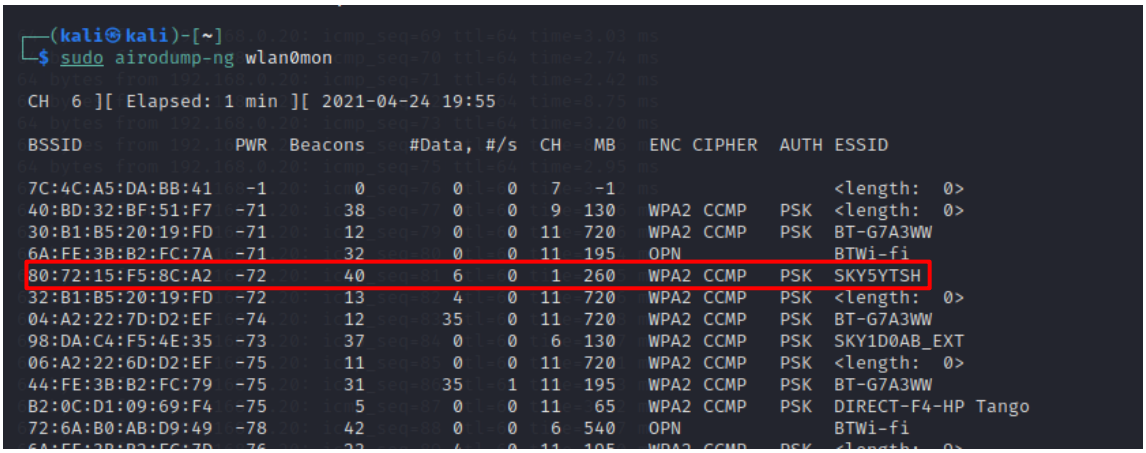


Figure 14: Airodump-ng Wireless Network Scan

Here I was able to note that my network was running on channel 1 which is important in order to focus on the channel in the next command, “sudo airodump-ng wlan0mon --bssid 80:72:15:F5:8C:A2 --channel 1 -w DDoSwemoAttack”, which includes my network card name, the BSSID (mac address) of the router, the channel of the router and “-w” which is a .cap (capture) file that I will open in Wireshark for further analysis of the attack.

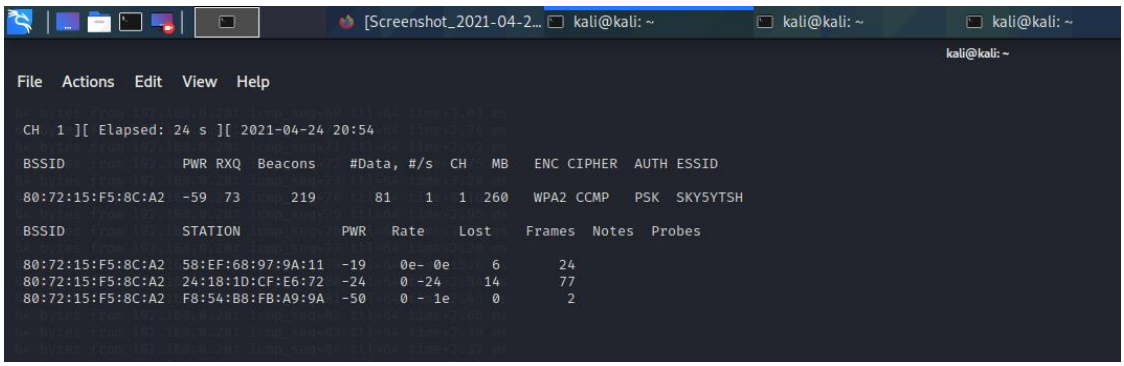


Figure 15: Channel Before Deauthentication Attack

Next I needed to run another terminal window and run the command for the deauthentication attack “sudo aireplay-ng --deauth 2000 -c 58:EF:68:97:9A:11 -a 80:72:15:F5:8C:A2 -D wlan0mon” (Figure 16). The “2000” represents the amount of deauth attacks. If you wanted to only run 20 deauth attacks you would change this to 20 or if you wanted to run infinite deauth

attacks you would change this to 0. The “-c” stands for the client you are attacking and includes the device's MAC address. The “-a” is the router, what is the router the victim is connected to, in this case, it is my own personal router. “-D” can be used to disable AP detection as some modes will not proceed if the AP beacon is not heard and “wlan0mon” is the name of the network card that is still in monitor mode.

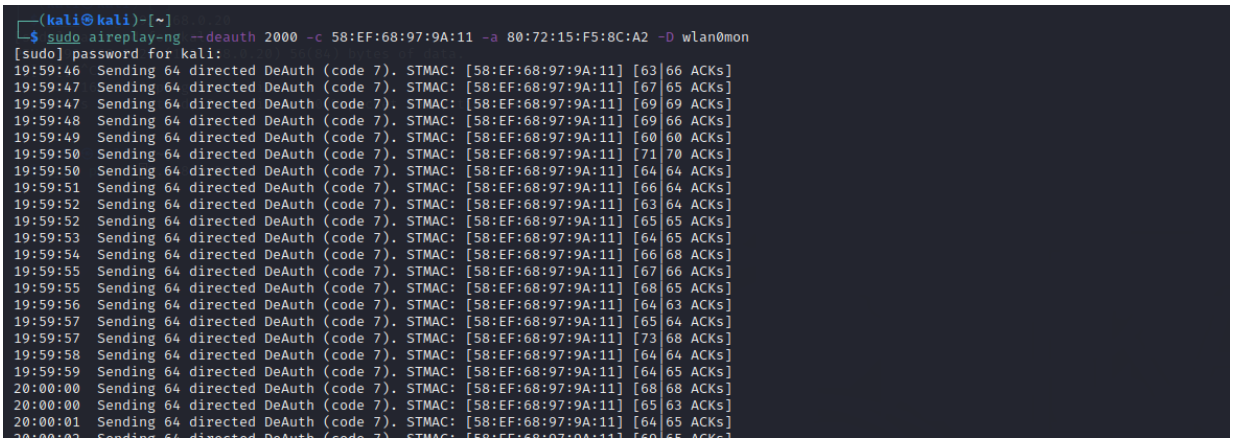


Figure 16: Deauthentication Attack on WeMo Smart Plug

The attack was successful and resulted in the device being booted off the network, with the WeMo Application unable to detect the device and connect to it. As seen in Appendix C, before the attack took place, I was able to monitor the ping of the device using its IP address which was sending packets normally. Once the deauthentication attack occurred the destination host was said to be unreachable, before connecting once again and working as normal.

Most importantly I was able to capture the Wi-Fi WPA2 Handshake using this attack which was captured while the smart plug (who knows a valid password to the wireless access point) was reconnecting to the network. This handshake contains enough information to decrypt the network password which is a very dangerous vulnerability and can give attackers access to vulnerabilities existing within the network as well as outside of it.

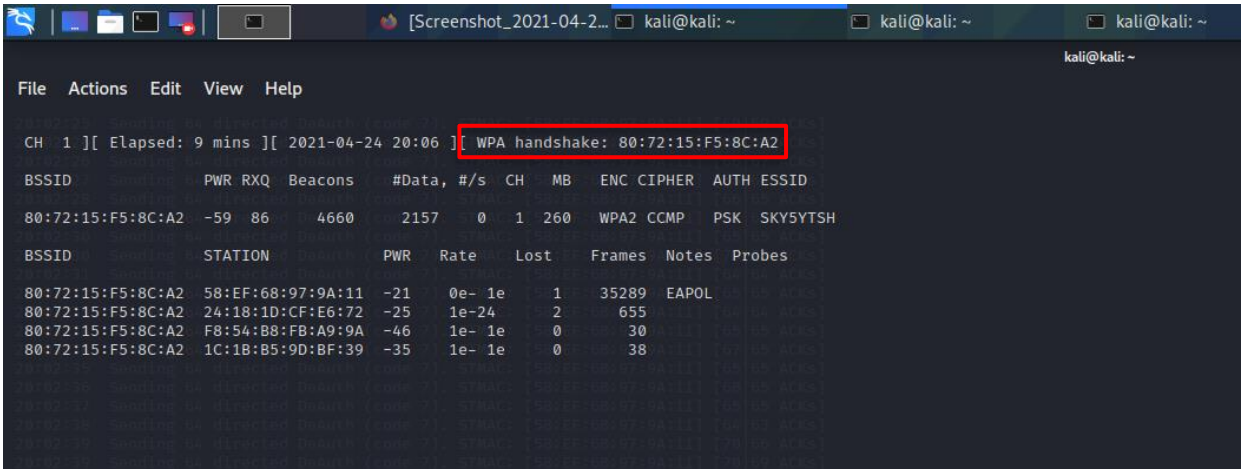


Figure 17: Channel After Deauthentication Attack Showing WPA Handshake

Looking further into this, I opened the ".cap" file in Wireshark in order to analyse the attack in more detail. Here I was able to locate the EAPoL packets by filtering out the handshake. Extensible Authentication Protocol (EAP) over LAN (EAPoL) is a network port authentication protocol that gives a generic network sign-on to access network resources.

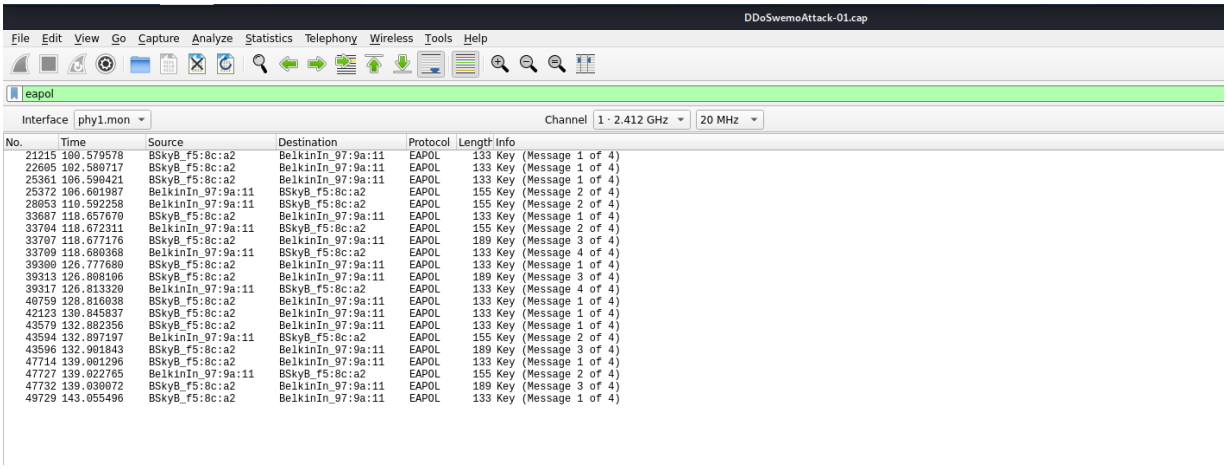


Figure 18: Searching for EAPoL Packets using Wireshark

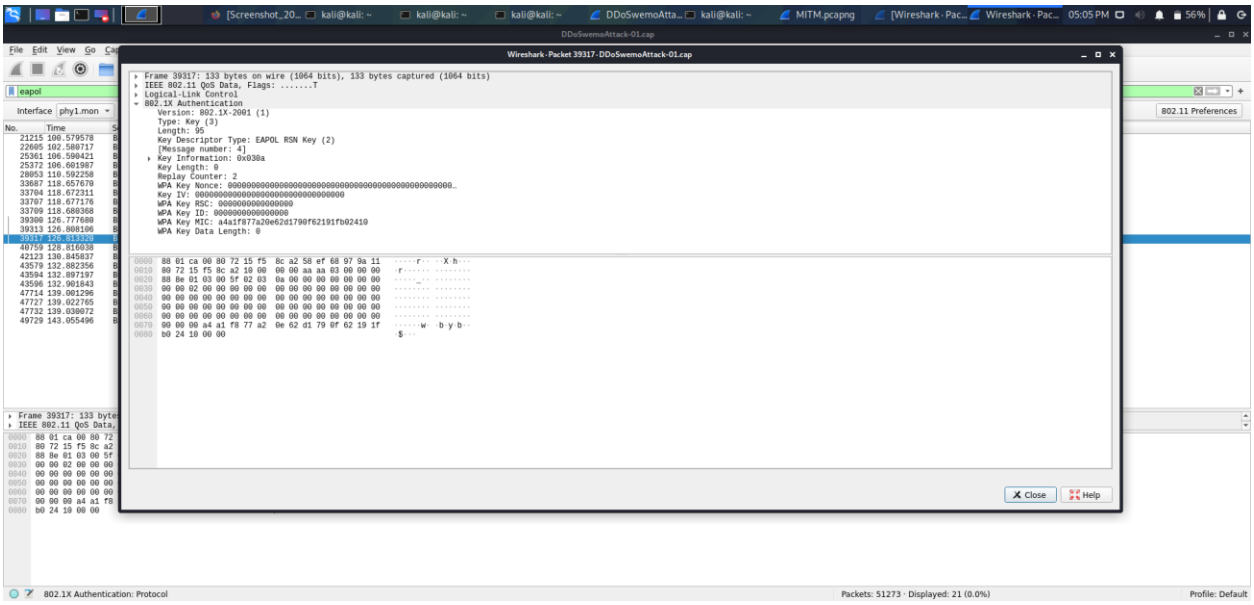


Figure 19: Key Message 4 Showing WPA Key Information

This .cap file would be used to launch a brute-force or dictionary attack on the password in order to crack it. For this you can use aircrack-ng to input the capture file and a dictionary file using "aircrack-ng [name of .cap file] -w [location of dictionary file]". Your dictionary can be large or small and there are many dictionaries online with common passwords that you are able to download and use. The tool would go through all the possible passwords in the dictionary and see if they match up to the WPA handshake key. Success will depend upon how complex the password is and how extensive and comprehensive the dictionary is.

