Design and implementation of a firewall system for working in unsafe environments

Master's Thesis submitted to the
Faculty of Informatics of the *Università della Svizzera Italiana*
in partial fulfillment of the requirements for the degree of
Master of Financial Technology and Computing
main track

presented by
Bin Yong

under the supervision of
Prof. Student's Advisor
co-supervised by
Prof. Student's Co-Advisor

February 2024

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Bin Yong
Lugano, 2 February 2024

Abstract

Design and implementation of a firewall system based on Raspberry Pi. The firewall monitors network activities, encrypts some plaintext traffic, and alerts invalid certificates. It helps device owners take back control of their own devices. It is designed for someone who needs to work in unsafe environments. The system sacrificed some compatibility to pursue better security but still achieved a balance between security and convenience. Decisions are made by the combination of the analyzation to security events and my past experience. A sensitive target, like an investigative journalist, could be a potential user of this device. The firewall showed success protection against Domain Name System (DNS) hijacking, man-in-the-middle attacks, and port scanning. It successfully blocked some management tools, and decreased the data visibility to eavesdropping attacks.

Acknowledgements

Contents

Figures

Tables

Chapter 1

Introduction

With the development of surveillance and management tools, people are losing control of their own devices. Here are some facts that the general public may underestimated the impacts of their combinations:

1. Any device with a microphone (sometimes a speaker may do the same) can 'hear' sounds people are making around.
2. Any device with a camera can 'see' what is happening around.
3. Any device connected to internet can upload what they know and receive commands from remote.
4. Any device may have been hacked and functioning as not what they were designed.
5. Some devices are designed with surveillance functionality.

List 1.1. Some facts to show the possibilities of surveillance

Based on the facts above, many devices could be used as surveillance tools. For example, smartphones and smart speakers are both able to hear voices and communicate with the internet. Most people do not have the skills to investigate and monitor what they actually uploaded to a remote server. To be able to respond to "Hi, Siri." or "Hey, Alexa.", devices have to process every sound they receive when they are quiet. Some are even worse: Intel Management Engine (Intel ME) or Intel Converged Security and Management Engine (Intel CSME) and Intel Active Management Technology (Intel AMT) for example. Obviously, they are made for management, a function that is unnecessary to most individual owners. According to their official documents. "even when the platform is powered off, as long as the platform is connected to power and to a network" [Corporation, 2022] means that simply powering off the computer could not stop this kind of management. "up before the main operating system" [Corporation, 2023] means that most users could not even notice it and could do nothing about it, and if something goes wrong, people cannot fix it by simply reinstalling operating system. It can access "LAN/Wireless LAN" [Corporation, 2022], check the facts. "present on most Intel platforms, including client consumer and commercial systems, workstations, servers, and IoT (Internet of Things) products."

[Corporation, 2022] means there is almost no escape in their ecosystem. The details of its documents also show the capability to tunnel remote commands (Serial Over Local-Area Network [Corporation, 2022]), the capability of accessing local storage "Universal Serial Bus Redirect" [Corporation, 2022], "Keyboard, Video and Mouse" [Corporation, 2022] remote control over network. Besides, A person may accept rules of the management functionality, but the criminal will not tell the person when the functionality is abused.

So many things are out of the control of the owners of the devices. People need something to restrict the management and to show what is happening to the network. The goal of this work is to help with that. It aims to increase the chance of survival of the user from hacking, eavesdropping, digital fingerprinting, and unwanted managements. It uses a screen to display selected real-time network activities. The system limits network activities and applies encryption standards.

Using hardware firewalls has several advantages over software firewalls. A hardware firewall is a physical device that plays the role of firewall, while software firewalls are usually software components that run inside a user's computer which also works for other purpose. Firstly, a hardware firewall runs completely independent to high-risk user devices, where threat thrives: CPU with possible hidden instruments, firmware with possible infections, motherboards with doubtable proprietary management technologies, and applications with possible surveillance or unpublished vulnerabilities. Second, it brings convenience to the user: the configuration of this portable device restricts everything behind it, so people do not need to do the time-consuming configuration work on different software and operating systems repeatedly. Third, it brings possibilities: Distrusting a built-in root certificate on mobile devices is hard and the methods with root or jailbreak involved may void the warranty. With the help of a firewall, managing certificates is no longer a problem.

This paragraph introduces the structure of this paper. In chapter 2, The decisions behind the hardware selection and the operating system selection are explained, and the cost comparison to commercial devices is made. It proved the low-cost of this system. The general decisions about how packets are forwarded are included in chapter 3. In chapters 4 to 6, the deficiency of the network protocols that are being used or allowed are discussed and the counter measurements are provided. In chapter 7, we discuss how the firewall handle and process plaintext and other protocols. In chapter 8, the problem of using Network Time Protocol (NTP) is discussed and a new method to synchronize time between computer and internet is designed and implemented. In chapter 9, a command-line tool to simulate network activities is designed and implemented. It is a tool being shipped with the firewall system but it needs to be executed manually by the user. In chapter 10, the dispatcher, the script to react to changes of the system, is introduced. The chapter 11 explains how the firewall is installed to the base operating system and explains the reasons why some system files are modified. In chapter 14 various attacks are used to test the firewall and the results show the success of protection.

Chapter 2

Hardware and operating system

2.1 Design

"hood" (see appendix B.10) should work as a USB Ethernet device by default, so people do not need to connect many cables. See fig. 2.1. With proper configurations, the device can also work as a wireless access point or a wireless USB Ethernet device. The cost of the hardware should be lower than the average price of a commercial hardware firewall.

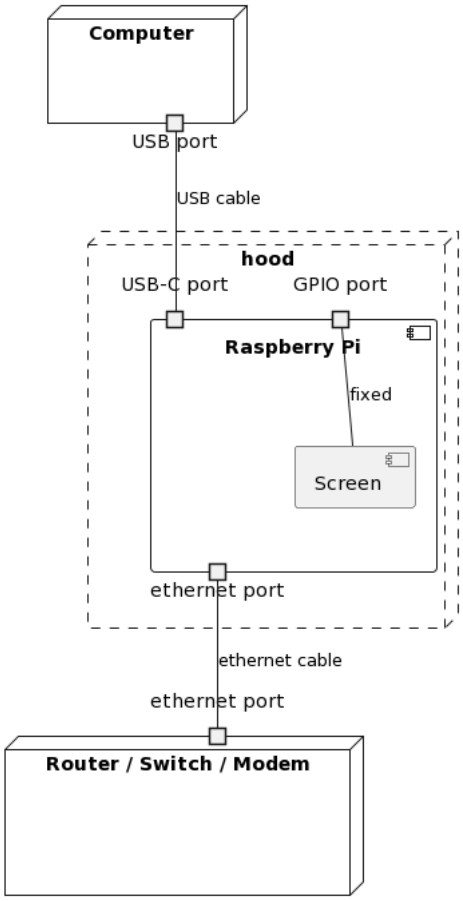

Figure 2.1. Default hardware deployment view

2.2 Implementation

2.2.1 USB device

To make computers recognize the firewall as a USB device, the support from both the hardware and the operating systems (OS) are required.

For hardware, Raspberry Pi 4 is the first model in Raspberry Pi family that both supports USB device mode and contains an Ethernet port. Raspberry Pi 5 can do the same but it is more expensive.

For OS, Linux has USB gadget mode. Using the combination of CDC-ECM (See appendix B.5) and Remote Network Driver Interface Specification (RNDIS, see appendix B.13) mode, most Linux, Windows, BSD, and macOS could be supported. FreeBSD USB stack has a device mode and provides 3 virtual network interface templates but none of them works with Microsoft Windows[Project, 2023b]. OpenBSD has renowned security-focused nature but it does not

contain a device mode in its USB stack. Thus, to support as many devices as possible, Linux is selected as the base OS of this firewall device.

2.2.2 The base hardware

This paragraph explains the reasons of using Raspberry Pi. Since its first release in 2012, Raspberry Pi has upgraded its hardware models and software multiple times, has accumulated a lot of real-world applications, and has maintained an active user community with a large number of experienced users behind it. All of them show that by using Raspberry Pi instead of a cheap new brand, reduced time cost could be expected on fixing the bugs of the computer itself or waiting for the answers from the community because its users have asked and answered many questions and have found and fixed many problems. Besides, its large number of users could also help in spreading the firewall project and help in pointing out security flaws of the new upgrades to the computer.

Joy-IT RB-TFT3.5 was the only screen that I had. The dtbo file (see appendix B.9) of the Waveshare 35A screen also works with this hardware. Thus, the cheaper one in the market should be able to replace another. The resolution of 480x320 is enough for printing logs.

2.3 Cost comparison to the market

2.3.1 Theory of the cost

Commercial firewalls are usually more expensive than development boards that contain the same level of hardware functionality. On one hand, It's acceptable because they provided additional value through their software and services and all of the development and maintenance of the product, the service, and the company itself are costing money. On the other hand, it also indicates that an open-source solution would reduce the lower bound of the cost of accessing a hardware firewall.

2.3.2 Market research on commercial firewalls

Despite the firewall implemented in this paper is not a general-purpose firewall, the difference in cost between market products is still worth knowing. From tables 2.1 and 2.2, the prices of hardware firewalls are concentrated around 250.74 EURO and most of them are more expensive than the price of a Raspberry Pi 4B (68.79 EURO) selling at the same website. The only exception is TP-LINK ER605. Its price is 55.5 EURO which is still higher than the price of a Raspberry P 3B on the same website. After checking the details of this exception, its central management design and the corresponding cloud-based controller both show that it is designed for a totally different threat model. Its exceptional price could be explained as the device is under the control of the company. However, spending money on a hardware firewall usually implies the reluctance to give control of devices to a company. To sum up, using Raspberry Pi would cost less than usual general-purpose firewalls.