Cupp Dictionary Attack

For more complex passwords you are able to use tools such as “Cupp” (Common User Password Profiler) to generate smart custom dictionaries based on a target's known personal information. This attack would not work on my personal home router password as I have a very complex password that is not personal and randomly generated with several special characters and capital letters, however, many people personalise their network passwords in order to easily remember them which an attacker could use to easily crack them.

In order to access “cupp” you must first download it to Kali Linux using the command “sudo apt-get install cupp”. You can then run the package by typing in “cupp” in the terminal. “Cupp” has a range of tools and I will be looking at the “Interacting questions for user password profiling” argument in order to generate a dictionary of personal passwords. However, there is also the option, “-l” to download large dictionaries online which may be relevant to the target.

I was met with a series of questions including the victims: name, surname, birthdate, name of partner if applicable, partner’s nickname, partner's birthday, name of child if applicable, child's nickname, child's birthdate, pet’s name if applicable, company name. I was then asked whether I would like to: add some keywords about the victim, add any special characters at the end of words, add some random numbers at the end of words and use leet mode (i.e., leet = 1337).

I input a range of dummy data, however, I made sure that this information was information that was available for a potential attacker to see online as if the attacker did not know the person, they would do their research through social media accounts to get as much data as possible. I also chose “y” to all the “y/n” questions asked as a typical user would tend to have these qualities in their passwords.

Cupp then saves the potential passphrases in a file called “[victims first name].txt” and gives you a count of the total number of passphrases that it has generated based on the information that I gave. In my case it was a file called “maria.txt” and I had 37534 words generated.

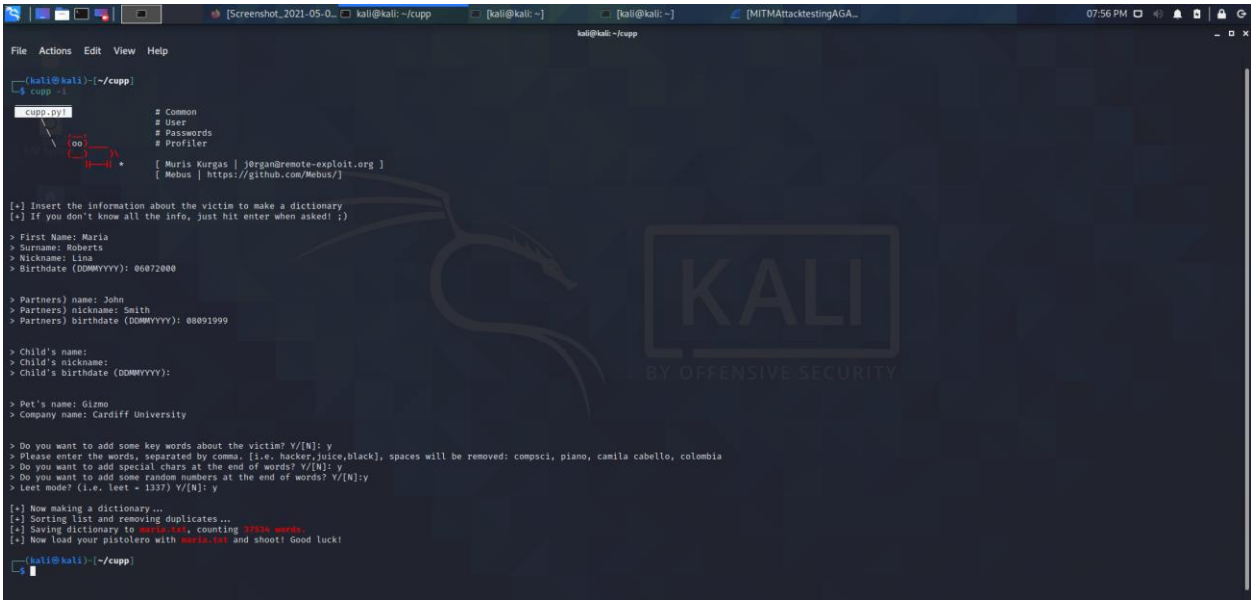


Figure 20: Creating a Personal Victim Dictionary using Cupp

In order to look at the file, I typed in “head -n 100 maria.txt” to view the first 100 lines of the dictionary. As seen in Appendix D, there is a list of generated passwords based on the information I provided.

As an attacker, I was then able to go ahead and try to crack my password using the new dictionary that I just generated. This would be done using the aircrack-ng command stated above, “aircrack-ng [name of .cap file] -w [location of dictionary file]”, in the terminal. In my case, I tested out the dictionary on the .cap file that I had gained previously using the command “sudo aircrack-ng DDoSwemoAttack-01.cap -w cupp/maria.txt”. As expected, the attack did not succeed in cracking the password as seen in Figure 21, however, this may not be the case for other targets with simpler and more personal passphrases.

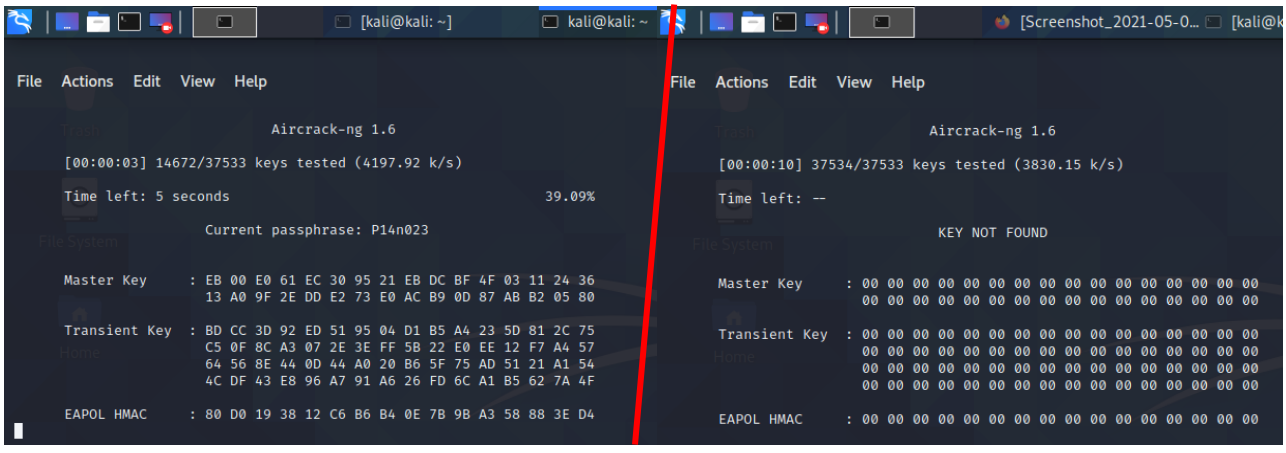


Figure 21: Attempting to Crack Home Network Password using Created Dictionary

ARP Spoofing

This attack is a type of man-in-the-middle attack and can be used in order to sniff and monitor traffic between a target and their connected router and is a way to intercept communication between two nodes. Firstly, I had to forward all the IPv4 network packages which is an important step in making my machine act as a router. If this is not done, then the users internet connection will freeze, and no information will be obtained. To do this I ran the command “sudo sysctl -w net.ipv4.ip_forward=1”

I then had to intercept the packages using “arpspoof”. To do this I typed in the command “arpspoof -i eth0 -t 192.168.0.20 192.168.0.1” and ran it. The “-i” means the network interface name that you will be using, “-t” means the target, in this case I have chosen to target the smart plug (IP 192.168.0.20) and its router (IP 192.168.0.1). The IP addresses are put down in this order. This will now be intercepting packets from the target to the router. I left this to run while typing in my next command in a new terminal. This was the command “arpspoof -i eth0 -t 192.168.0.1 192.168.0.20” which is similar to the previous statement, but the victim IP and the victim router IP have now switched in the statement. This is in order to start intercepting packets from the router to the target. At this point I had successfully started sniffing packets from the victim, however, I then needed to use Wireshark in order to pick out information from TCP streams it was observing.

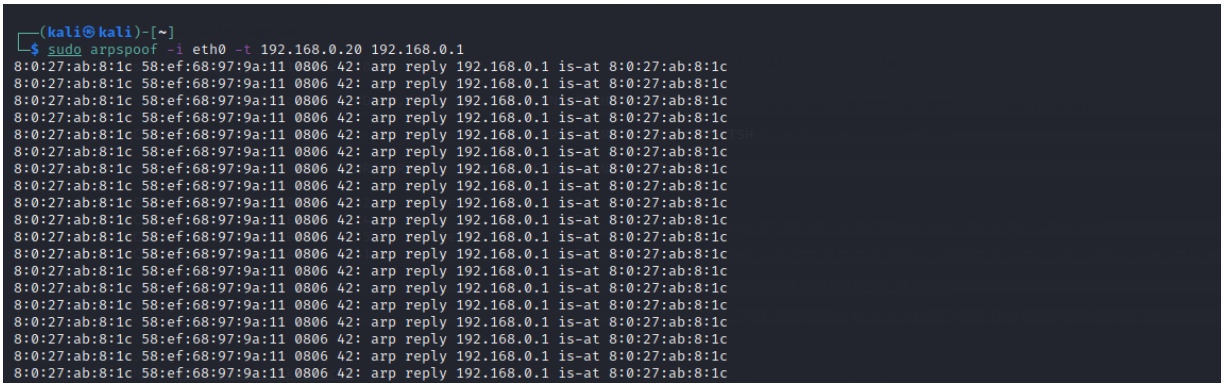


Figure 22: Terminal 1 Intercepting Packages from WeMo Smart Plug with arpspoof

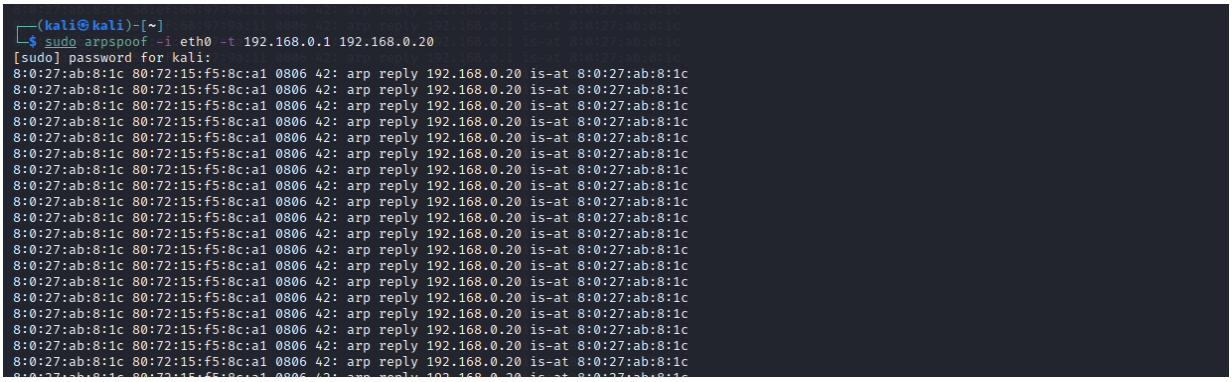


Figure 23: Terminal 2 Intercepting Packages from Sky Router with arpspoof

I opened Wireshark in order to see the traffic I was sniffing, and I was able to see communication between the plug and the router. Due to the plug using Wi-Fi for communication with the Samsung Galaxy S8+, I was able to see the plug being turned on and off by filtering the traffic coming in from the WeMo Smart Plug and it's TCP push packets using "ip.addr == 192.168.20 && tcp.flags.push".

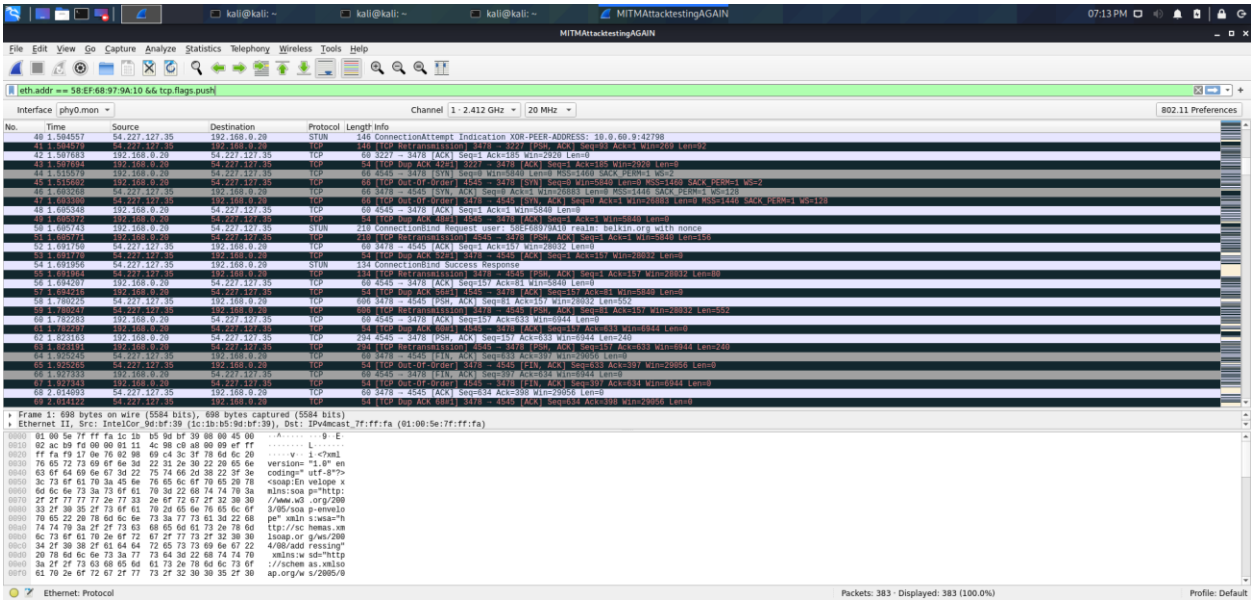


Figure 24: Packets Observed using ARP Spoofing

As seen in Figure 24, you can see the packets being sniffed in Wireshark during the man-in-the-middle attack which is why there are many instances of "TCP Dup Ack" packets. Wireshark was also able to monitor the ARP attack meaning an attacker would be unable to go unnoticed using this tool, however, there was no warning or indication by the WeMo mobile application or the smart plug that there was someone listening to the traffic.

Another tool to see the capture was "sudo tcpdump" where I was also able to see the packet capture between the WeMo smart plug and the network, seeing acknowledgements between the client and the server. "Tcpdump" also picked up the ARP attack (see Appendix F).

SYN Flood Attack

For this attack I have conducted a SYN Flood Attack on the smart plug. Due to the device communicating with TCP sessions, I was able to carry out the attack. As described previously, to form a TCP session there is a three-way handshake involved. The SYN Flood Attack is able to exploit this three-way handshake by overloading the target machine with many SYN packets and consuming all the resources of the target system to create a denial-of-service consequence as the ACK packet fails to come.

I firstly conducted a SYN Stealth Scan using nmap with the command “sudo nmap -p1-65535 80 192.168.0.20”. This attack identified similar open ports to my previous port scan which were port 53 and port 49153 as open ports. The SYN scan is relatively unobtrusive and is stealthier since it never completes TCP connections. [24]

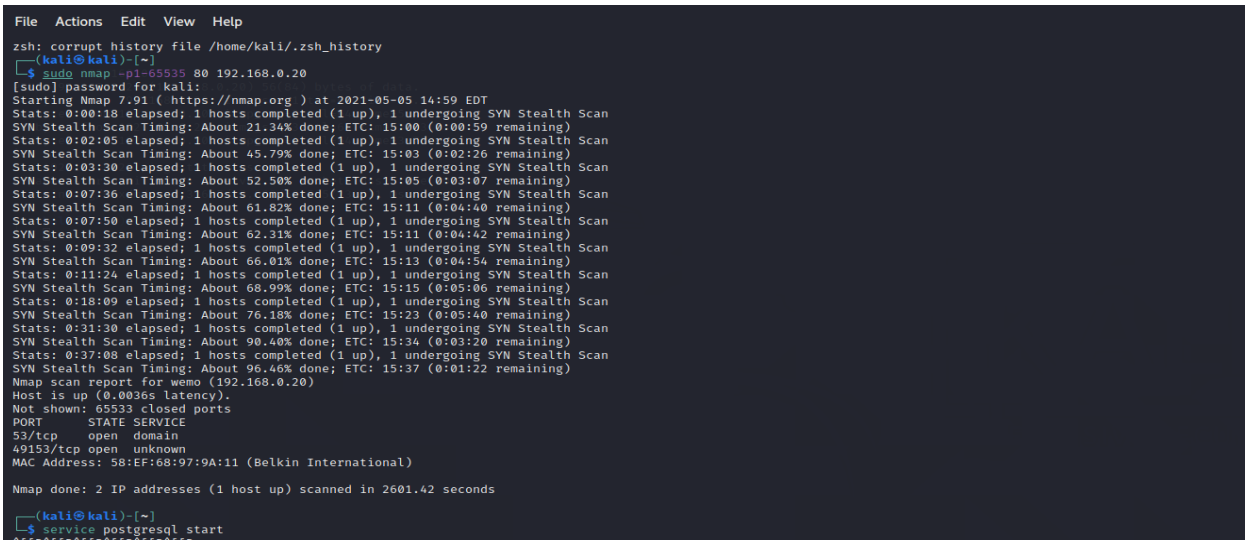


Figure 25: Conducting a SYN Stealth Scan using Nmap

In order to conduct the attack on these open ports I used a tool called “Metasploit” to conduct the SYN Flood Attack. To be able to use Metasploit I needed to use the command “service postgresql start” otherwise Metasploit will fail to run. To start Metasploit using root privileges I used the command “sudo su” for root privileges followed by “msfconsole” to run it.

Figure 26: Searching for SYN Flood Attack using Metasploit

I was then able to search for Metasploit's SYN Flood Attack using "search synflood". This came up with the location where the specific auxiliary is located, and I specified that I would like to use it using the command "use auxiliary/dos/tcp/synflood". The "show options" command was useful here so that I could see the module options and its default values.

To set up my target host I needed to use the command "set RHOST 192.168.0.20" which locked onto the WeMo smart plug's IP address as its target. From here I was able to start going through the open ports that I found using the SYN Stealth Scan (ports 53 and 49153) with the command "RPORT [target port]". I then used the "show options" command every time I made a change to make sure I was targeting the correct things. To start the attack on the smart plug I entered "exploit" into the terminal. Figure 27 shows port 53 being targeted which was my first port of interest, all other port scans have been included in Appendix G-I.

Figure 27: Conducting a SYN Flood Attack on Port 53

I looked at Wireshark to see the impact it was having on the smart plug and I was able to see a very large amount of SYN packets being sent to the ports with over 14,000 packets captured. When attacking port 53 and port 49153 it did not make much of a difference to the smart plug's functionality and it was working normally even with all the packets being sent.

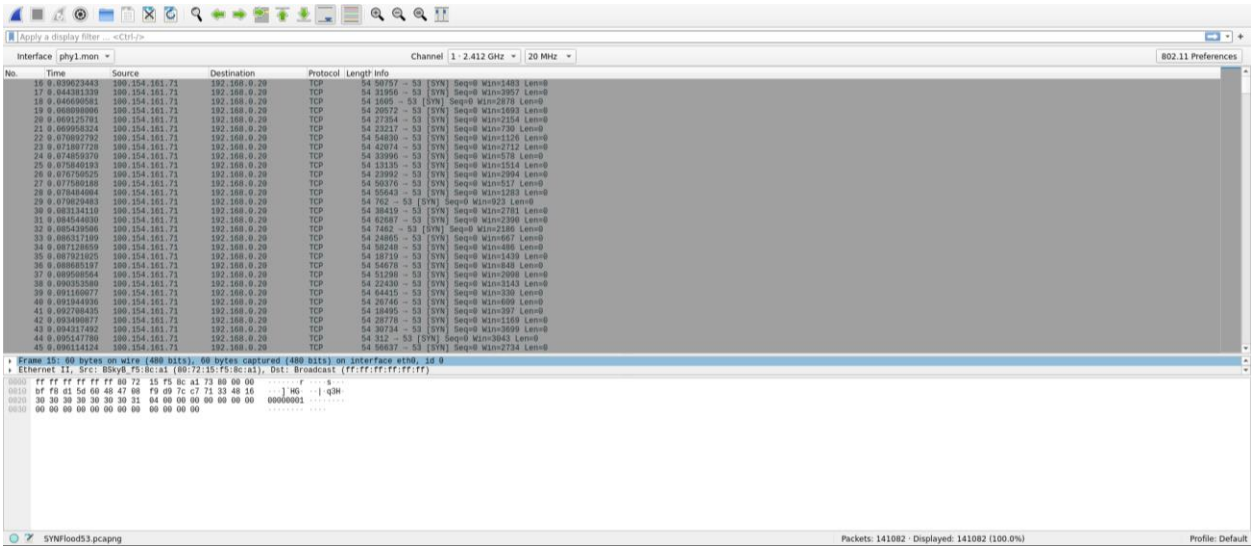


Figure 28: Wireshark Dissection after SYN Flood Attack on Port 53

Due to this I decided to look further into which ports may cause more of a disruption using the previous ARP Spoofing Attack results that I conducted previously. Looking at the capture I found that the ACK/SYN packets mostly came from ports 3478 and 4545 which is seen in Figure 24. I decided to attack these ports and see if it caused more of a denial-of-service consequence. This was successful as when attacking both ports there was a clear lag when trying to turn the plug on and off, with the most damage occurring when attacking port 4545.

To visualise the effects, I decided to conduct a graphical ping scan using “gping” which I downloaded by following the steps given on gping’s GitHub page. Typing in “gping 192.168.0.20” into the terminal to target the smart plug when an attack was not occurring versus when an attack was occurring, the results show a clear disruption with the round-trip time for the packet to reach the host machine and the response to return to the sender, with the time (milliseconds) taking longer when the attack was happening and spikes of disruption reaching 1000 milliseconds.

Figure 29: Graphical Representation of Ping Before SYN Flood Attack on Port 4545

Figure 30: Graphical Representation of Ping During SYN Flood Attack on Port 4545

Evaluation

During my investigation, I was able to point out some security vulnerabilities when conducting a penetration test on the WeMo Wi-Fi Smart Plug. These attacks could result in many consequences depending on the malicious attacker's intent and each of these consequences relating to the attacks carried out will be discussed below as well as suggested countermeasures that may help users and developers when interacting with the systems.