Name	Price
TP-LINK ER605 5 Port Dual/Multiple WAN VPN Router (up to 4 Gigabit WAN Ports, Highly Secure, Omada SDN, Central Management, Intelligent Monitoring, Firewall) Black, Ideal for Office Network	€55.49
Micro Firewall Appliance, Mini PC, Pfsense Plus, Mikrotik, OPNsense, VPN, Router PC, Intel N4505, HUNSN RS41k, AES-NI, 4 x 2.5GbE I226-V, Console, Type-C, HDMI, DP, SIM Slot, 4G RAM, 32G SSD	€188.99
ZyXEL ZyWALL 350Mbps VPN Firewall, Recommended for up to 10 Users [USG Flex 50]	€259.00
Micro Firewall Appliance, Mini PC, pFsense Plus, Mikrotik, OPNsense, VPN, Router PC, Intel Alder Lake-N 12th Gen N100, HUNSN RJ42, 4 x 2.5GbE I226-V, 2 x HDMI, DP, TF, Type-C, 8G DDR5 RAM, 128G SSD	€279.99
KingnovyPC Firewall Micro Appliance, 4 Port i226 2.5GbE LAN Fanless Mini PC N5100, 2* DDR4, HDMI, DP, RJ45 COM, 4*USB Gigabit Ethernet AES-NI VPN Router Openwrt Barebone	€146.53
New J4125 Quad Core Firewall Micro Appliance, Mini PC, Nano PC, Router PC with 8G RAM 128G SSD, 4 RJ45 2.5GBE Port AES-NI Compatible with Pfsense OPNsense	€279.00
Cisco Meraki Go - 5 Port Security Gateway - Power Supply to EU Standard GX20-HW-EU	€102.53
Protectli Vault FW4B - 4 Port, Firewall Micro Appliance/Mini PC - Intel Quad Core, AES-NI, 4GB RAM, 32GB mSATA SSD - Compatible with pfSense/OPNsense etc	€284.87
Firewall Hardware, Pfsense, OPNsense, Mikrotik, VPN, Network Security Appliance, Router PC, Intel Atom N2600, HHUNSN RS31, 4 x Intel Gigabit LAN, 2 x USB, COM, VGA, Fan, 4G RAM, 32G SSD	€221.99
Firewall Mini PC with 2.5G Gigabit LAN, Firewall Micro Appliance Celeron J4125,4 x I225 Gigabit LAN,Fanless Mini PC AES-NI,12V,WiFi,HDMI,RS232 COM,USB 3.0,8GB RAM/128GB SSD	€265.00
FORTINET FortiGate 40F Hardware - Next Generation Firewall Protection and Security	€520.90
Micro Firewall Appliance, Mini PC, VPN, Router PC, Intel Alder Lake-N 12th Gen N100, HUNSN RJ42, 4 x 2.5GbE I226-V, 2 x HDMI, DP, TF, Type-C, Barebone, NO RAM, NO Storage, NO System	€288.99
Cisco Systems Go Router Firewall Plus | Cloud Managed | VPN | Cisco [GX50-HW-EU], White	€318.09
Zyxel Secure Cloud-Managed Router/Firewall with AXE5400 Tri-Band WiFi Subscription Free Network Security, Managed via Nebula APP/Ideal for Small Offices/Small Branches. [SCR 50AXE]	€173.5
Micro Firewall Appliance, Mini PC, VPN, Router PC, Intel Alder Lake-N 12th Gen N100, HUNSN RJ46, 6 x 2.5GbE I226-V, 2 x HDMI2.1, TF, Type-C, Barebone, NO RAM, NO Storage, NO System	€242.99
HSIPC 11th Gen i3 1115G4 Firewall Micro Appliance, Mini PC, Nano PC, Router PC (16G 256G) With 6 RJ45 2500M, AES-NI, HDMI USB3.0 Console, Compatible with Pfsense OPNsense	€384.00

Table 2.1. The first page of search results of "hardware firewall" on amazon.de. Captured at 5 P.M. of December 26th, 2023 (UTC +1), Lugano

Statistics of the prices in search results			
Mean	250.74	Median	262.0
Min	55.49	Max	520.89
Standard deviation		109.98	

Table 2.2. Statistics of table 2.1

2.3.3 Conclusion

In conclusion, Raspberry Pi 4B has sufficient hardware and software support to make it work as the USB device designed in the first section. Its price is also much lower than the majority of commercial firewalls. It is not the cheapest option that can fit the design, but it is a good choice after evaluating technical support and user base.

Chapter 3

Packet filtering and forwarding in general

3.1 Design

As the general principle, all traffic passed through firewall should be checked against eavesdropping. Plaintext protocols should be channeled through strong encryptions or discarded if unable to ensure their integrity. Firewall rules should filter out as much as possible and as early as possible because everything on the path of a packet is potentially vulnerable, and the less network traffic passed to the following components, the less chance a vulnerability can affect the firewall.

Network traffic should be forwarded by proxies instead of kernel. The following content explains the advantages of doing so. First, a proxy can run at the user level. When there is a vulnerability, a userland one naturally be less harmful than a kernel one. Second, proxies are easier and also safer to configure. It's hard to configure operating system firewalls well. When the rules are not strict enough, unexpected things may happen. For example, allowing any connection to 127.0.0.1 to succeed without a log, could cause potential problems if a malicious Dynamic Host Configuration Protocol (DHCP, see chapter 4) server assigned the address 127.0.0.1 to a physical network interface. Also, simply filtering packets by port numbers, addresses, and interfaces could not guarantee that allowed packets are used by the desired protocol. An attacker can simply let the receiving end of a reverse shell listen to an allowed port to bypass this kind of defense. By contrast, a proxy only works for the protocols it can understand. For example, the chance of the mistakes in configurations of a DNS proxy enabling HTTP requests to pass through that proxy is very rare. Attackers have to hack the firewall or create tunnels over those protocols to make their reverse shell bypass the firewall, which makes their operations more difficult. Third, the packets from the proxy are encoded again by the Linux network stack. Re-encoded packets have only Linux TCP and IP signatures, rendering fingerprinting techniques that target TCP and IP properties useless.

3.2 Implementation

The balance between compatibility and security: To maximize compatibility, the firewall should allow everything to pass through it, regardless of protocol and destination, but to maximize security, nothing should be able to pass through the firewall. As a balance, modern browsers can satisfy most of the needs for working. So, allowing only HTTP and HTTPS connections could make most of the work done when facing a hostile network environment.

Nftables (see appendix B.11) is integrated into modern Linux systems. It can classify and filter network traffic. The firewall uses it to filter network packets. UDP 67 or 68 port and ARP packets are allowed for assigning IP addresses. UDP 53 is allowed for users to make DNS requests to the firewall device, TCP 80, and TCP 443 are allowed for HTTP and HTTPS traffic. All other packets are filtered as early as possible.

Different proxies and services are created to check connections, to forward traffic, and to apply encryptions. Details are included in the chapters of corresponding protocols. See figs. 6.2, 7.1 and 7.2. Several configuration files are created for blocking IP subnets and domain names, they are used by proxies and services. See appendices A.3 to A.5.

Chapter 4

Dynamic Host Configuration Protocol (DHCP)

4.1 Introduction

DHCP is widely used to dynamically assign an IP address to a new device connected to a network.

4.1.1 The issue

The issue is that information leaked by protocol can be used to identify the owner of the device. The host name of the device could be leaked to the network from the 'client identifier' option or the 'sname' field in the protocol.[Droms, 1997] If the name of a computer is straightforward enough, like "Yongbin's MacBook Pro", the network administrator will be able to know the name of the owner of the computer at first glance. The hardware address exposed to the network can also reveal more information than just the address. Hardware address or Media Access Control (MAC) address can be used to reverse lookup the manufacturer and the manufacturer can help identify the owner. For example, the network administrator may use the prefix of a MAC address to find out that the manufacturer of the device is company A. If company A only sells its products in Country B and C is the only person who had ever been to Country B then the chance that the device belongs to C would be high.

4.2 Design

Firewall should randomize both host name and MAC address. The randomized computer name should look normal to prevent being spotted easily.

4.3 Implementation

NetworkManager is used to manage physical connections of the firewall. The MAC address randomization is done by NetworkManager by adding 'ethernet.cloned-mac-address=random'

and 'wifi.cloned-mac-address=random' to the 'connection' section of NetworkManager.conf. The 'ifconfig interface link random' command can also randomize the MAC address of an interface manually.

Host names of the firewall device are randomly generated during the startup of the system. The relevant code is inside rc.local. The generated host name is crafted to look normal. The pattern of default computer names of Microsoft Windows and common names like iPad are used. Because the implementation of the DHCP client could also be vulnerable, AppArmor (see appendix B.4) rules are applied to dhclient (see appendix B.6) to mitigate this problem.

Dnsmasq is used to assign the IP address of the users of the firewall it also tells the devices behind the firewall to use the firewall as the DNS resolver.

Chapter 5

Domain Name System (DNS)

5.1 Introduction

DNS is a protocol that translates the human-readable domain names to the IP addresses of the remote server. The default configurations of many operating systems are to make plaintext requests, which could easily be monitored by simply recording the packets and could easily be attacked by methods like DNS spoofing.

5.2 Design

From the network administrator's view, all DNS queries sent out from the firewall should be encrypted. From the user's view, nothing should need be specially configured.

5.3 Implementation

5.3.1 Outgoing requests

The firewall system created hood-name-service.py to resolve domain name queries from other firewall services. It is a local remote procedure call (RPC) service that resolves domain name queries via public DNS-over-HTTPS (DoH) services. DoH can mitigate both passive surveillance and DNS spoofing attacks[Hoffman and McManus, 2018]. DNS-over-TLS (DoT) is another DNS encryption standard that can do almost the same as DoH. The advantage of DoH over DoT is that DoT uses a unique server port number, 853, that can be easily filtered or identified as an encrypted DNS connection, while DoH uses the same port number and protocol as HTTPS, which makes it much harder to be identified or filtered.

5.3.2 Inside firewall

A dnsmasq is started locally to do DNS hijacking to make the firewall proxies system-wide.

5.3.3 Incoming requests

A dnsmasq is started to do DNS hijacking to make the devices behind the firewall use the firewall proxies.

Chapter 6

Transport Layer Security (TLS)

6.1 Management of certificates

6.1.1 Introduction

Modern major operating systems manage a collection of selected trusted certificates issued by certificate authorities (CAs). Those certificates are used to prove the validity of a public key. When making a TLS connection, local applications check the path of the certificate provided by remote with a local trusted collection to ensure secure connections.

The issue

The problem is that most of TLS clients do not alert their users when the remote website presents a different certificate, which gives CAs the power to do man-in-the-middle (MITM) attacks. Even if a CA has no intention to do evil, its private key could still have been stolen. Thus, none of them are strictly trustworthy. However, the whole TLS is based on it, if we trust none of those authorities, there will be almost no website we can use and without TLS, things will only be worse.

Exsting solutions

Certificate pinning can prevent MITM attacks from a trusted CA. When the user knows the CA of the remote endpoint is A, if another CA does the MITM attack, the client could know that the certificate received is not the correct one because it is not issued by A. However, configuring certificate pinning to all the endpoints is hard and time-consuming for average users, and without a basic trusted environment, people are unable to know whether the configuration has been done correctly. Even if the configuration is correct, it is still not strange for an endpoint to switch to another CA. Thus, to a firewall, this technique cannot be used as a general solution to the issue.

Removing suspicious CAs from the trusted list could protect people from MITM attacks initiated by those CAs. However, some systems and devices do provide the function of distrusting a CA. For example, an iPhone needs a computer to enable such configuration and most Android devices do not provide such entry in settings.

6.1.2 Design

The firewall should trust only the CAs that are being widely used on the internet. It now respects th usage statistics of SSL certificate authorities for websites published by W3Techs [W3Techs, 2024] with some exceptions added for my work. People should be able to further decide which certificates to trust. Consider distrust following CAs when making such decisions: the CAs of the place you are staying, the CAs of the place you come from, and their enemies and allies.

6.1.3 Implementation

Since SSL pinning can only be used on a small number of endpoints by an advanced user who can clearly know what is this meant to be, it is not implemented now. Manually management of the certificate can be done by Linux commands. The certificates of all remote TLS endpoints are checked by TLS proxy.

6.2 Mismatch problem

6.2.1 Problem

The mismatch problem means the mismatch between the changing best current practice and the unchanged implementation. It occures whenever a public protocol standard updated to fix a problem. There are delays between the time of a deficiency of a protocol being published and the time when the majority of protocol implementations fixed the problem. A living example is the latest version of Python 3.12.1 released on December 2023, as part of its built-in libraries, the implementation of the function create_default_context() from the SSL library is still creating a context that supports TLS 1.0 and TLS 1.1 by default [Foundation, 2023], while they have already been deprecated by the best practice rfc8996 since March 2021 [Moriarty and Farrell, 2021].