Exploiting Android Application Package (APK)

After conducting a DREAD analysis on this vulnerability, it was given a high-risk rating overall. Accessing the APK was very simple and someone with very little knowledge of hacking would be able to obtain it as it is readily available on the Google Play Store to download using an APK downloader. Analysing the vulnerabilities of this would be difficult without using a third-party application and most of the good threat analysing tools online are very expensive and hard to obtain. Using MobSF as a free application, however, still proved to be useful, where I found possible vulnerabilities of the mobile application which I would later be able to exploit. It is important to note that MobSF found that the application is signed with v1 signature scheme, making it vulnerable to Janus vulnerability on Android <7.0 and the application is signed with SHA1withRSA. SHA1 hash algorithm is known to have collision issues. The attacks relating to SHA1 occur when two different files produce the same hash and an attacker can use this to create two input strings with the same SHA1 hash with less computational power than it would take in the case of a more secure hash function, therefore potentially resulting in a major security breach.

Countermeasures

In order to reduce the risk of an attack relating to exploiting the WeMo mobile application APK, users must make sure that they have the latest firmware available downloaded onto their device as this helps with having access to any security updates that need to be implemented onto the device. Developers of the application need to also make sure that there is no sensitive information hidden in the source code for the general public to see and potentially use maliciously.

Lock out user

For this threat, the DREAD rating was relatively low and achieved a low threat rating overall. The WeMo application did not have security measures in place for attacks involving multiple incorrect passwords or usernames. This meant that if an attacker were to try and brute-force the password using dictionaries or automated enumeration attacks, they may succeed in locking the user out of the account by changing the password. This may also be dangerous as the password used may also be used for the victim’s other personal accounts meaning the attacker will have a record of this passphrase and may be able to use it in other ways. The application also showed no sign of multi-factor authentication, which is when a user is granted access to the application only after successfully presenting two or more pieces of evidence to an authentication mechanism showing

that they are the owner of the account, meaning that it would pose more of a threat than if there was. This was also present on the WeMo website where it showed no sign of password attempt limits or multi-factor authentication.

Countermeasures

In order to reduce the risk of brute-force attacks to the WeMo mobile application login page or the WeMo website developers must make sure to limit the attempts on passwords when they are wrongly attempted too many times on a certain user's username. This may be implemented by adding a password limit and then incorporating a password cooldown period which gradually gets bigger the more times an incorrect password is attempted. Developers may also add a two-step or dual-factor authentication in order to better protect both the user's credentials and the resources the user can access. This may be in the form of mobile pin verification, email verification or a question feature where the user answers a question that they had previously answered when creating their account.

Control the device

The DREAD threat rating for this attack was rated low and I found that it is also very easily avoidable if the steps to secure the application from a brute-force attack are implemented. The device is only able to be accessed and linked up to another WeMo account if it undergoes a factory reset which is available to do on the physical plug itself or by resetting it from a linked account. The only way for two devices to control the device is through the same WeMo mobile application account where two mobile devices linked to the same account can control the device simultaneously. When connecting another device to the plug, however, there is no indication that another device has also connected to the smart plug, meaning that if the attacker were to gain access to the account using a brute-force attack, the victim would not know that there is another user on the account and if connected to the same Wi-Fi, the attacker would not need to set up the device again on their mobile device. Although being able to control the plug may not seem like a huge threat, there are some instances where this may cause large inconveniences or may cause harm. An example might be if the attacker has connected to the plug in a Cafe and the plug is connected to a refrigerator, the attacker could turn off the refrigerator and make the food in it go bad and harm someone if they consumed it. Most WeMo smart plugs can also be connected to Amazon Alexa and Google Home devices meaning that an attacker with a lot of experience may be able to get access to devices that control many things in the home/business. Since the vulnerability is to do with the WeMo mobile application, this threat does not only relate to the Wi-Fi smart plug, but other devices connected to the app such as WeMo Security Cameras, Light Switches etc meaning the attacker can take advantage of all the features on the application such as timers, automation, schedules and alerts.

Countermeasures

Like the countermeasures regarding locking out a user, appropriate security measures must be applied when incorrect passwords are attempted during a brute-force attack. Another useful

feature would be to let the user know which mobile devices are connected to the WeMo devices so that users may be able to spot potential malicious actors easily and a notification system could be implemented to let the user know when another secondary user mobile device has been added.

Port Scanning

The DREAD rating for a port scanning attack was given an overall risk rating of medium. The port scan was simple to do with only a few tools on Kali Linux needed, however, the issue that I found is that when scanning all ports, it took up to 30 minutes which means an attacker would have to have a lot of time on their hands to conduct this attack. Although Port Scanning is not inherently hostile, it is often the first step used by attackers when they are trying to infiltrate a network or get access to sensitive data. I found that ports 53 and 49153 were open using nmap. It also indicated the service that was used for the scan, both TCP protocols, were in use as well. A malicious actor could search port vulnerabilities that are available online and exploit them using the researched attacks. For port 53 specifically, not only does this give an increased potential for the attacker to make Denial of Service attacks, but other attacks such as DNS Tunnelling and DNS Hijacking could occur.

Countermeasures

There may be measures that can be taken in order to secure the open ports and close unused ports. Users have a responsibility to investigate their ports on a regular basis which I believe is something that is widely unknown and could be included as security guidance on the WeMo application or device manual (as well as all IoT device manuals or applications). The user should review all open ports in order to confirm that the ports are being used in the right way and that any system that uses open ports are secure and protected from known vulnerabilities. Users may get a port scanner detector or an intrusion detection system to see whether an attacker is scanning their network and open ports for malicious intent.

Deauthentication Attack

For a deauthentication attack, the DREAD score was rated medium risk overall for the device being unable to use as a result of a flooding attack and an overall rating of high when in conjunction with sniffer tools such as airodump-ng and Wireshark in order to obtain a WPA handshake. When conducting the attack, I found that it was around medium difficulty to carry it out as I needed to purchase and set up tools such as a wireless adapter in order to put it into monitor mode for sniffing which I had not needed to do for previous attacks. However, when the sniffing network was all set up, it was not that difficult to conduct a deauthentication attack. The attack could both boot the user off the device, meaning that the application malfunctioned, and I had to conduct a factory reset in order to use the device again and the device would be booted off the router and eventually reconnect. This would then acquire the WPA handshake which could

be used to crack the network password. Although I was not able to crack the handshake, it was easy to see how easy it would be to do so when a network password is simple and personal. When having access to the network passphrase, an attacker could then conduct a whole new range of attacks from inside the victims' network. The packet data acquired when conducting the attack would also be useful for hackers when starting a reverse engineering attack on the device.

Countermeasures

In order to help prevention of deauthentication attacks and avoid network passphrases being cracked 802.1x authentication methods could be used which will provide better security than WPA and WPA2 pre shared keys. By implementing this, each client's traffic will have unique encryption, and therefore an attacker trying to eavesdrop on the packets will not be able to decrypt the traffic. Another method of prevention would be for users to always make network passphrases complex and hard to crack and maybe even change these passwords on a regular basis. Individuals are also able to enable client isolation which is used to prevent clients from communicating with other wireless clients, however, this measure will prevent devices like printers from operating normally which means it should only be an option if you require maximum security like in businesses etc.

ARP Spoofing

This attack was given a DREAD overall risk rating of medium. The attack was successful and resulted in me being able to sniff packets of TCP information which I was not able to see before when just using Wireshark normally. I was able to see the TCP handshake involving ACK, PSH, SYN and FIN acknowledgements. An attacker with a lot of experience might be able to figure out and record the transmitted data sniffed from turning the plug on and off and pipe the data found to the port that the device is listening on and using. This would make the attacker able to control the plug from their attacking machine. Wireshark and tcpdump was able to pick up the ARP attack happening which means it would not be able to go unnoticed if users were monitoring their network. Not only is this a means of denial-of-service attacks and man-in-the-middle attacks but session hijacking may also be a huge issue when users are using public Wi-Fi since attackers can use ARP spoofing to intercept session ID's and open doors to victim's private data.

Countermeasure

To help prevent ARP Spoofing attacks, users may use a VPN (Virtual Private Network) which will provide the user with an encrypted tunnel to be able to stop the sniffing from ARP spoofing attacks. Instead of just focusing on prevention of ARP Spoofing Attacks it might be useful for the user to have a detection method put in place using a third-party detection tool which can detect the ARP Spoofing Attack while it is occurring and try to apply methods to stop it. A suggested tool might be XArp which is a free software that aims to use active and passive modules to detect attackers trying to perform ARP attacks on the victims' network.

SYN Flood Attack

This attack has a DREAD overall risk rating of low and conducting the attack was relatively easy with the right tools, however, the SYN Stealth Scan took over 30 minutes to complete meaning the attacker would need the time to conduct the attack. The attack was successful in causing disruption to the smart plug, making it buffer when turning the plug on and off, but it did not cause the plug to stop working completely. It is important to note that when using this attack, the attacker tends to spoof their IP address meaning that their identity is more difficult to discover if they were to be caught and this feature is automatic when using the synflood auxiliary on Metasploit. Although it might not seem like a lot of impact, this attack could still be dangerous as often an aggressive attack could cause enough disruption to result in the device crashing and in scenarios where the smart plug must be kept on, it may cause harm.

Countermeasures

There are some methods out there that users can use so that they can prevent falling victim to a SYN Flood Attack such as increasing the backlog queue since each operating system on a victim device has a specific amount of half-open connections that it may allow and, therefore, a response to an abundance of SYN packets during a SYN Flood Attack may be to increase the maximum number of possible half-open connections the operating system will allow. This means that the system needs to reserve extra memory so that it can handle these new requests. Users may also install an IPS (Intrusion Prevention System) to detect the attackers' movements and installing up-to-date networking equipment which includes rate-limiting capabilities may also be useful to stop the attack having its denial-of-service consequences.

Future Work

Reverse Engineering the Firmware

The firmware of a device is a binary that is written to a hardware device which enables control of user applications and various system functions. It contains low level programming code, and this enables the devices software modules to access its hardware functions. Embedded systems, for example, mobile phones, smart devices, drones, etc, tend to contain firmware binaries and are devices which have limited hardware resources, such as memory and storage. The firmware of a device consists of a bootloader, kernel, filesystem, and a number of other resources. There are different types of firmware built upon embedded Linux, embedded Windows, Windows IoT core, and various Real Time Operating Systems (RTOS). [16]

In the future, I would like to improve this project by exploring the firmware of the smart plug device. The firmware download was unavailable online as WeMo does not provide a firmware binary download for the F7C027uk Wi-Fi smart plug model or for any of the other smart plugs similar to it. This, therefore, would involve me needing to take the device apart and acquiring the firmware by dumping it directly from the device. The device also requires a triangle style screwdriver in order to open it which are frequently used in electronics and appliances and its design makes it difficult for people to tamper with safety covers, adding an extra layer of security to the device. This process involves UART, SPI, or JTAG and finding its flash storage. You can then use your UART pins directly or use an 8-pin SOIC chip-clip so that you can dump the firmware using flashrom and an SPI-enabled hardware board such as a Shikra. You would then dump the firmware contents to a bin file using flashrom.

From here you can feed a tool called “binwalk” the binary file using the command “binwalk -e [firmware-binary-name]” and it will reverse engineer the firmware to create a table of recognisable content including its decimal location, hexadecimal location and its description. Binwalk also has a useful tool that lets you detect the entropy of a given firmware binary file, with the command “binwalk -E [firmware-binary-name]” and shows you a graph depicting whether the image is encrypted or not. Figure 31 shows an example of this with the firmware binary download for the “Belkin Surf Wireless Router, F7D2301” which is one of the firmware downloads available on Belkin’s support page. The example shows that some of the data is compressed but none of the data is encrypted as the entropy table shows no edges are equal to 1.

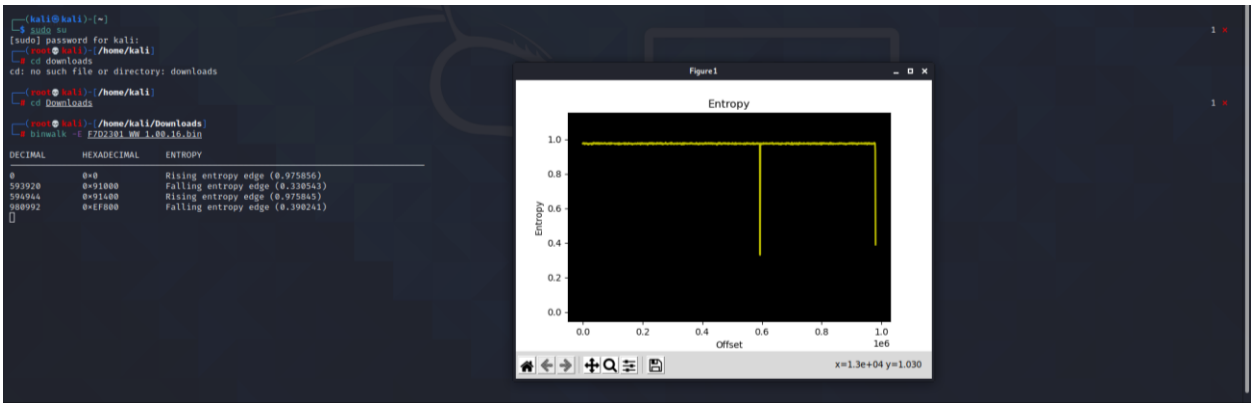


Figure 31: Entropy Detection of Belkin Surf Wireless Router Firmware Binary

After this you are able to perform a vulnerability analysis on the firmware, which is a manual process because, as of yet, there are no full free frameworks that allow you to perform a firmware vulnerability scan. This manual analysis may involve digging deeper into its configuration files, looking at web directories, looking at password files, hunting for backdoors, etc. If any vulnerabilities are found, an attacker would be able to exploit these depending on their skill set and how many weaknesses are found.

WeMo Mobile Application Emulator Testing

For my future work I would also like to look deeper at the WeMo mobile application's vulnerabilities as the application's weaknesses would pose a threat just as much as attacking the smart plug. For this I would use an Android Emulator which is a system that enables one computer system (called the host) to behave like another computer system (called the guest) [25]. I would also be using Burp Suite which is a popular vulnerability testing tool that comes downloaded with Kali Linux. It is a proxy-based tool and can be used specifically for checking web application security.

To visualise the Android Emulator, I would use Android Studio's AVD Manager to create a virtual device using the APK Source Code that I previously was able to download to analyse the applications source code. I would then need to set up Burp Suite to listen to the device and start capturing traffic on the Emulator. From here I would be able to search for application vulnerabilities and see whether an attacker would be able to see login and account information if they managed to sniff traffic from the mobile device.

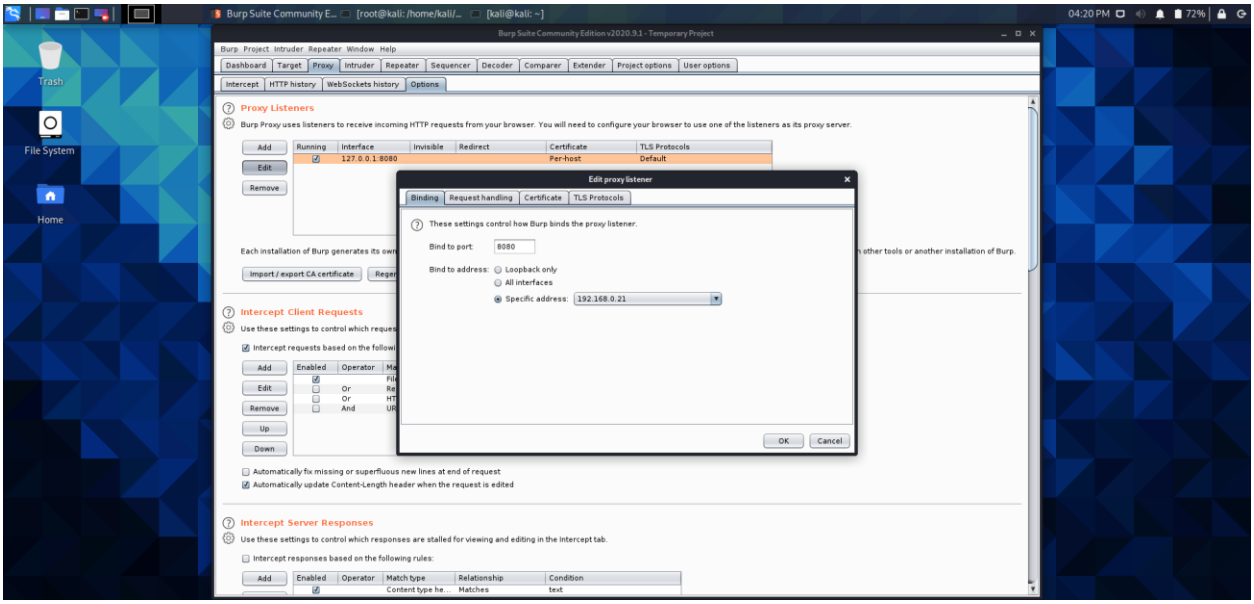


Figure 32: Burp Suite Proxy Listener

Brute-Force WeMo Mobile Application

Finally, I think that another improvement to the project would be another look at the WeMo mobile application and try to brute-force a login password using the vulnerabilities I found during the project where there was an unlimited number of attempts on incorrect passwords without the username having a password restrictor. A brute-force attack aims to allow the attacker to systematically check all possible passwords and passphrases until the correct one is found. This attack would also be interesting to see if the application has any rate limits per IP when guessing more than 100 password attempts per username.

To do this I could also use an Android Emulator and a password cracker tool but an attacker with more experience may be able to attempt this attack on a real mobile device. This could also be relevant on Belkin's website login page where I found the same issue.

Conclusion

In conclusion, when conducting a penetration test on the security vulnerabilities of the WeMo Wi-Fi Smart Plug I found that it is lacking in some security measures which would be simple to implement and further add to the security that has already been put in place on the device. Most of the attacks were successful, finding that prevention against denial-of-service attacks and attacks involving packet sniffing to be some of its biggest weaknesses. Both attacks have proved that they would cause a lot of disruption and may result in the leakage of personal and sensitive data. Furthermore, the WeMo application adds to the device's vulnerability as there are measures for brute-force attacks to be successful if they are attempted meaning that an attacker would be able to gain control of the device and have access to more sensitive information. The device would be its most vulnerable depending on its usage. If the device needs to be always on it would be dangerous for the makers of the device and application not to add extra security measures to the device and its mobile application. Although the WeMo smart plug is advertised for home use, the plug may also be used commercially meaning that areas with public Wi-Fi would be the most vulnerable target.

Some security measures that are successful in the device's architecture are its mobile application usage of SSL certificate pinning and the application being signed with a code signing certificate. Another good security measure is that Belkin has not made the firmware of the device public and readily available to download online which means attackers would have to find harder ways of acquiring the firmware and exploiting it for vulnerabilities. The device is with only a few external features meaning there is less that an attacker could do to exploit these. There is also usage of a tri-angle screw holding the smart plug together which is not a very common screwdriver type meaning that an attacker would have to go to extra lengths to acquire the firmware from the device itself. Finally, since this type of WeMo smart plug has been the target of attacks in the past, Belkin makes sure that there are frequent firmware updates and notifications that edge the user to update their firmware when it is available as soon as possible for enhanced security.

The most useful recommendation that would add another degree of security to the device would be to implement a notification system that tells the user whether their device is being monitored, whether their device is lagging or not working due to a denial-of-service attack and if another user has control of the smart plug using another mobile device. For protection against information leakage and an account take over, I recommend that the mobile application puts in a system that limits the number of incorrect passwords that a user can attempt. I would finally recommend a two-step verification system to be put in place when a login occurs.

Overall, we must take into consideration that the price of the device is proportionate to the amount of security it will contain and with this device costing £27.99, it contains more security measures than I would have expected, however, it might be useful to implement additional layers.

Reflection on Learning

Before choosing my project title, I wanted to embark on a piece of work that was challenging yet exciting. During my final year at university, I had gained an interest in cyber security and forensics after completing relevant modules, however, I was keen to dig deeper and try something that I had never tried before. This project has helped me do this and I have learned a lot of valuable knowledge and skills throughout. At the beginning of the project, I had very little knowledge in the area, but I had a drive for learning which helped me along the way. I find that now I have completed my research, I can better appreciate the lengths companies have to go through to secure their IoT products and was surprised at how much damage their vulnerabilities could cause. Along with my gained knowledge of IoT and security, I have enhanced my abilities using the Linux terminal and I now have a great understanding of how to use different tools that come with Kali Linux. I had also never previously installed and used a Virtual Machine, nor had I used a Wireless Adapter to capture information.

Approaching the project was difficult for me, I had never practiced this aspect of computer science before, and I did not really know where to start my investigation. I therefore used my previous knowledge of agile project management from my relevant modules and group projects and applied it to my individual project. Purchasing “IoT Penetration Testing Cookbook” was also able to greatly help me with my methodology process. Lacking knowledge in packet reading was also one of the main challenges that I faced when doing the project and it took me a while to be able to wrap my head around all the information that was being presented to me. A lack of equipment was also one of the main issues of my project meaning that I could not complete attacks such as reverse engineering the firmware as this would have required me to purchase a tri-angle screwdriver, UART pins or an 8-pin SOIC chip-clip and an SPI-enabled hardware board too late into the project. However, this is something that I would like to do in the future for further investigation into the WeMo Wi-Fi smart plug.

Although I had some previous knowledge of project management, I had never implemented this when working alone and doing my own project. My scheduling skills have greatly improved as I was able to follow my timetable that I created in my project plan and I was able to catch up on my work and keep on schedule when unforeseen commitments occurred, such as a three-day hackathon for my CM3202 module and other personal commitments. I was able to use and improve my risk management and identification capabilities, taking into consideration not only risks of the IoT device but risks of conducting the project such as privacy and ethical issues. I believe that my biggest skill improvement relating to project management was my planning skills because I was constantly planning out each of the individual attacks that I had identified. Another skill that I improved was my research skills as not only did I need to conduct a lot of background research on the Internet of Things, but I was always researching new things and new ways of approaching tasks when trying to conduct my attacks. My general problem-solving skills have also increased as problem solving was a huge aspect when trying out the attacks. I needed to use these problem-solving skills to find the vulnerabilities and connect them to information that I found to exploit the device even further. As well as improving on my written communication skills, a skill that I was not expecting to improve was my oral communication skills. This came in the

form of Microsoft team meetings with my supervisor and fellow peers which allowed me to communicate my progress every two weeks and present some of this progress to my classmates so that they may have been able to benefit from my experiences with the cyber security related project. Finally, the project has been able to help me get a much better understanding of the topic of cyber security and penetration testing which I expect will help me greatly in the future when looking for related work.

Key Stages

Here I will be going through the key stages of the project and specifying what was successful when completing these sections and stating any improvements that could be made to the implementation, if applicable.

The first stage of the project was to conduct background research, I did this using reliable sources such as academic papers from Google Scholar, IEEE Xplore and Microsoft Academic. This improved my research skills as I was not just using the first article on Google that I came across but using a range of academic journals that best suited my target research topics. I think that if I were to improve on my background research, I would spend some more time looking through academic papers as there was a lot to choose from and I may have missed some interesting previous projects. In the future, I could also include more protocols that are used in IoT devices and not just the device that I am investigating to be a bit more informative if someone is trying to replicate a similar project.

The methodology stage of the implementation allowed me to explore many methods of conducting penetration tests which I could use when conducting my own. It improved my network diagram skills and was helpful in letting me visualise my setup and the tools I was using. Here I was also able to gain a better understanding of the privacy implications that come with conducting penetration tests and I was, therefore, better informed for when I did my tests.

I was then able to conduct a hardware analysis on the device. Looking at the external of the device was simple and I was able to gather valuable information about the device, however, I first realised that I needed a tri-angle screwdriver here and I also did not want to take the device apart as I was not experienced in the area and did not want to accidentally cause damage at the start of the project. This led me to have to use a hardware examination that was available online instead of my own. Next time I would, therefore, do further research on how to safely conduct a hardware research and inspect the internal features of the device myself.