6.2.2 Solution

An idea to save the clients that contain severe deficiencies is to attack their deficiencies and then hijack and channel their communication to remote endpoints through the TLS connection with the current best practices applied. Applying this methodology to solve the downgrade attack of Secure Socket Layer (SSL) version 2.0 was the initial motivation of the works on the TLS proxy, but due to the drastic changes to the protocol since SSL 2.0, the final implementation is instead to use the proxy to check whether the best practice is applied.

6.3 Proxy

Design

The proxy should filter out the connections that are negotiated to use cipher suites not recommended by RFC best practice by checking the handshake packets. It should also filter out the connections that are using certificates signed by CAs that are not trusted by the firewall

system. So that even if the user device has a pre-installed malicious certificate, the certificate will be filtered by firewall and the MITM attack will fail.

The proxy should not use the MITM attack to decrypt SSL sessions. Despite decrypting SSL sessions to inspect encrypted connections has been used in firewalls and proxies for many years [FORTINET, 2021], the TLS proxy used in this firewall should not implement the same functionality. The reason is that when the firewall itself is compromised, the ability to decrypt TLS sessions will cause a disaster to the devices behind it: the attacker will be able to easily modify the encrypted data transferred over a MITM proxy while the devices behind the firewall will alert nothing about the modification. However, if the firewall could not do MITM attacks at all, the compromised firewall is still unable to see and modify encryptions, because the certificate verification of the devices behind this firewall is still working.

Implementation

Since TLS proxy is in charge of processing and forwarding the majority of the traffic of the firewall, c++ is used to develop the proxy due to its high-performance capability.

This paragraph explains how the proxy filters certificates. The key_share extension sometimes makes the certificate invisible to the proxy. Thus, to filter certificates, the proxy handshakes with the remote server with a separate connection to verify the certificate used by the remote host. A cache of trusted endpoints is used to reduce delays caused by certificate check connections.

fig. 6.1 shows how hood-tls-proxy receives and processes TLS connections.

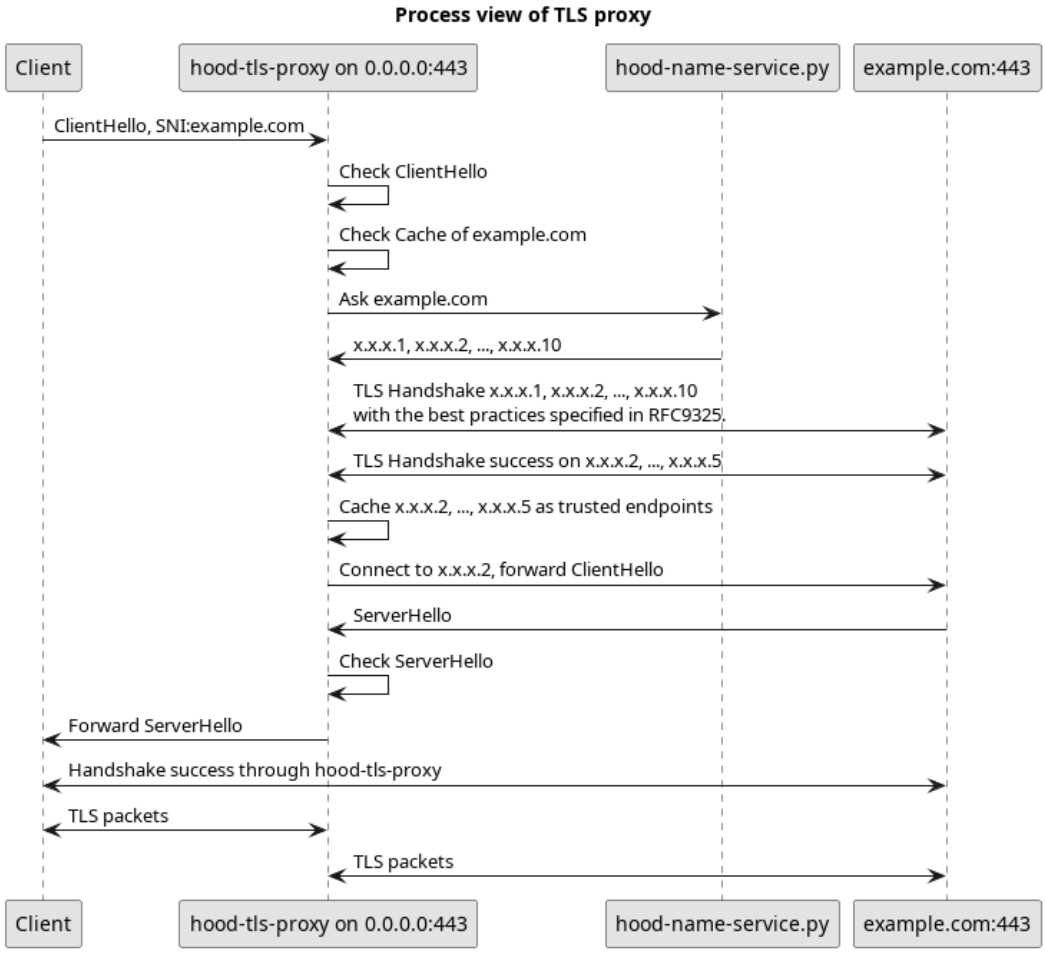

Figure 6.1. Process view of TLS proxy

This paragraph explains how the proxy checks the ClientHello of TLS handshake. Upon receiving ClientHello from a client, the proxy first checks the version of the protocol, and the supported_versions extension to see if the client supports establishing a safe TLS channel, TLS 1.1 and TLS 1.2 are the only allowed versions [Sheffer et al., 2022], if a ClientHello shows that the client supports neither, it will be discarded. After the version check, the proxy checks the ciphersuites field, if it contains none of the values listed in list 6.1, it will be considered unable to be safe and discarded [Sheffer et al., 2022]. If a ClientHello passed all of the security checks, the proxy will extract the Server Name Indication (SNI) from the ClientHello to determine the actual destination of the request. Strengthen the connection by modifying handshake packets without MITM is not possible because the HMAC of all handshake packets is used for encryption.

- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_AES_128_GCM_SHA256
- TLS_AES_256_GCM_SHA384
- TLS_CHACHA20_POLY1305_SHA256

List 6.1. Allowed cipher suites values

Following content explains how the proxy resolve host names and select trusted endpoints. The proxy first checks if there are any cached trusted endpoints available, if not it uses a RPC call to hood-name-service.py to resolve the domain name. Then TLS handshakes following the same restrictions to the ClientHello check are made to all of the addresses returned from the name service. The handshake process will not only ensure remote endpoint meets the requirements (versions and ciphersuites) of applying the best practices described in rfc9325, it will also check whether the remote endpoint could provide a valid certificate of the hostname from a trusted CA of the firewall.

This paragraph explains how the proxy checks the ServerHello of TLS handshake. After forwarding the ClientHello to the server, a ServerHello will be responded and the proxy will check the final result of the handshake applies to the best practice. If a naughty attacker made the negotiation result unsafe, TLS 1.0 or the cipher suite RSA_WITH_NULL_MD5 for example, the connection will be blocked.

Known issue and solution

A known issue is caused by some maverick servers that prefer some weaker ciphersuites while both ends supports a stronger ciphersuite. This preference makes the TLS connections to those servers will never success because the firewall disallow weak encryption but theoretically a safe connection is possible to be made to the server. To solve this issue, a script named chromium.sh is created. It can detect current installed chrome or chromium browser and run the browser with hardening parameters which include the parameter to disable the support of weak ciphersuites. The remote server's preference to those weak ciphersuites is nullified after the support to the preferred weak ciphersuite is removed from the browser, because the negotiation result from the server should not be choosing a ciphersuite that is not supported by the client.

6.4 Process view of traffic

fig. 6.2 shows how a TLS connection are handled and interacted by different components in the whole firewall system.

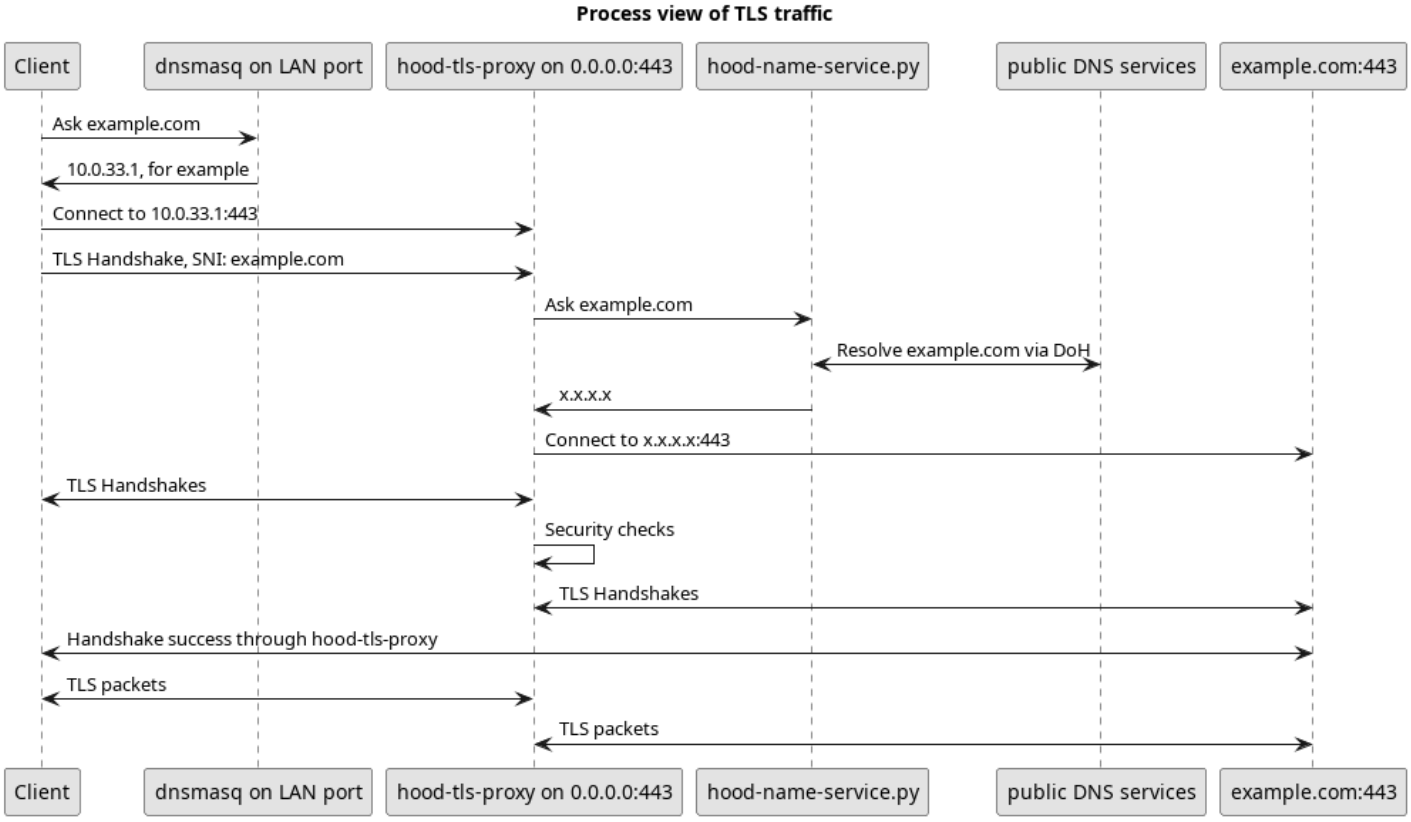

Figure 6.2. Process view of TLS traffic from user

Chapter 7

Plaintext connections and other services

7.1 HTTP

7.1.1 Design

Plaintext HTTP connections should be tunneled and encrpte through TLS proxy, if the remote server does not support encrypted connections, the request should fail.

7.1.2 Implementation

hood-http-handler.py is the proxy created to filter and protect HTTP plaintext connections. The proxy checks the host name with a known list of Online Certificate Status Protocol (OCSP) and Certificate Revocation List (CRL) servers. Only the connections to OCSP servers are allowed to be plaintext because those connections usually be OCSP over HTTP connections, which have no reason to be protected. Figure 7.1 shows how OCSP traffic are received, handled, and forwarded to remote server. The connections to other servers are tunneled via TLS protocol through the hood-tls-proxy service. Figure 7.2 shows how other HTTP traffic are received, handled, and forwarded to remote server.

Figure 7.1. Process view of OCSP over HTTP traffic from user

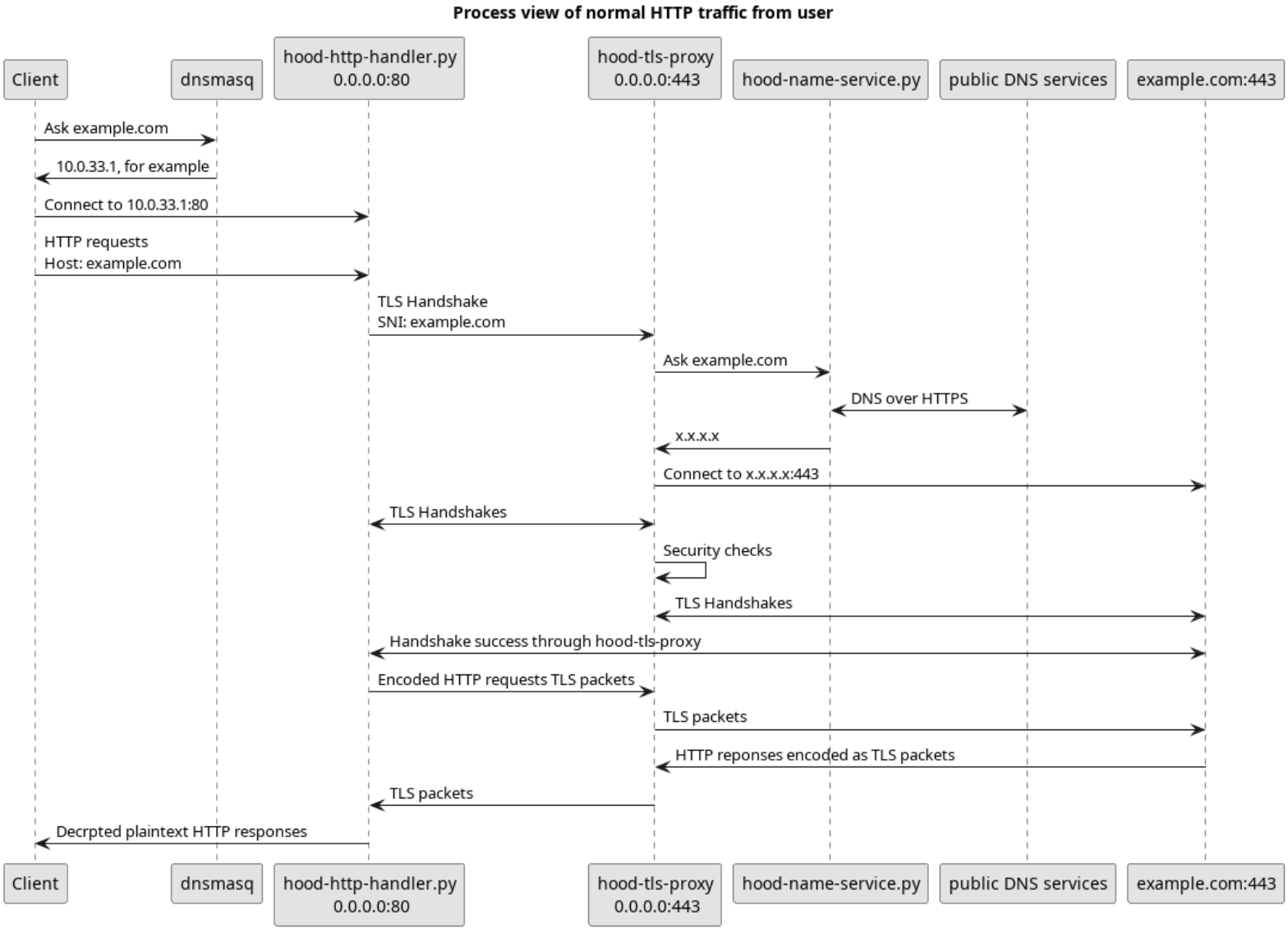

Figure 7.2. Process view of normal HTTP traffic from user

Chapter 8

Date time synchornization

8.1 Introduction

A correct time is required for TLS protocol to check certificates. Network Time Protocol (NTP) is used by most modern systems to synchronize the clock with a remote server.

8.1.1 The issue

A problem with using hardware like a Raspberry Pi is that it does not have a battery to keep the clock ticking after the power source is cut. So a time synchronization mechanism is a must for this firewall.

Many operating systems are pre-configured to use an NTP server which has a unique domain name that can be used by a network administrator to identify the operating systems being used by the client.

Both the unique port number (UDP 123) and the unique domain names (time.windows.com, time.apple.com, and ubuntu.pool.ntp.org) make NTP a protocol that can be easily identified and targeted. Various attacks can be done to the protocol itself[Kwon et al., 2023]. The difference in NTP behavior between different operating systems has been used in OS fingerprinting and tethering detection [Chen et al., 2014]. Not to mention that the operating system information being exposed by an NTP domain name can also help an attacker target the NTP client being used by the device.

8.2 Design

Give up on NTP, use HTTP/HTTPS instead. Use the "Date" header from the response of an HTTP server to synchronize date time with acceptable accuracy. To hide from security detection, the attack vectors targeted to HTTP/HTTPS protocol usually do not want to make mistakes in the header. Even if an attacker targeted the method used by the hood firewall, It's hard to say whether this request is made for time synchronization. Thus the date field of an HTTP response is generally trustworthy.

8.3 Implementation

The tool is implemented in a Python script named hood-timesync.py. It accepts two command line arguments as listed in table 8.1. It at first tries to receive an HTTP response through a TLS connection from the domain name specified by the host argument. If it fails, the reason could be the website is down or the system time is too different from the actual time and made a valid certificate unable to pass the validation. Thus, a plaintext connection to the host name provided by the last resort address argument is made as the second attempt. After the second attempt, the tool starts another attempt to receive time from the domain of the host argument via TLS connection, because TLS has less chance to be attacked and because the reason why the first step failed could be the website is temporarily down, the reason could also be the huge time difference made the validation of the certificate failed. In both cases, a HTTPS request to the domain may become possible to succeed after the second attempt. An Epoch time is used as an 'anchor' time. Based on the assumption that time only moves forward, any time before this value will be rejected.

Name	Type	Default	Description
–host	str	www.bing.com	The website to request time from
–last-resort-host	str	1.1.1.1	The website used as the last resort, should be an IP address
–time-anchor	int	1703703361	An Epoch time that being anchored as past. Based on the assumption that time only moves forward, any time before this value will be rejected. Set 0 to disable this test. Example: -time-anchor=$(stat /etc/os-release -c %W)

Table 8.1. Command line arguments of hood-timesync.py

Chapter 9

Network activity simulation

9.1 Introduction

In this chapter, the goal of the firewall changed to leaking fake information instead of preventing the information leakage because the sad truth is that the combination of all efforts on those protocols is not enough when facing a more sophisticated network analysis. For example, the connections to Windows update servers and App Store servers can be used to detect Windows and Apple devices [Chen et al., 2014], but the firewall has no reason to stop those connections. Apart from that, the domain name leaked from TLS handshakes can also tell the eavesdroppers what websites a person is viewing and then may infer what the person is doing. Virtual private networks (VPNs) may prevent that information from being captured by local eavesdroppers but VPNs also have an egress, where could have other eavesdroppers. Thus, when visiting internet websites, the leak of information is inevitable.

9.2 Design

A service to simulate different network activities is designed to solve this issue. The goal is to hide user network activities inside the flood of fake network activities to increase the difficulty of an analyzer to produce useful reports. However, the risk of using this service is to produce a false alarm immediately and make the user blocked for tethering, make the user fired for browsing unrelated websites, etc. To prevent the firewall from being compromised by browser vulnerabilities, browser-based simulation tools, like Puppeteer (see appendix B.12), should not be used for implementation. It should also make plaintext traffic to attract attention from attackers and eavesdroppers to make them less focused on the actual user traffic.

9.3 Implementation

The service is implemented in a Python script named hood-actor.py. It can parse and execute action files. To simulate the behaviors of a human user or a computer service. All implementations inside the script are based on the base modules of Python, to avoid the version management of third-party libraries being involved in the installation process of the firewall.

9.3.1 Action files

Action files are in JSON format. An action file contains an array of objects. All objects in the array have a "type" field to let the parser know how to process it. Supported object types are described in tables 9.1 to 9.3.

Value	Description
properties	Properties of the action
task	Tasks to be done in the action

Table 9.1. "type" field of action file objects

Field	Type	Description
type	string	"properties"
safe_age	integer	Minimum age safe to do this
proper_age	list of two numbers [min, max]	Proper range of age to do this.

Table 9.2. "properties" object of action file

Field	Type	Description
type	string	"task"
name	string	Name of the task, must be unique in current file
before	list of strings	Names of tasks that should run no earlier than this.
after	list of strings	Names of tasks that should run before this.
delay_between_actions	list of two numbers [min, max]	Range of seconds to wait between actions.
actions	list of strings	list hood action script

Table 9.3. "task" object of action file

9.3.2 HoodExecutor

HoodExecutor is a general-purpose multi-thread task executor written in Python. It is part of hood-actor.py It can run and schedule in parallel tasks that have dependency relationships and delay requirements. It has a thread pool of workers to execute tasks and a single thread for scheduling delayed tasks. It is required to simulate the paralleled requests sent out from real-world browsers without a third-party HTTP client based on the Python asynchronous I/O module asyncio. It is also the cornerstone of fulfilling all the execution requirements of the tasks and actions in an action file.