The next section was IoT threat modelling and here I was able to learn a lot of new skills. Firstly, I was able to learn about, understand and use both STRIDE and DREAD methodologies to model vulnerabilities of my IoT device and its applications. I was also able to learn how to do a threat model diagram and description which was very useful later in the project. To improve this section, I would go into further detail when modelling the threat diagram. There are also other tools available which further aid in the visualisation of IoT architecture which I would like to use in the

future. CVSS methodologies to describe threats tend to also be more informative, as seen when using the MobSF tool, however, this methodology can sometimes be hard to understand.

The first attack was an Android APK source code analysis which allowed me to see vulnerabilities residing in the mobile applications code. When conducting this attack, I first struggled to find a source code analyser that was available online for free as most of them required me to ask the company for a demo or were hard to install and use. I then came across MobSF which was a great, easy to use, free tool, however, I think that the paid versions would have been more informative. Downloading the APK file was also very easy to do, and this was just a case of Googling a suitable downloader and using the Google Play Store to obtain it.

The attack which involved locking out the user aimed to find out what steps the mobile application took when trying out a large amount of incorrect password attempts. Although I found that there were no password attempt limitations, I think that this section of the project could be greatly improved by attempting a brute-force attack to access the users account and change their password or close their account.

When attempting to control the device using a separate account, the attack was unsuccessful but was still informative as I found the attacker would need access to the primary users account to be able to control the plug, but the user would not be informed if an attacker were able to log into their account or the attacker would have to initialise a factory reset. To improve this attack, I would need to implement the attack used to lock out the user (brute-force attack) and then attempt to control the plug.

Port scanning the device was successful and I was able to see open ports. However, the ports that were open did not seem to be that informative and I was not sure what information I could extract when looking for vulnerabilities with these open ports. In the future, an improvement would be to do some more research on the vulnerabilities of these open ports and try to exploit these instead of just identifying them.

The deauthentication attack was successful as it was able to glitch and boot the mobile device off the plug, resulting in me having to set up the smart plug again and it was also able to boot the plug off and then allow it back on the network meaning that I was able to get the WPA handshake. However, since my home network password was very complex, I was not able to conduct a successful dictionary attack on the passphrase. To improve this attack, I would like to conduct a test where I ask a family member to change my home network password to something personal and then make a complex dictionary based on their personal information and see whether I would be successful in cracking the password. This attack took me a while to do as when inputting “arp -a” into the command line in my Windows machine, I was getting the MAC address “80:72:15:F5:8C:A1” for my home network router which I was attempting to use for the deauthentication attack with no success. However, I realised that when using my Wireless Monitor, “iwconfig” stated that the MAC address was actually one digit higher (which I now know is normal for some devices MAC addresses) and was actually “80:72:15:F5:8C:A2” which was the MAC address I used to cause a successful attack.

The attack that involved ARP Spoofing was successful in me being able to read packets of TCP data that I was unable to see before with Wireshark and making the plug relay information from the network to my machine. However, in the future I think exploring these packets further for more information would be more useful instead of just noting that I was able to see the device turn on and off from the data. I would also like to explore further attacks that can be done with the ARP Spoofing that I have stated in the report.

My last attack was a SYN Flood attack which was successful in sending many SYN packets to different ports of the device to cause disruption. Here I used my problem-solving skills to identify ports that I had seen in the packets in my ARP Spoofing attack and used these to cause lagging in the device. In the future, I would like to delve deeper when using Metasploit as it offers a good range of attacks which would be interesting to explore.

Finally, I think that to improve my project I would need to reverse engineer the firmware instead of just describing how to do it in order to get even more vulnerabilities of the device. There are a very large range of options available for attack when reverse engineering the firmware and this task would greatly improve my understanding and skills of hardware and software analysis.

References

1. Sicari, S. et al. 2012. Internet of things: Visions, application and research challenges. Available at: http://www.dicom.uninsubria.it/~sabrina.sicari/public/documents/journal/2012_IoT_vision.pdf [Accessed 18 February 2021].
2. Johnson, J. 2021. Worldwide digital population as of January 2021. Available at: https://www.statista.com/statistics/617136/digital-population-worldwide/#:~:text=How%20many%20people%20use%20the,the%20internet%20via%20mobile%20devices [Accessed 18 February 2021].
3. Li, S. et al. 2014. The internet of things: a survey. Available at: https://link.springer.com/content/pdf/10.1007/s10796-014-9492-7.pdf [Accessed 18 February 2021].
4. Angrishi, K. 2017. Turning Internet of Things (IoT) into Internet of Vulnerabilities (IoV): IoT Botnets. Available at: https://arxiv.org/pdf/1702.03681.pdf [Accessed 21 February 2021].
5. Yang, Y. 2017. A Survey on Security and Privacy Issues in Internet-of-Things. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7902207 [Accessed 22 February 2021].
6. Najib, W. 2019. Survey on Trust Calculation Methods in Internet of Things. Available at: https://www.sciencedirect.com/science/article/pii/S187705091931957X/pdf?md5=6feac7661d74cc48867cfd700419add4&pid=1-s2.0-S187705091931957X-main.pdf [Accessed 24 February 2021].
7. OWASP. 2018. OWASP Internet of Things Top 10. Available at: https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf [Accessed 24 February 2021].
8. Deep, S. 2020. A survey of security and privacy issues I the Internet of Things from the layered context. Available at: https://onlinelibrary.wiley.com/doi/pdf/10.1002/ett.3935?saml_referrer [Access 25 February 2021].
9. Davidfu. 2011. F7C027 WeMo Smart Switch User Manual Belkin. Available at: https://fccid.io/K7SF7C027/User-Manual/User-Manual-1609816 [Accessed 11 March 2021].
10. Roberts, P. 2018. McAfee Researchers Exploit Smart Plug to attack Smart TV! Available at: https://securityledger.com/2018/08/researchers-exploit-smart-plug-to-attack-smart-tv/ [Accessed 1 March 2021].
11. Gupta, A. 2019. The IoT Hacker's Handbook: A Practical Guide to Hacking the Internet of Things. Available at: http://www.ime.cas.cn/icac/learning/learning_3/201907/P020190724586712846107.pdf?fbclid=IwAR38tZ1dqi_AcrIuloU3Y0JgPu_kymg-gKdMD4IV5e_BomsMKj3QsbYZ4jE [Accessed 3 March 2021].
12. SecureTeam. 2018. Penetration Testing Under UK Law. Available at: https://secureteam.co.uk/articles/penetration-testing-under-uk-

law/#:~:text=In%20order%20to%20ensure%20that,infrastructure%20with%20a%20means%20of
[Accessed 5 March 2021].
13. Romano, P. 2014. WiFi Standards 802.11a/b/g/n/ vs 802.11ac: Which is Best? Available at: https://www.semiconductorstore.com/blog/2014/WiFi-standards-802-11a-b-g-n-vs-802-11ac-Which-is-Best/806/
[Accessed 9 March 2021].
14. Kali. 2021. Kali Linux: The most advances penetration testing distribution. Ever. Available at: https://www.kali.org/
[Accessed 10 March 2021].
15. Wacker, G. 2017. Belkin WeMo Switch Teardown. Available at: https://www.ifixit.com/Teardown/Belkin+WeMo+Switch+Teardown/77953
[Accessed 11 March 2021].
16. Guzman, A. and Gupta, A. 2017. IoT Penetration Testing Cookbook: Identifying vulnerabilities and secure your smart devices. Available at: https://books.google.co.uk/books?hl=en&lr=&id=rEFPDwAAQBAJ&oi=fnd&pg=PP1&dq=iot+penetration+testing&ots=DqBez-Jutj&sig=z9gkqnnxpKFgPSDGHa87Q2mG1qE&redir_esc=y#v=onepage&q=iot%20penetration%20testing&f=false
[Accessed 15 March 2021].
17. Zvelo. 2021. IoT Security Technology For Router & Gateway Manufacturers: Core Technology For Device Profiling and Threat Detection. Available at: https://zvelo.com/industries/iot-security/
[Accessed 16 March 2021].
18. Abdalla, P. and Varol, C. 2020. Testing IoT Security: The Case Study of an IP Camera. Available at: https://www.researchgate.net/publication/342184780_Testing_IoT_Security_The_Case_Study_of_an_IP_Camera
[Accessed 16 March 2021].
19. Telefónica. 2021. IoT Threat Detection. Available at: https://iot.telefonica.com/en/solutions/secure/iot-threat-detection/
[Accessed 16 March 2021].
20. Wikipedia The Free Encyclopaedia. [No Date]. Available at: https://en.wikipedia.org/wiki/DREAD_(risk_assessment_model)#:~:text=The%20DREAD%20name%20comes%20from%20the%20initials%20of%20the%20five%20categories%20listed.&text=When%20a%20given%20threat%20is,to%20prioritize%20among%20different%20issues
[Accessed 22 March 2021].
21. Braham, A. 2021. Mobile Security Framework (MobSF). Available at: https://github.com/MobSF/Mobile-Security-Framework-MobSF
[Accessed 22 March 2021].
22. Wikipedia The Free Encyclopaedia. [No Date]. Available at: https://en.wikipedia.org/wiki/Port_scanner
[Accessed 24 March 2021].

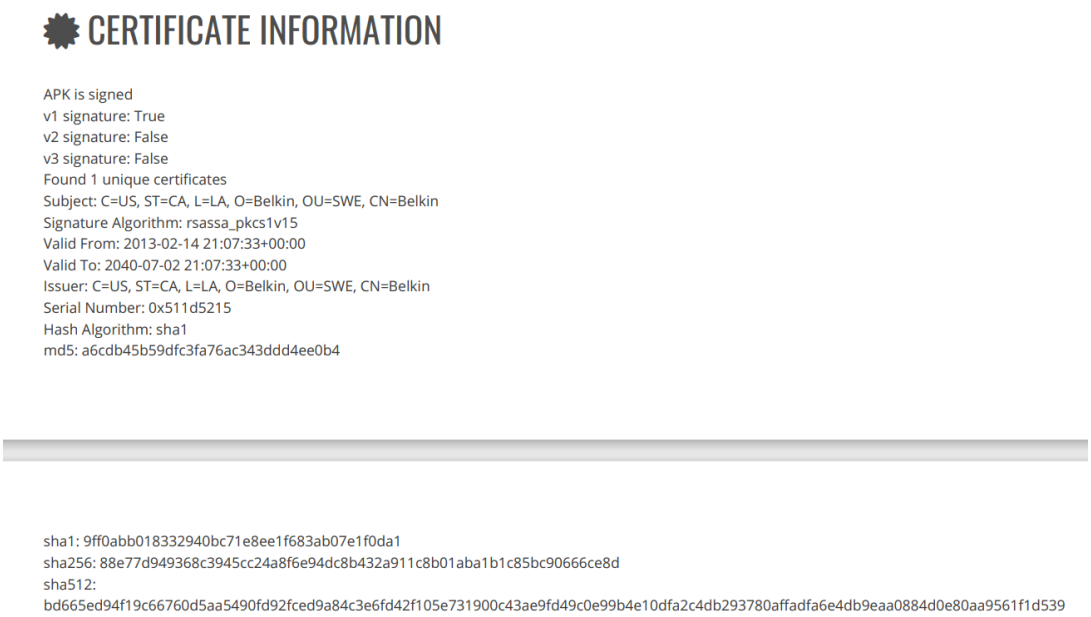
23. Lui, H. 2019. Uncovering Security Vulnerabilities in the Belkin WeMo Home Automation Ecosystem. Available at: https://homepages.inf.ed.ac.uk/ppatras/pub/sptiot19.pdf
[Accessed 26 March 2021].
24. Nmap. [No Date]. TCP SYN (Stealth) Scan (-sS). Available at: https://nmap.org/book/synscan.html
[Accessed 14 April 2021].
25. Cigniti Technologies. [No Date]. Which are the best mobile testing tools – Simulators, Emulators, or Real devices? Available at: https://www.cigniti.com/blog/mobile-testing-with-simulators-emulators-physical-devices/#:~:text=Virtual%20testing%20involves%20testing%20the,the%20applications%20to%20be%20tested
[Accessed 20 April 2021].