9.3.3 Browser

A fake browser is created to simulate network activities of browsing. It records the cookies specified in the HTTP response headers and sends them out like a real browser. It parses the

HTML content of the web page to extract the resources to be loaded and load them in parallel just like a real browser. It uses headers that are used by real browsers and uses different headers on different resources just like a real browser. Cookies are stored in memory and used in the running session. However, due to security concerns, and also to reduce the performance impacts caused by the actor, the fake browser will neither render the web page nor evaluate scripts used by the web page. The reasons are discussed in chapter 13.

9.3.4 Hood action script

A hood action script is a script to describe actions to be done to the actor. Each line of the script can be seen as three parts: namespaces, commands, and arguments. For example, "hood:browser:goto:http://www.example.com". "hood:browser:" is to refer browser namespace from the hood namespace, "goto" is a command inside the browser namespace to tell the interpreter to use the browser to load a web page. "http://www.example.com" is the parameter of the "goto" command. In this example, it represents the URL of the web page to be loaded.

hood:browser:goto command

Its usage is "hood:browser:goto URL". Its effect is to use a fake browser (See section 9.3.3) to load a URL. When the special value "random_link" is used as the argument, the actor randomly picks a link from current web page of the browser to go to, but nothing will be done if the current web page contains no links.

hood:loop command

Its usage is "hood:loop:CONDITION:COMMAND". It is used to declare a loop. Multiple conditions can be declared in the CONDITION part by syntax "key=value[,key=value]". Currently, only two types of conditions are implemented: "exit_chance" and "delay". "exit_chance" describes the chance of the loop to be stopped. "exit_chance=0.33" means the loop has a 33% chance of termination at the beginning of each loop, "0" means it will never have an end. "1" means the loop will never be executed. "delay" describes the range of delay between each loop. "delay=0-2.5" means the delay between each loop is from 0 seconds to 2.5 seconds. A Just-In-Time compiler is implemented to convert the CONDITION field into Python code. The COMMAND part is the hood action script to be executed in the loop.

9.3.5 Logical view

fig. 9.1 is the logical view of the hood-actor.py, it shows the calling relationship between different components. The program use load_action function to load the action script and then calls play_in_action function which uses the HoodExecutor to schedule tasks in the action. The action tasks will call Actor class to evaluate scripts. The Actor class may call the Browser class to simulate the browsing event. The Browser class uses HoodHTMLParser class to extract the resources need to be loaded for the web page from HTML and then use HoodExecutor to load them in parallel.

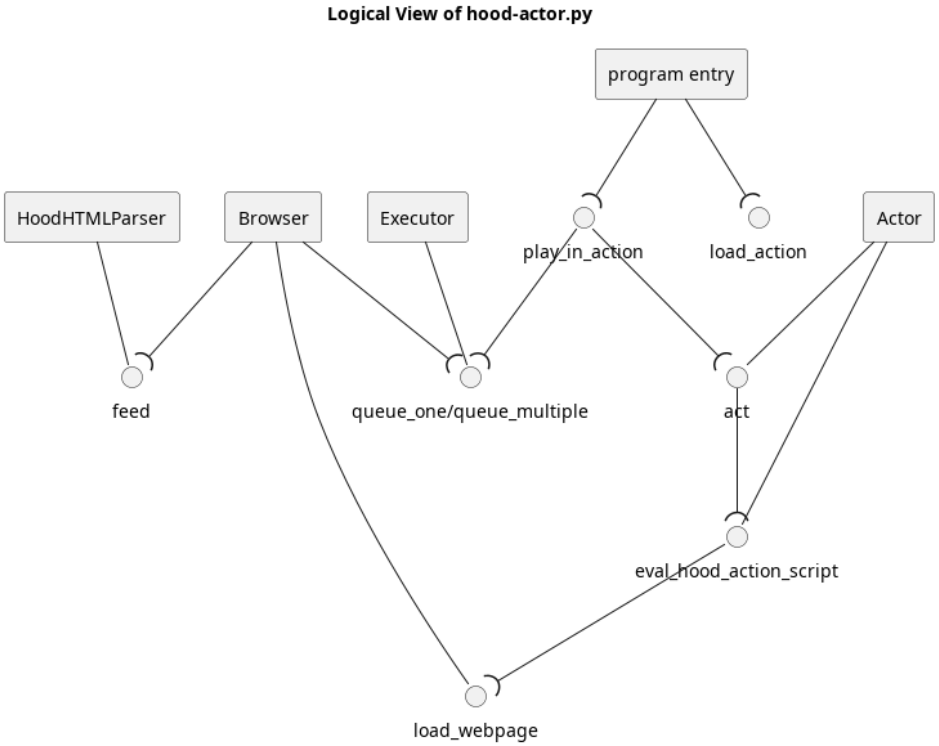

Figure 9.1. Logical view of hood-actor.py

Chapter 10

Dispatcher and firewall rules

Dispatcher is a shell script named 02-hood-dispatcher. It interacts with the changes in the network status or the system status to dynamically configure firewall rules and to start or stop network services. Its callers are three systemctl services: before-network.service (See appendix A.1), systemd-udevd.service (See appendix B.15), and NetworkManager-dispatcher.service.

10.1 Initial state of firewall rules

The initial state of firewall rules (See appendix A.7) is basic because the design of nftables made the rules with interface name involved requires to be done dynamically. It only allows IPv4 and ARP protocol at layer 3. It allows anything to lo interface but rejects anything else to communicate from 127.0.0.1 or to 127.0.0.1. It allows UDP/TCP packets in connection track established and related state. It allows the device to send or receive ICMP fragmentation-needed packets. Anything else is ignored.

10.2 On udev add event

10.2.1 udev

For udev, see appendix B.15.

10.2.2 netdev

In nftables, multiple networking levels are abstracted into families and netdev is the family that can see network packets just after the NIC driver passes them up to the networking stack. This very early location in the packet path is ideal for dropping packets. Dropping packets from an ingress chain is twice as efficient as doing so from a prerouting chain [Netfilter, 2021].

10.2.3 netdev filter

A default netdev filter will be added to both LAN and WAN interfaces. It has to be added dynamically because both its ingress and egress filters require the name of the device. The filter accepts only ARP and IP packets to pass through. The firewall rules from other places

already achieved the same thing but it is added because it is the earliest place to filter out a packet and the earlier a packet is filtered the less chance the packet to cause a problem.

10.3 On system startup

Upon system startup, the dispatcher will be called by before-network.service. It will start two dmesg (see appendix B.7) instances one is for redirecting firewall-related logs to a log file another is for formatting the log to a more human-readable form. In the iptables era, it was used for initializing iptables rules.

Since the udev event could be triggered at a very early stage of the startup, so early that the nftables service may not be available, and adding rules may fail. The dispatcher tries to add a netdev filter on all available network instances again because the timing before-network.service is sufficient to ensure the availability of nftables.service.

10.4 On NetworkManager-dispatcher pre-up event

This event means that a network interface is connected to the network but is not yet fully activated. The dispatcher checks the device path with the device specified during the installation process to know whether it is a WAN port. The device path is the real path of the network device. For example, on Linux, the device path of eth0 is the result of "realpath /sys/class/net/eth0".

WAN port

For the WAN port network interface, the dispatcher allows UDP packets to be sent out from the local 68 port to the remote 67 port and to be received from the remote 67 port to the local 68 port for DHCP protocol.

LAN port

For LAN port network interfaces, the dispatcher at first picks a randomized start number $\in [2, 255]$ as the third byte of the IPv4 address, and all following LAN ports use the result of

$$2 + (LastUsedNumber - 1) \mod 254$$

as the third byte. The randomization of the start number is made after each system starts. LAN subnet randomization could help increase the difficulty of malicious code in detecting the firewall from the LAN port.

After randomization, the following firewall rules are added:

- Accept the incoming TCP connections through the LAN interface if the destination IP address is the LAN IP address and the destination port is 80 or 443. For hood-http-handler.py and hood-tls-proxy
- Accept the incoming UDP packets through the LAN interface if the destination IP address is the LAN IP address and the destination port is 53. For the dnsmasq instance.
- Allow UDP packets to be sent out from the local 67 port to the remote 68 port and to be received from the remote 68 port to the local 67 port. For the dnsmasq instance.

List 10.1. Firewall rules to be added for LAN ports

A dnsmasq instance is started for this network interface, to assign IP addresses to connected devices, to let connected devices use the IP address of the LAN port of the firewall as the DNS server, and to respond to the IP address of the firewall LAN port for DNS requests.

10.5 On NetworkManager-dispatcher up event

The dispatcher only reacts to this event for the WAN port. It at first waits for DHCP to finish. After DHCP is done, the dispatcher removes DHCP related accept rules from nftables and adds a rule to accept outbound TCP connections from the WAN instance with the source address as the DHCP result to remote 80 and 443 ports. This rule is for HTTP and HTTPS protocols. After that, the dispatcher restarts the hood-network-services.service (appendix A.2). It will also check the existence of /do_upgrade or /run_once files to do a system upgrade or to run the script.

10.6 On NetworkManager-dispatcher down event

The dispatcher removes the firewall rules dynamically added for this instance and kills the dnsmasq instances for this interface.

Chapter 11

Installation

Installation now can only be done from Linux. install.sh is the shell script that is in charge of the installation process. It copies and applies the scripts, executables, and configuration files to the target Raspberry Pi OS file system. It accepts the command line arguments listed in table 11.1.

Name	Default	Description
usb_tether=	1	Share network to computer via USB cable
harden_only=	0	Only apply hardening parts. Let the target SBC can still used as a computer.
disable_wireless=	1	Disable WiFi and Bluetooth.
disable_gpu=	1	Disable GPU.
target=	/	The target root/device to install firewall.
wan_port_ device_path=	auto-built-in-eth	The path of the device to be used as WAN port. auto-built-in-eth means find built-in Ethernet port automatically.

Table 11.1. Command line arguments of install.sh

11.1 Checking target system

Multiple checks to the target path are done before the beginning of the installation process to avoid potential harm to the user's computer when a wrong path is used. It first checks the type of the target, if the target path is not a directory target will be treated as a device or file and will be mounted as the pattern of a live system of Raspberry Pi OS. If the target is a file, then the target will be mounted as a disk image. If the target does not exist, "/dev/" prefix will be added and will be treated as a name of a device. Then if the content of /boot/firmware/config.txt of the target does not contain the string "dtparam", the target will not be treated as a Raspberry Pi OS file system and the installation process will be aborted.

11.2 Disable wireless

Multiple methods are used to ensure the complete disablement of wireless devices. See table 11.2 and list 11.1.

Configuration file	Contents added
boot/firmware/config.txt	dtoverlay=disable-bt dtoverlay=disable-wifi
/etc/modprobe.d/bin-y-disable-wireless-blacklist.conf	blacklist bluetooth blacklist btbcm blacklist hci_uart blacklist i2c_brcmstb blacklist i2c_dev blacklist brcmfmac blacklist brcmutil blacklist cfg80211

Table 11.2. Configurations added to disable wireless

- /lib/firmware/brcm/*
- find /lib/linux-image*/broadcom -type f
- find /usr/lib/modules/ -name bluetooth

List 11.1. Files deleted from target to disable wireless

11.3 Disable GPU

Multiple methods are used to ensure the complete disablement of GPU. See table 11.3 and list 11.2.