Table of Abbreviations

Abbreviation	Definition
IoT	Internet of Things
RFID	Radio-Frequency Identification
DDoS	Distributed Denial of Service
RF	Radio Frequency
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
IP	Internet Protocol
OWASP	Open Web Application Security Project
ARP	Address Resolution Protocol
WPA	Wi-Fi Protected Access
STRIDE	Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege
DREAD	Damage Potential, Reproducibility, Exploitability, Affected Users, Discoverability
APK	Android Application Package
SSID	Service Set Identifier
deauth	Deauthentication
Cupp	Common User Password Profiler
SYN	Synchronize
ACK	Acknowledgement

Appendices

Appendix A - WeMo Mobile Application Certificate Information



Appendix B - WeMo Mobile Application Possible Vulnerable Source Code Locations

b/a/q/g/e/a.java
b/a/q/t/a.java
b/a/q/o/k/b.java
b/a/g/a.java
b/a/g/b.java
com/localytics/android/MarketingProvider.java
b/a/q/g/a.java
b/a/q/o/k/c.java
b/a/p/j.java
com/localytics/android/ProfileProvider.java
com/localytics/android/BaseProvider.java
com/localytics/android/MigrationDatabaseHelper.java
com/localytics/android/AnalyticsProvider.java
ch/qos/logback/classic/android/SQLiteAppender.java
b/a/q/g/b.java
b/a/m/b/a.java

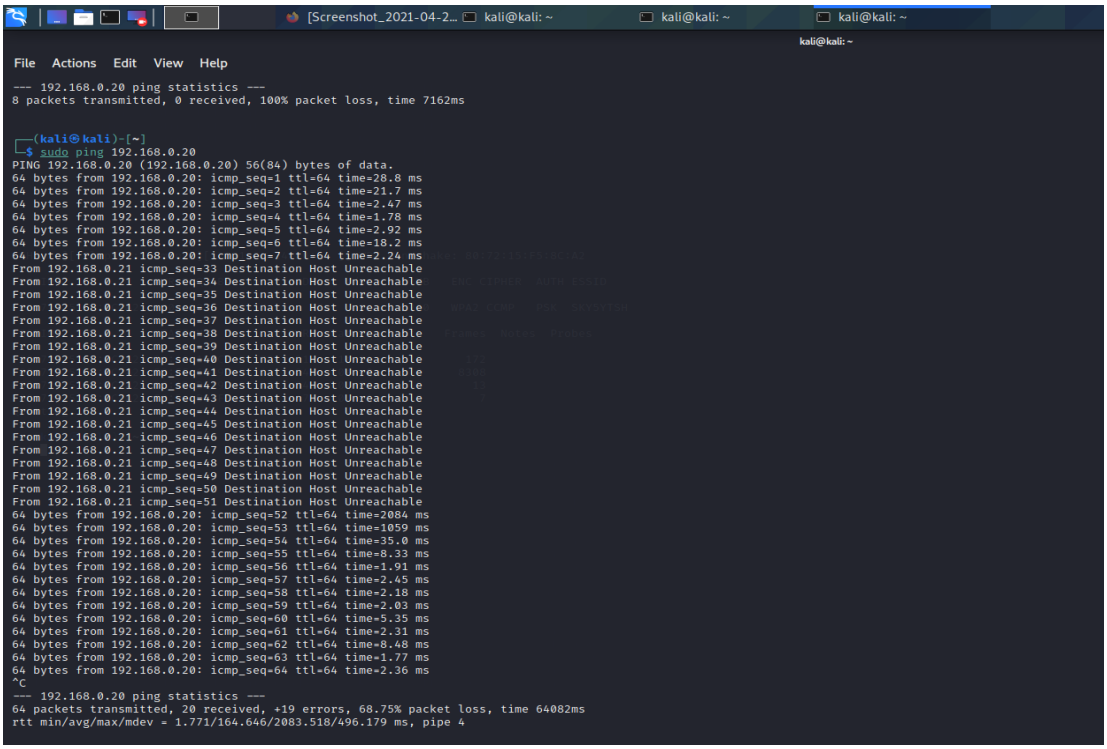
io/liteglue/SQLiteAndroidDatabase.java
com/localytics/android/Datapoint Helper.java
c/a/a/e.java
ch/qos/logback/core/net/DefaultSocketConnector.java
b/a/m/c/a.java
com/localytics/android/LocalyticsManager.java
com/localytics/android/MarketingDownloader.java
a/i/a/b.java
a/d/l/f.java
a/d/l/s.java
b/a/g/a.java
b/b/a/z.java
com/belkin/cordova/plugin/SetupPlugin.java
c/a/a/g.java
b/a/q/g/h/i.java
com/localytics/android/BaseUploadThread.java
a/j/a/a.java
c/a/a/h.java
a/d/l/z.java
a/d/l/b.java
com/belkin/cordova/plugin/AndroidPreferences.java
a/d/e/e.java
b/a/o/a/k.java
com/belkin/cordova/plugin/WeMoSMARTUIPlugin.java
b/a/q/l/d/c.java b/a/g/b.java
com/localytics/android/Marketing Provider.java
a/d/e/c.java
a/d/l/r.java
ch/qos/logback/classic/net/SimpleSocketServer.java
b/a/f/a/d.java
com/localytics/android/MarketingCondition.java
b/a/k/e/b/b/a.java
a/a/k/a/a.java
b/a/f/a/c.java
b/a/o/a/j.java
a/d/e/g.java
b/a/q/g/c/l.java
ch/qos/logback/classic/spi/ThrowableProxy.java
b/a/o/a/e0.java
a/d/e/j.java
com/localytics/android/MarketingRulesAdapter.java
com/localytics/android/AnalyticsHandler.java
com/localytics/android/MarketingDialogFragment.java
com/localytics/android/Localytics.java

a/k/a/a/h.java
com/localytics/android/ProfileProvider.java
b/a/q/g/h/m.java
b/a/o/a/d.java c/a/d/q/b.java
com/localytics/android/PushReceiver.java
com/localytics/android/BaseHandler.java
b/a/o/a/e.java
a/a/n/g.java
com/localytics/android/BaseProvi der.java
com/localytics/android/MigrationDatabaseHelper.java
com/localytics/android/JavaScript Client.java
b/a/p/d.java
com/belkin/cordova/plugin/AuthC odeGenerator.java
a/d/l/a0/b.java
b/a/q/w/f.java
com/localytics/android/ProfileHandler.java
b/a/q/g/d/g.java
b/a/o/a/y.java
b/a/l/c.java
b/a/p/e.java
b/a/m/b/c.java
com/belkin/cordova/plugin/StartApp.java
c/a/d/f.java
com/localytics/android/Analytics Provider.java
a/d/l/h.java
ch/qos/logback/classic/android/LogcatAppender.java
b/a/q/g/c/v1.java
com/almworks/sqlite4java/SQLite.java
com/belkin/cordova/plugin/NativeUtilPlugin.java
com/soundcloud/android/crop/e.java
org/slf4j/helpers/Util.java
b/a/p/g.java
com/belkin/cordova/plugin/DevicePlugin.java
b/a/q/l/d/a.java
b/a/o/a/s.java
com/splunkmint/SplunkMint.java
b/a/q/g/g/a.java
com/belkin/wemo/push/service/FCMMessagingService.java
ch/qos/logback/classic/pattern/TargetLengthBasedClassNameAbbre viator.java
a/d/e/f.java
com/localytics/android/TestModeListView.java
b/a/p/f.java
ch/qos/logback/core/subst/Node.java
b/a/o/a/b.java

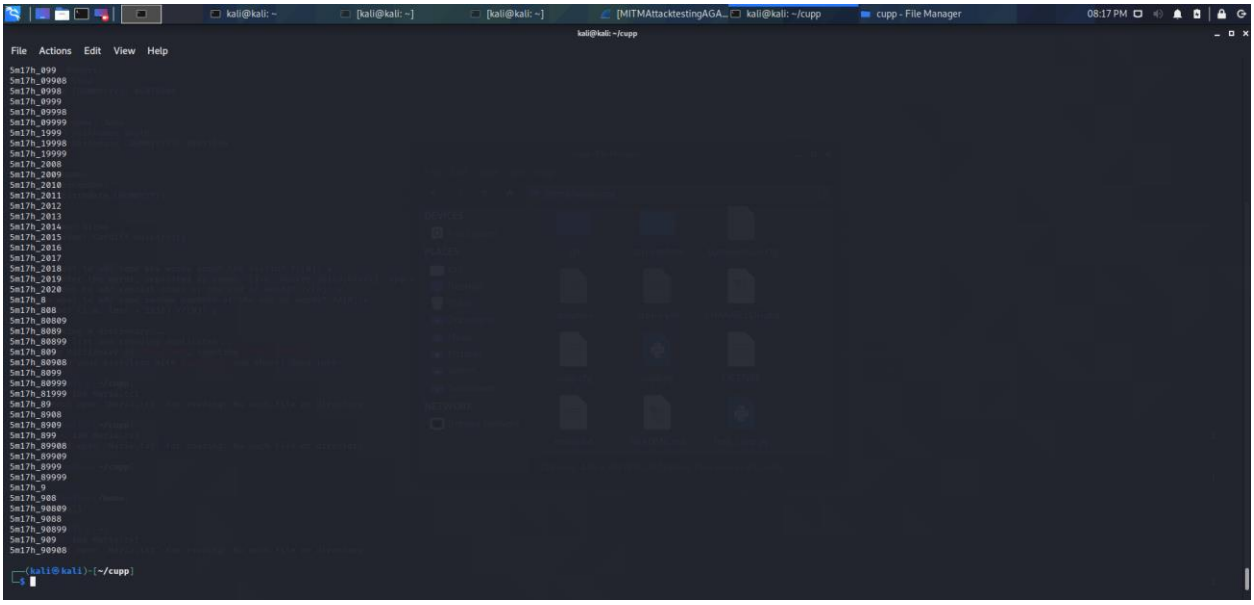
a/d/k/b.java
com/belkin/cordova/plugin/AccountPlugin.java
ch/qos/logback/core/net/SocketConnectorBase.java
com/localytics/android/MarketingHandler.java
com/belkin/cordova/plugin/HttpWrapperPlugin.java
b/a/c/a.java
b/a/q/g/d/p/p.java
a/d/e/k.java
a/h/a/a.java
com/localytics/android/TestModeButton.java
com/localytics/android/ReferralUploader.java
a/d/l/u.java
c/a/d/q/d.java
c/a/d/e.java
com/belkin/wemo/cache/data/DeviceInformation.java
c/a/d/q/f.java
b/a/q/g/h/n.java
com/fasterxml/jackson/databind/cfg/PackageVersion.java
b/a/q/g/c/s.java
c/a/d/q/j.java
c/a/d/q/h.java
b/a/q/g/h/u.java
b/a/q/n/f/b/a.java
b/b/a/n0.java
b/a/p/q.java
com/belkin/cordova/plugin/AuthCodeGenerator.java
c/b/a/a/a/w/t/d.java
b/a/q/g/h/n.java
b/a/q/g/c/g.java
com/belkin/cordova/plugin/WebServicePlugin.java
b/b/a/p0.java
b/a/q/t/b.java
net/lingala/zip4j/crypto/AESEncrpyter.java
b/a/j/b.java
c/b/a/a/a/w/t/c.java
net/lingala/zip4j/crypto/StandardEncrypter.java
b/b/a/p0.java
b/a/q/t/b.java
ch/qos/logback/core/android/AndroidContextUtil.java
b/a/p/j.java
b/a/q/o/u/c.java
b/a/q/n/f/b/a.java
com/localytics/android/JsonObjects.java
ch/qos/logback/classic/sift/ContextBasedDiscriminator.java

ch/qos/logback/core/rolling/helper/IntegerTokenConverter.java
com/belkin/wemo/cache/data/DeviceInformation.java
ch/qos/logback/classic/joran/action/ConfigurationAction.java
com/localytics/android/Constants.java
ch/qos/logback/core/net/ssl/SSL.java
com/localytics/android/MigrationDatabaseHelper.java
ch/qos/logback/core/rolling/helper/DateTokenConverter.java
com/localytics/android/ProfileHandler.java
ch/qos/logback/core/CoreConstants.java
com/localytics/android/AnalyticsProvider.java
com/soundcloud/android/crop/b.java
b/b/a/n0.java
b/a/p/q.java
b/a/q/n/c/a.java
b/b/a/p0.java

Appendix C - Ping Scan during Deauthentication Attack showing the WeMo Smart Plug as an unreachable host

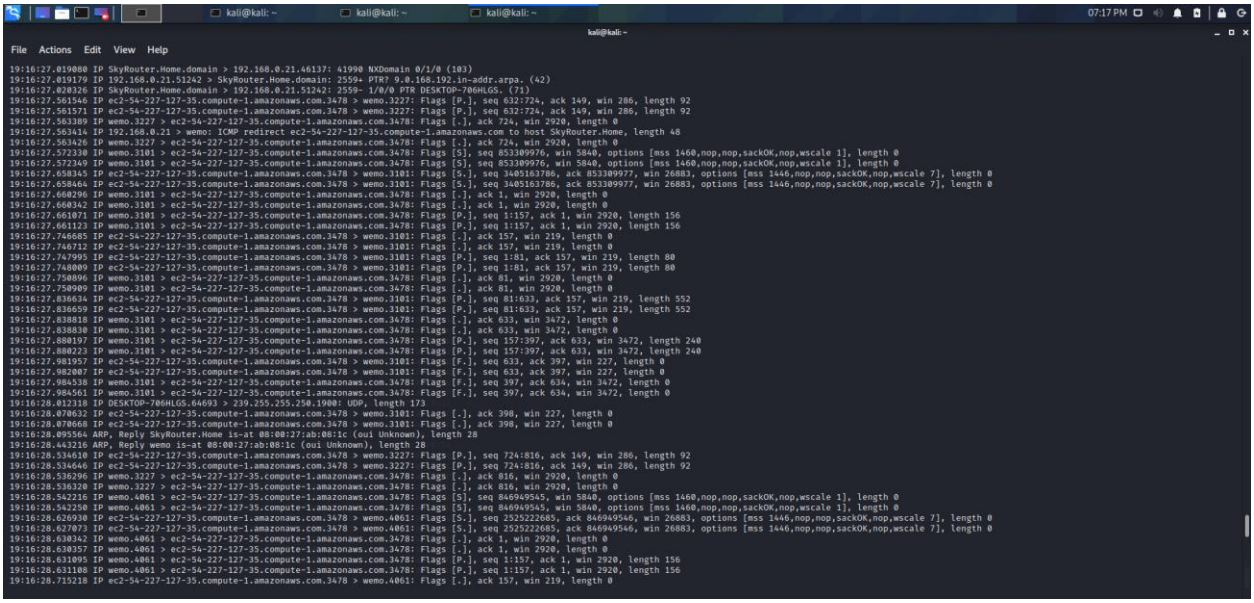


Appendix D - List of Generated Passphrases with Cupp

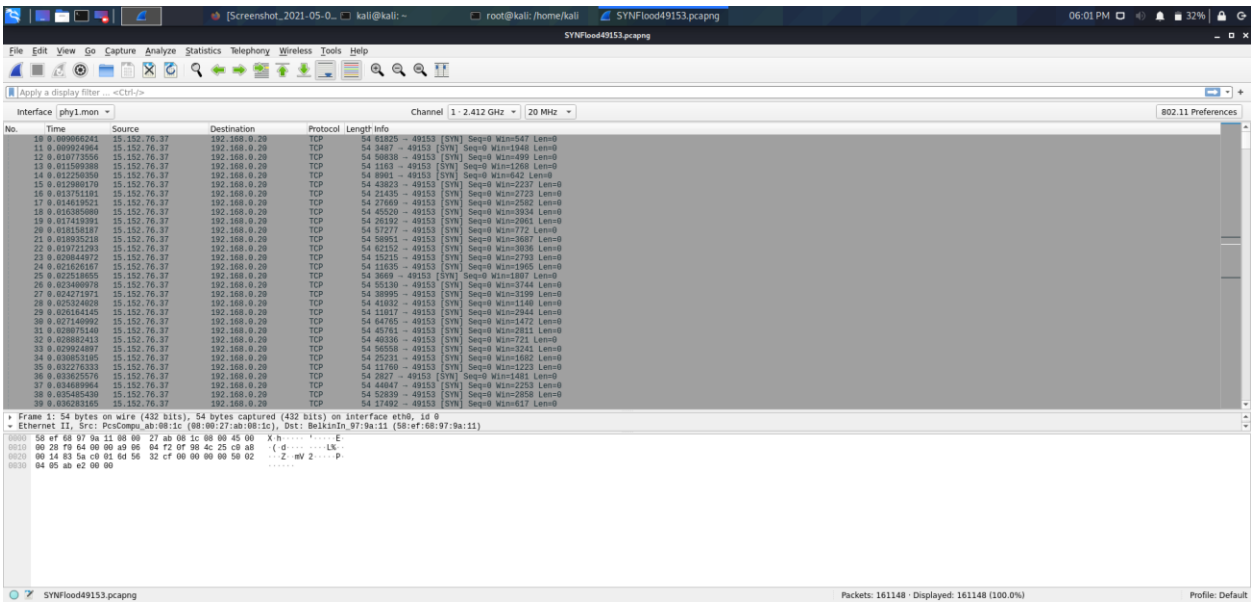


Appendix E - Using Aircrack-ng and Dictionary Generated using Cupp In An Attempt To Crack Network Password

Appendix F - Packets Sniffed with Tcpdump during ARP Spoofing Attack

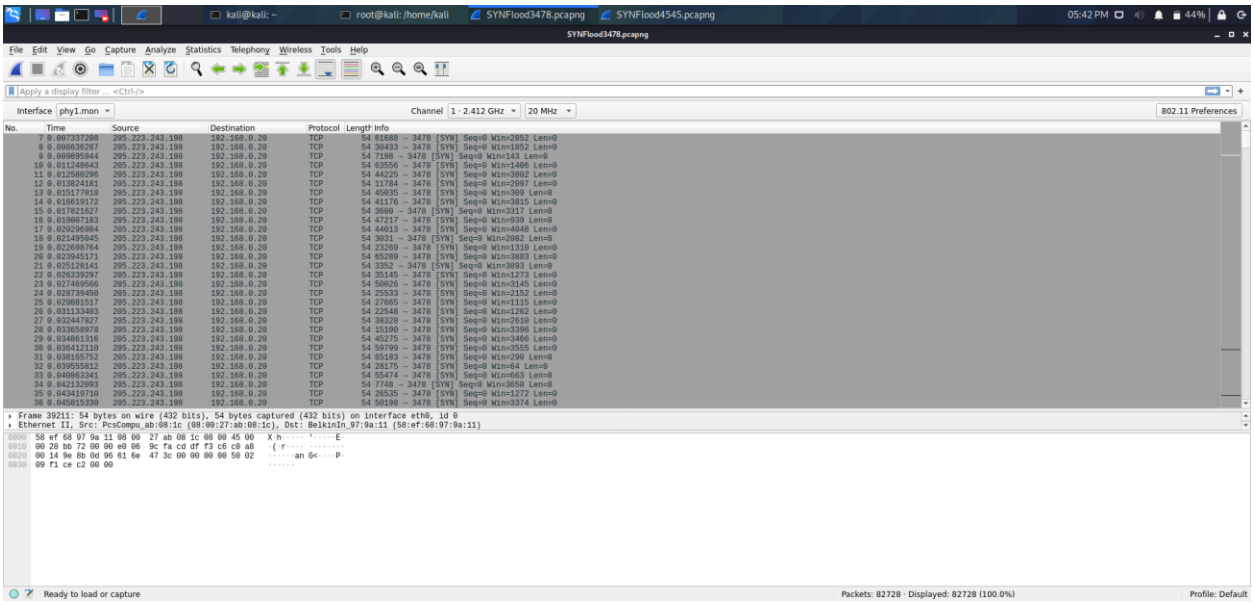


Appendix G - Attacking Port 49153 with a SYN Flood Attack

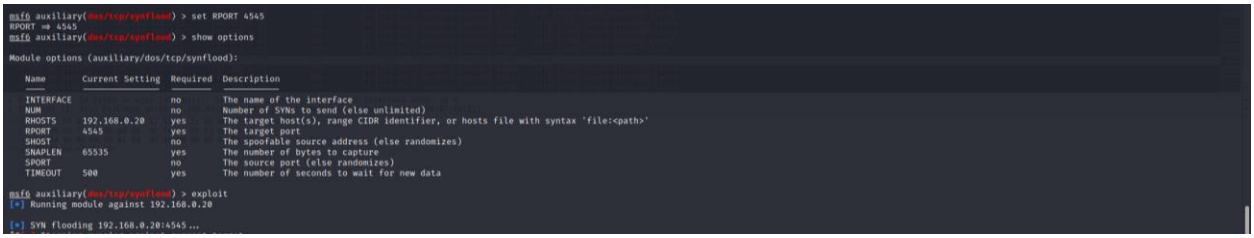


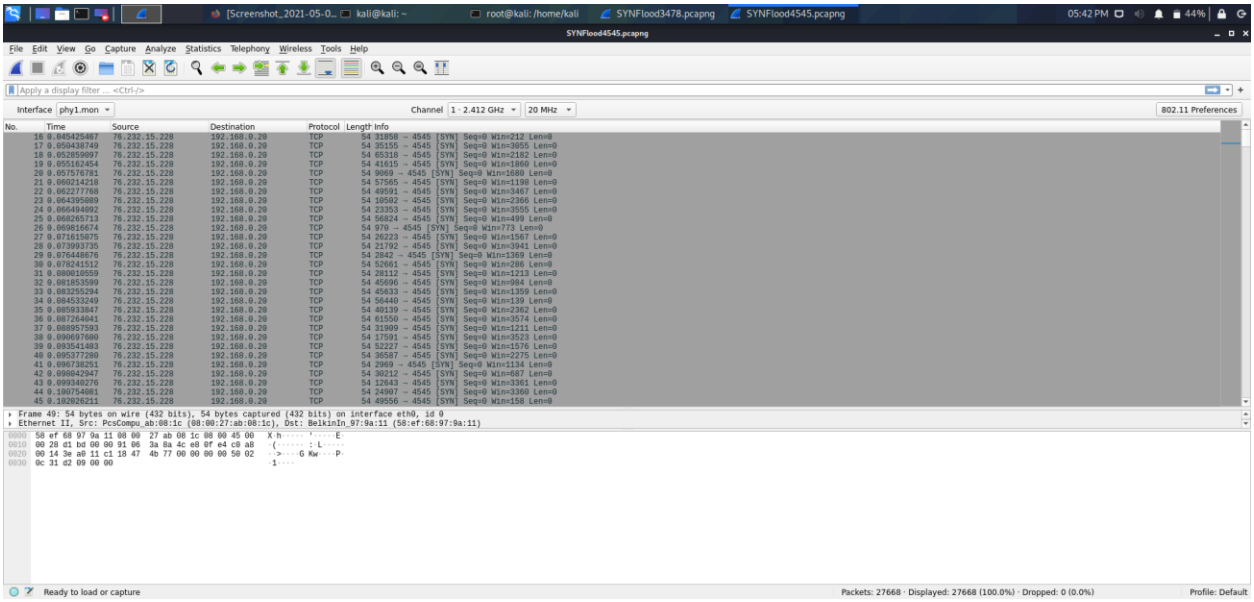
Appendix H - Attacking Port 3478 with a SYN Flood Attack

```
msf6 auxiliary(dos/tcp/synflood) > set RPORT 3478
RPORT => 3478
msf6 auxiliary(dos/tcp/synflood) > show options

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

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   INTERFACE                   no        The name of the interface
   NUM                         no        Number of SYNs to send (else unlimited)
   RHOSTS     192.168.0.20     yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT      3478             yes       The target port
   SHOST                       no        The spoofable source address (else randomizes)
   SNAPLEN    65535            yes       The number of bytes to capture
   SPORT                       no        The source port (else randomizes)
   TIMEOUT    500              yes       The number of seconds to wait for new data

msf6 auxiliary(dos/tcp/synflood) > exploit
[*] Running module against 192.168.0.20

[*] SYN flooding 192.168.0.20:3478 ...
```



Appendix I - Attacking Port 4545 with a SYN Flood Attack





Appendix J – WeMo Device Showing Flashing Orange LED During Deauthentication Attack


Appendix K – WeMo Application Showing Device Not Detected During Deauthentication Attack