Configuration file	Contents modified
boot/firmware/config.txt	Removed: dtoverlay=vc4-kms-v3d dtoverlay=vc4-fkms-v3d
/etc/modprobe.d/bin-y-disable-gpu-blacklist.conf	Added: blacklist v3d blacklist drm blacklist drm_panel_orientation_quirks

Table 11.3. Configurations modified to disable GPU

- Any directory under /usr/lib/modules/ with the name "gpu"

List 11.2. Files deleted from target to disable GPU

11.4 Other configurations modified

Configuration file	Contents modified	Purpose
/boot/firmware/cmdline.txt	Added: ipv6.disable=1 apparmor=1 security=apparmor	disable IPv6 enable AppArmor
/etc/modprobe.d/bin-y-blacklist.conf	Added: blacklist ipv6 blacklist hci_uart blacklist i2c_brcmstb blacklist i2c_dev	disable IPv6, i2c, and UART
/boot/firmware/config.txt	Added: enable_uart=0	disable UART
/etc/nftables.conf	See appendix A.7	nftables rules
/etc/hosts	OMITTED	provide address of some DNS services
/etc/sysctl.conf	OMITTED	sysctl related hardening
/etc/rc.local	OMITTED	host name randomization
/etc/pki/nssdb/cert9.db /etc/pki/nssdb/key4.db /etc/ca-certificates.conf	OMITTED	manage trusted certificates
/etc/NetworkManager/NetworkManager.conf	OMITTED	MAC address randomization
/etc/apt/sources.list /etc/apt/sources.list.d/raspi.list	OMITTED	use HTTPS mirrors for apt
files related to systemd services	OMITTED	enable hood services, enable required system services, and disable unsafe services

Table 11.4. Other configurations modified

Chapter 12

Showing network activities

Network activities of the firewall system are logged to multiple places by multiple services (see table 12.1). All logs are simplified to human-readable forms and aggregated to a tty device to let users access them easily. If the firewall was installed with the parameter harden_only=1, aggregated logs will be sent to tty8, and if the installation parameter harden_only was 0, aggregated logs will be sent to tty1. The network activities caused by hood-actor.py (see chapter 9) will not be included in the aggregated log.

Service	Log location
hood-name-service.py	/var/log/hood-name-service.log
hood-http-handler.py	/var/log/hood-http-handler.log
hood-tls-proxy	/var/log/hood-tls-proxy.log
hood-dispatcher	/var/log/hood-dispatcher.log
nftables	kernel ring buffer

Table 12.1. Log locations of network services.

Chapter 13

Analyzation to security events

This chapter analyzes security events and provides reasons why many things that could bring convenience are disabled in the firewall. Related data can be found from project GitHub repository [Yong, 2023].

13.1 Security releases and updates

The mentions counted in this section are from security releases, which means the components that did not cause a security issue would not be mentioned. Thus, the high number of mentions in this section represent high chance of causing problems.

Some people interpret high number of mentions as something has been more closely checked, and more vulnerabilities have also been fixed. However, such interpretation ignored the higher frequency of N-days attacks triggered by those security releases. They may have been more frequently checked by maintainers and users but may also have been more frequently checked by attackers. The data collected in this section are from 2020 to 2024, years after the initial release of those tools and technologies. Which means after all these years of patching, they are still causing vulnerabilities more frequently than others and implies that the bad habits of the developers or the problems in the design may have not been fixed by time.

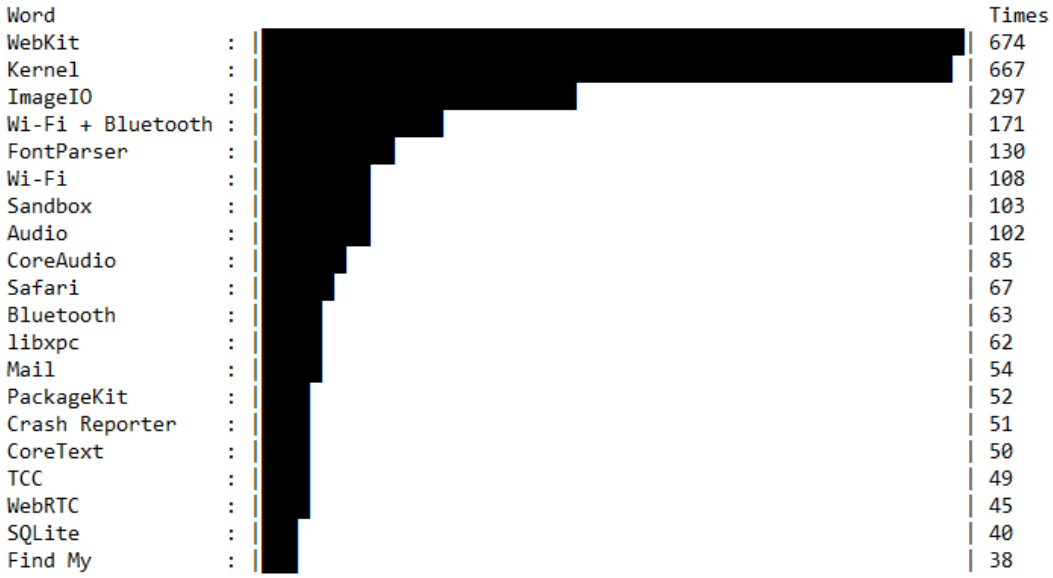

Figure 13.1. Top 19 mostly mentioned components in Apple security releases from January 8th, 2020 to January 12th, 2024

Texts of Apple security releases of its operating systems between January 8th, 2020 and January 12th, 2024 are collected and the number of times of their components being mentioned are counted. See fig. 13.1. They are the top 19 mostly mentioned components with one exception: the number of "Wi-Fi + Bluetooth" is the sum of the two.

The result explains how risky using a browser could be: WebKit, the underlying engine of Safari is the top 1 on the list, the browser Safari itself is the 9th place, and many things related to browsing are on the list: ImageIO, FontParser, CoreText, and WebRTC. Without the time spent on creating a 'fake' browser, the simulation of web-browsing network activity may have to choose between crude bash scripts or risky browser-based tools.

It also demonstrates why firewall disables wireless by default: Both "Wi-Fi" and "Bluetooth" are on the list, and their sum can take the fourth place.

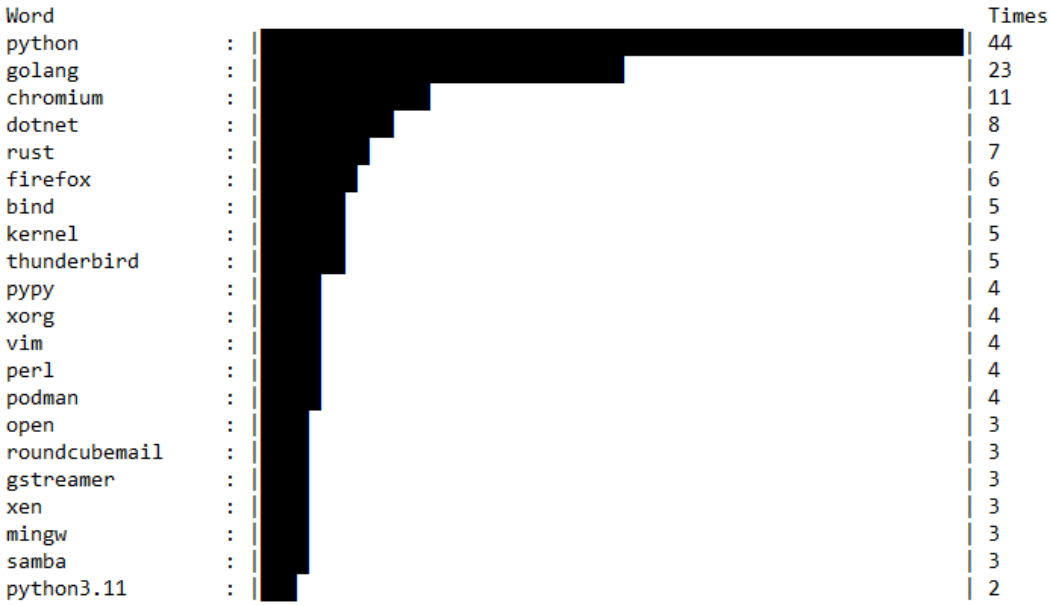

Figure 13.2. Top 20 mostly mentioned package name prefixes in Fedora 39 security updates from Feburary 14th, 2020 to January 10th, 2024
"python3.11" listed in the chart is not part of top 20

Package names of Fedora 39 security updates between February 14th, 2020 and January 10th, 2024 are collected, and the number of times of prefixes was mentioned are counted. See fig. 13.2. The data is not complete because only the titles of the links are collected and some titles omitted some packages like "a, b, c & X more". In this case, the "X more" is not counted. There are 18 records contains "X more" within 285 records, but the result can still spit out some truth because the more frequently a component causes a security update, the less the times of being published with a lot of other packages contributes to the total. The result also proves the high risk of browsers: both "firefox and "chromium" are in the top 10. The result also seemingly indicates the high risk of Python as "python" is 1st place, which renders the use of Python as a mistake, but the "python" prefix is shared by both the packages of multiple Python versions and the packages of different Python libraries. To evaluate the true risk of using Python, as the firewall only uses one Python version and only uses built-in modules, the prefix "python3.11" was counted additionally and the result was only twice.

13.2 IPv6

This section explains why firewall disables IPv6 by default. I searched both the keyword "IPv6" and the keyword "IPv4" from the website `https://cve.mitre.org`. On Jan 14th, 2024 (UTC +1) 00:00, there are 615 CVE records that match the keyword "IPv6" and 360 CVE records that match the keyword "IPv4". From 2022, there are 81 results match the keyword "IPv6" and 36 result match the keyword "IPv4". Those numbers indicate that the overall implementations of IPv6 is less stable than of IPv4. It could be explained as IPv4 implementations have a longer history of bug fixes.

13.3 GPU

One of the reasons to disable GPU is a stealthy GPU-based keylogger reported on EUROSEC 2013. [Ladakis et al., 2013]

13.4 Conclusions

As conclusion, most of decisions on disabling things are based on the analyzation of different security events and supported by numbers and related threats.

Chapter 14

Penetration test

To be close to real-world application of this tool, a computer from the university classroom and my iPhone are used to test the firewall. The computer has remote management tools pre-installed and I do not have the administrator permissions to configure the computer or to install some protection tools. Thus, without the help from the firewall, there is very little I can do. Experiment data can be found from project GitHub repository [Yong, 2023].

14.1 Man-in-the-middle (MITM) attack, DNS hijacking, and IP hijacking

14.1.1 Experiment tools

Following devices are used in the experiment: Two computers running live Parrot OS from Parrot-htb-6.0_amd64.iso. A Raspberry Pi 4B running hood firewall installed with the parameter "harden_only=1" to 2023-12-05-raspios-bookworm-arm64.img.xz.

14.1.2 Experiment steps

Computer 1 plays the role of internet provider and the role of certificate authority. Computer 2 plays the role of victim.

Create shared network

At first, enable network sharing on Computer 1 with following commands, and connect two computers via Ethernet ports.

```
– Computer 1, Terminal 1
> nmcli device set eno1 managed no
> sudo ip address add dev eno1 192.168.22.1
> sudo ip route add 192.168.22.0/24 dev eno1 src 192.168.22.1
> sudo sysctl –w net.ipv4.ip_unprivileged_port_start=0
> sudo sysctl –w net.ipv4.ip_forward=1
> sudo nft add table ip filter
```

```
> sudo nft add chain ip filter forward { type filter hook forward
    ↪ priority filter\; policy drop\; }
> sudo nft add rule ip filter forward ip saddr 192.168.22.0/24 iif
    ↪ eno1 oif wlp2s0 accept
> sudo nft add rule ip filter forward ct state {established, related
    ↪ } accept
> sudo nft add rule ip filter forward log prefix "Dropped " flags
    ↪ all drop
> sudo nft add table ip nat
> sudo nft add chain ip nat postrouting { type nat hook postrouting
    ↪ priority srcnat\; policy accept\; }
> sudo nft add rule ip nat postrouting ip saddr 192.168.22.0/24 ip
    ↪ daddr != 192.168.22.0/24 iif eno1 oif wlp2s0 masquerade
> sudo /usr/sbin/dnsmasq --conf-file=/dev/null --no-hosts --no-
    ↪ resolv --keep-in-foreground --interface=eno1 --except-
    ↪ interface=lo --clear-on-reload --strict-order --listen-address
    ↪ =192.168.22.1 --bind-interfaces --no-poll --server=1.1.1.1 --
    ↪ cache-size=0 --dhcp-range
    ↪ =192.168.22.1,192.168.22.255,255.255.255.0,400w --dhcp-
    ↪ authoritative --dhcp-leasefile=/tmp/yongbin.lease --dhcp-
    ↪ option=option:dns-server,192.168.22.1 --dhcp-option
    ↪ =3,192.168.22.1 --log-queries --log-facility=-
```

Without attack and protection

The following records show what happens without MITM attack: the text posted successfully and looks the same to both computers.

```
# post text without MITM attack
- Computer 2, Terminal 1
> echo "Thank you very much for the feedback. Your suggestion on X
    ↪ sounds legit. I will make X able to Y after I finished Z. I
    ↪ will send you another email once I finished related tasks." |
    ↪ curl -sS -F 'sprunge=<-' https://sprunge.us
http://sprunge.us/yqnNHp
> curl https://sprunge.us/yqnNHp
Thank you very much for the feedback. Your suggestion on X sounds
    ↪ legit. I will make X able to Y after I finished Z. I will send
    ↪  you another email once I finished related tasks.
- Computer 1, Terminal 2
> curl https://sprunge.us/yqnNHp
Thank you very much for the feedback. Your suggestion on X sounds
    ↪ legit. I will make X able to Y after I finished Z. I will send
    ↪  you another email once I finished related tasks.
# everything is correct
```

With MITM attack by DNS hijacking, without protection

Following commands are used to prepare MITM attack by DNS hijacking. The script used by mitmproxy is available at appendix A.6.

```
- Computer 1, Terminal 2
> sudo kill $(ps -ef |grep dnsmasq|grep yongbin -m 1|sed "s/\w*
    ↪ *//"|cut -d " " -s -f 1)
> sudo /usr/sbin/dnsmasq --conf-file=/dev/null --no-hosts --no-
    ↪ resolv --keep-in-foreground --interface=eno1 --except-
    ↪ interface=lo --clear-on-reload --strict-order --listen-address
    ↪ =192.168.22.1 --bind-interfaces --no-poll --address
    ↪ =/#/192.168.22.1  --cache-size=0  --dhcp-range
    ↪ =192.168.22.1,192.168.22.255,255.255.255.0,400w --dhcp-
    ↪ authoritative --dhcp-leasefile=/tmp/yongbin.lease --dhcp-
    ↪ option=option:dns-server,192.168.22.1 --dhcp-option
    ↪ =3,192.168.22.1 --log-queries --log-facility=-

- Computer 1, Terminal 1
> mitmproxy --listen-port 443 --listen-host 192.168.22.1 -s
    ↪ mitmattack.py

- Computer 1, Terminal 3
> python3 -m http.server -d ~/.mitmproxy/

- Computer 2, Terminal 1
> wget 192.168.22.1/mitmproxy-ca-cert.pem

- Computer 1, Terminal 3
> ^C
```

Following records show what happened under attack: The posted text looks correct to the victim but looks different to other person.

```
# post text when CA is doing MITM attack, without hood
- Computer 2, Terminal 1
> echo "Thank you very much for the feedback. Your suggestion on X
    ↪ sounds legit. I will make X able to Y after I finished Z. I
    ↪ will send you another email once I finished related tasks." |
    ↪ curl -sS --cacert mitmproxy-ca-cert.pem -F 'sprunge=<-' https
    ↪ ://sprunge.us
http://sprunge.us/ARbKPO
> curl -sS --cacert mitmproxy-ca-cert.pem https://sprunge.us/ARbKPO
Thank you very much for the feedback. Your suggestion on X sounds
    ↪ legit. I will make X able to Y after I finished Z. I will send
    ↪  you another email once I finished related tasks.
# The result looks correct to the victim
```

```
– Computer 1, Terminal 3
> curl https://sprunge.us/ARbKPO
Who do you think you are? How dare you to give me suggestions like
    ↪ this? Your naive ideas will never work. You have been blocked.
    ↪  *NEVER* send me emails again.
# However, the result looks different to other person
```

With MITM attack by DNS hijacking, with "hood" protection

Next, test what happens to the victim with "hood" firewall. Steps are following. 1. Connect Raspberry Pi to computer by USB cable 2. Disconnect the Ethernet cable from computer 2 and connect it to Raspberry Pi. 3. Use following commands to give direct access to other websites but still attacking the website sprunge.us.

```
– Computer 1, Terminal 3
> sudo kill $(ps -ef |grep dnsmasq|grep yongbin -m 1|sed "s/\w*
    ↪ *//"|cut -d " " -s -f 1)
> sudo /usr/sbin/dnsmasq --conf-file=/dev/null --no-hosts --no-
    ↪ resolv --keep-in-foreground --interface=eno1 --except-
    ↪ interface=lo --clear-on-reload --strict-order --listen-address
    ↪ =192.168.22.1 --bind-interfaces --no-poll --address=/sprunge.
    ↪ us/192.168.22.1 --server=1.1.1.1 --cache-size=0  --dhcp-range
    ↪ =192.168.22.1,192.168.22.255,255.255.255.0,400w --dhcp-
    ↪ authoritative --dhcp-leasefile=/tmp/yongbin.lease --dhcp-
    ↪ option=option:dns-server,192.168.22.1 --dhcp-option
    ↪ =3,192.168.22.1 --log-queries --log-facility=-
```

Following records show what happened: The victim bypassed the attack.

```
# Computer 2 starts to use hood firewall
– Computer 2, Terminal 1
> curl https://sprunge.us/ARbKPO
Who do you think you are? How dare you to give me suggestions like
    ↪ this? Your naive ideas will never work. You have been blocked.
    ↪  *NEVER* send me emails again.
# DNS redirect attack stopped working
```

With MITM attack by IP hijacking, with "hood" protection

In this section, the attacker used IP hijacking to achieve MITM attack. Computer 1 used following commands to conduct IP hijacking.

```
# Computer 1 starts to use more advanced method to achieve MITM
– Computer 1, Terminal 2
> sudo nft add chain ip nat prerouting { type nat hook prerouting
    ↪ priority dstnat\; policy accept\; }
```

```
> sudo nft add rule nat prerouting ip saddr 192.168.22.0/24 ip daddr
    ↪ {$(dig +short sprunge.us|tr "\n" ",")} iif eno1 dnat to
    ↪ 192.168.22.1
```

The same command starts to show error on the victim's computer.

```
– Computer 2, Terminal 1
> curl --cacert mitmproxy-ca-cert.pem https://sprunge.us/ARbKPO
curl: (35) OpenSSL SSL_connect: SSL_ERROR_SYSCALL in connection to
    ↪ sprunge.us:443
```

The output of the "hood" firewall shows to the victim that something is wrong with the certificate.

```
# output from hood firewall:
DNS query 'sprunge.us'
 verifying sprunge.us 172.67.195.185 :/CN=sprunge.us
sprunge.us handshake failed: certificate verify failed (SSL routines
    ↪ )
 verifying sprunge.us 104.21.84.187 :/CN=sprunge.us
sprunge.us handshake failed: certificate verify failed (SSL routines
    ↪ )
No endpoints sprunge.us
```

14.1.3 Conclusions

The experiment records show that the "hood" firewall can help a victim to bypass simple DNS hijacking conducted by network providers, the "hood" firewall could not protect the victim from IP hijacking but if the attacker wants to decrypt the TLS connection, even if the root certificate has been trusted by the victims computer, the "hood" firewall can still alert the victim and block the connection. However, without this firewall, the victim of the attacks could see nothing while using software tools normally.

14.2 Information gathering by metasploit

14.2.1 Experiment tools

Following devices are used in the experiment: A computer running live Kali Linux from kali-linux-2024-W03-live-amd64.iso. A Raspberry Pi 4B running 'hood' firewall installed with the parameter "harden_only=1" to 2023-12-05-raspios-bookworm-arm64.img.xz. A computer in the university classroom.

14.2.2 Information gathering from university computer

The following are selected commands and outputs, full records are available from project GitHub repository [Yong, 2023]:

```
# first, scan opening ports
db_nmap -Pn -A 10.10.6.64
[*] Nmap: Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-20
    ↪ 23:20 UTC
[*] Nmap: Nmap scan report for USILU-3325.usilu.net (10.10.6.64)
[*] Nmap: Host is up (0.00047s latency).
[*] Nmap: Not shown: 996 filtered tcp ports (no-response)
[*] Nmap: PORT     STATE SERVICE       VERSION
[*] Nmap: 135/tcp  open  msrpc         Microsoft Windows RPC
[*] Nmap: 139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
[*] Nmap: 445/tcp  open  microsoft-ds?
[*] Nmap: 3389/tcp open  ms-wbt-server Microsoft Terminal Services
[*] Nmap: | ssl-cert: Subject: commonName=USILU-3325.usilu.net
[*] Nmap: | Not valid before: 2023-11-22T23:00:22
[*] Nmap: |_Not valid after:  2024-05-23T23:00:22
[*] Nmap: | rdp-ntlm-info:
[*] Nmap: |   Target_Name: CAMPUS
[*] Nmap: |   NetBIOS_Domain_Name: CAMPUS
[*] Nmap: |   NetBIOS_Computer_Name: USILU-3325
[*] Nmap: |   DNS_Domain_Name: usilu.net
[*] Nmap: |   DNS_Computer_Name: USILU-3325.usilu.net
[*] Nmap: |   DNS_Tree_Name: usilu.net
[*] Nmap: |   Product_Version: 10.0.19041
[*] Nmap: |_  System_Time: 2024-01-20T22:20:28+00:00
[*] Nmap: |_ssl-date: 2024-01-20T22:21:11+00:00; -1h00m00s from
    ↪ scanner time.
[*] Nmap: Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
[*] Nmap: Host script results:
[*] Nmap: |_clock-skew: mean: -1h00m00s, deviation: 0s, median: -1
    ↪ h00m00s
[*] Nmap: | smb2-time:
[*] Nmap: |   date: 2024-01-20T22:20:28
[*] Nmap: |_  start_date: N/A
[*] Nmap: | smb2-security-mode:
[*] Nmap: |   3:1:1:
[*] Nmap: |_    Message signing enabled but not required
[*] Nmap: |_nbstat: NetBIOS name: USILU-3325, NetBIOS user: <unknown
    ↪ >, NetBIOS MAC: 5c:f9:dd:e0:97:34 (Dell)
[*] Nmap: Service detection performed. Please report any incorrect
    ↪ results at https://nmap.org/submit/ .
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 60.28
    ↪ seconds
```

```
# gather information from 445 port by scanning Server Message Block
    ↪ protocol

msf6 auxiliary(scanner/smb/smb_version) > run

[*] 10.10.6.64:445        - SMB Detected (versions:2, 3) (preferred
    ↪ dialect:SMB 3.1.1) (compression capabilities:LZNT1) (
    ↪ encryption capabilities:AES-128-GCM) (signatures:optional) (
    ↪ guid:{a77b41a6-e4e1-4033-8b65-bc9f0b5ff320}) (authentication
    ↪ domain:CAMPUS)
[*] 10.10.6.64:           - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

# gather information from 3389 port by scanning Remote Desktop
    ↪ protocol

msf6 auxiliary(scanner/rdp/rdp_scanner) > run

[*] 10.10.6.64:3389       - Detected RDP on 10.10.6.64:3389       (
    ↪ name:USILU-3325) (domain:CAMPUS) (domain_fqdn:usilu.net) (
    ↪ server_fqdn:USILU-3325.usilu.net) (os_version:10.0.19041) (
    ↪ Requires NLA: Yes)
[*] 10.10.6.64:3389       - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed

# gather information from 135 port by scanning Microsoft Remote
    ↪ Procedure Call protocol

msf6 auxiliary(scanner/dcerpc/tcp_dcerpc_auditor) > run

10.10.6.64 - UUID 99fcfec4-5260-101b-bbcb-00aa0021347a 0.0 OPEN VIA
    ↪ 135 ACCESS GRANTED
    ↪ 000000000000000000000000000000000000000000000000000000000000
[*] 10.10.6.64:135        - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

The experiment used only free tools from Kali Linux but still collected a lot of information about the computer: It's manufacturer, operating system, system version, clock skew, domain, host name, time. This shows the necessity of using a firewall.

14.2.3 Information gathering from iOS

Following materials are used in the experiment: A computer running live Kali Linux from kali-linux-2024-W03-live-amd64.iso. An iPhone.

The prepare steps are following. First, boot the computer into Kali Linux live environment. Then enable tethering on iPhone and connect it to the computer. Then press the "Trust" button on the confirmation dialog. Then use following command from metasploit framework.

It shows that iOS devices could be identified by their unique opened port numbers: 49152 and 62078.

14.2.4 Information gathering from "hood" firewall

The following are selected commands and outputs, full records are available from project GitHub repository [Yong, 2023]:

```
# scan opening ports
db_nmap -Pn -A 10.42.0.171
[*] Nmap: Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-01-22
    ↪ 22:40 UTC
[*] Nmap: Nmap scan report for 10.42.0.171
[*] Nmap: Host is up.
[*] Nmap: All 1000 scanned ports on 10.42.0.171 are in ignored
    ↪ states.
[*] Nmap: Not shown: 1000 filtered tcp ports (no-response)
[*] Nmap: Service detection performed. Please report any incorrect
    ↪ results at https://nmap.org/submit/ .
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 206.84
    ↪ seconds
```

No useful information collected from the firewall, it indicates the success of protection.

14.3 Eavesdropping

14.3.1 Experiment tools

Following tools are used in the experiment: A computer running live Kali Linux from kali-linux-2024-W03-live-amd64.iso. A Raspberry Pi 4B running the 'hood' firewall installed with the parameter "harden_only=1" to 2023-12-05-raspios-bookworm-arm64.img.xz. An Ethernet cable. A smartphone.

14.3.2 Experiment steps

Two environments were used to check the protection of "hood" firewall against eavesdropping.

First, check what network packet be captured by eavesdroppers under the protection of "hood" firewall. The experiment steps on eavesdropping attack are following. First, connect the computer to internet shared by the smartphone. Then create a shared network on the network interface of the Ethernet port (Advanced Network Configuration -> Wired connection 1 ->

IPv4 Settings Method Shared to other computers, IPv6 Settings disabled). Then, use Wireshark to capture on the interface of the shared network. Then use the cable connect the Ethernet port of the Raspberry Pi to the Ethernet port of the computer. Then, execute the command "curl example.com" on Raspberry Pi.

Second experiment is made for comparison. Use the Wireshark to capture on Kali Linux with internet connected and execute the command "curl example.com".

14.3.3 Experiment results

Results of the first experiment are following: Wireshark did not capture any plaintext DNS traffic from Raspberry Pi. Wireshark captured only one HTTP session, which connected to http://1.1.1.1. It was the mechanism used by the dispatcher (chapter 10) to detect internet connectivity. Wireshark captured the TLS handshake to example.com but could not see the content of the request and the response. The TTY8 of "hood" showed the activity of connecting to example.com and the certificates used by example.com (See fig. 14.1).

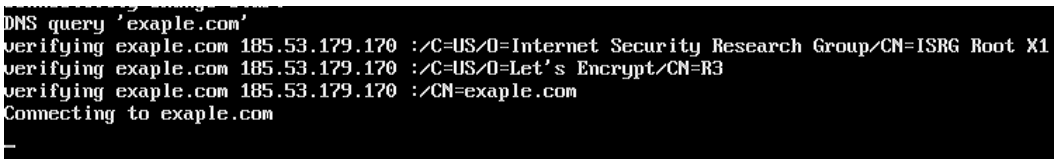

Figure 14.1. The output of TTY8 on Raspberry Pi

As the results of the second experiment, Wireshark captured plaintext requests and plaintext responses of both the DNS query to example.com and the HTTP request to example.com. The manufacturer of the MAC address used by "hood" could not be identified by Wireshark.

14.3.4 Conclusions

As conclusion, the "hood" firewall successfully protected against eavesdropping attacks targeting plaintext HTTP traffic and plaintext DNS traffic. It can also tell the user what certificates the remote endpoints are using and the fact that the firewall is connecting to that website

14.4 Management software

The computer at the classroom has TeamViewer pre-installed. When using the computer without "hood" firewall, the TeamViewer shows "Ready to connect (secure connection)" in its status bar, after using "hood" firewall, its status bar changed to "Only LAN connections are possible". However, other person could not access the LAN behind the firewall and thus managing this computer via TeamViewer remotely is no more possible. It shows that "hood" firewall can block some management tools.

Chapter 15

Conclusions

A working firewall system is designed and implemented during this work. It narrowed the attack surface by simplifying the hardware and by avoiding the use of high risk components. It mitigated many security flaws of existing network protocols by using creative solutions. The overall money cost of using this firewall is much lower than the median of popular firewall hardware. The statistics of past security events explained the reason of some decisions behind the design by proving the high risk of browsers and wireless communications. The test results proved the difficulty for a victim to notice a MITM attack, proved the easiness of gathering information from network devices, proved the possibility of eavesdropping, and proved the effectiveness of the protection provided by "hood" firewall against those attack. If I had more time or a better environment to work on this thesis, more content would have added to the paper. Current version could not show the full picture of this work.

15.1 Possible improvements for future

15.1.1 LSM and seccomp

Use Linux Security Modules (LSMs), like AppArmor, and seccomp (see appendix B.14) to protect as many processes as possible. The programs that do not support seccomp can be changed by LD_LOAD_LIBRARY.

15.1.2 Compile time hardening

Use strict compiler options to harden everything, including kernel, like what Gentoo Linux is doing now, to try to mitigate some unpublished vulnerabilities, and also to increase the difficulty of attacking the firewall itself.

15.1.3 Network stack fingerprinting

Spoof network stacks, like TCP stacks and TLS stacks, make the firewall and the devices behind it to be less detectable.

15.1.4 Further use to libcomposite

Libcomposite can make the device show multiple roles to a host (a computer), which means the device can at the same time work as a USB mass storage device or as a USB CDROM, both of which can be used as a media of a Live DVD. Then the computer will be able to boot a live system from it. For a computer that is password protected to boot from USB, we can also use this small device as a PXE server to make that computer boot from the network of the USB device.

Appendix A

Simple scripts and services created by firewall

A.1 before-network.service

before-network.service is a systemd service being enabled during installation process and run before network is available to system. It is created to trigger dispatcher (See chapter 10) to initialize firewall rules and network configurations.

A.2 hood-network-services.service

It is a service for running hood network services. It runs hood-network-services-runner.sh which starts hood-http-handler.py, hood-name-service.py, hood-tls-proxy, and a dnsmasq client for DNS queries made from the firewall system.

A.3 allowd_tls_ports.txt

TLS is widely applied to secure protocols, not just for HTTP. This file is created to support more TLS-encrypted protocols in the future. It is a text file with lines of port numbers. Lines starting with # are treated as comments and are ignored. Ports listed in the file will be allowed in nftables rules by the dispatcher and will be handled by hood-tls-proxy.

A.4 domain_blacklist.txt

It is a text file with lines of regular expressions. Lines starting with # are treated as comments and are ignored. Domains matching any one of the regular expressions in the file will be blocked.

A.5 ip_subnet_blacklist.txt

It is a text file with lines of IP subnets. Lines starting with # are treated as comments and are ignored. IP addresses matching any one of the subnets in the file will be blocked.

A.6 mitmattack.py

```
#!/bin/python3
from mitmproxy import http
import logging

hidden_truth={}
what_others_see="Who␣do␣you␣think␣you␣are?␣How␣dare␣you␣to␣give␣me␣
    ↪ suggestions␣like␣this?␣Your␣naive␣ideas␣will␣never␣work.␣You␣
    ↪ have␣been␣blocked.␣*NEVER*␣send␣me␣emails␣again."

def request(flow: http.HTTPFlow) -> None:
  if flow.request.text:
    if 'Content-Disposition:␣form-data;␣name="sprunge"' in flow.
        ↪ request.text:
      lines = flow.request.text.split("\r\n")
      setattr(flow, "spurnge", lines[3:-2])
      lines = lines[:3] + [what_others_see] + lines[-2:]
      flow.request.text = "\r\n".join(lines)
  for url, text in hidden_truth.items():
    print(url, flow.request.url)
    if flow.request.url.startswith(url):
      setattr(flow, "binyong", text)
      print(1111)
      break

def response(flow: http.HTTPFlow) -> None:
  if hasattr(flow, "spurnge"):
    truth = "\r\n".join(flow.spurnge)
    hidden_truth[flow.response.text.strip()] = truth
    hidden_truth["https" + flow.response.text.strip()[4:]] = truth
    print(flow.response.text)
  if hasattr(flow, "binyong"):
    flow.response.text = flow.binyong
    print(flow.response.text)
```

A.7 nftables.conf

```
#!/usr/sbin/nft -f
```

```
flush ruleset

table ip filter {
  chain input {
    type filter hook input priority filter; policy drop;
    iif lo accept
    ip daddr 127.0.0.1/8 log prefix "[HOOD D]" flags all drop
    ip saddr 127.0.0.1/8 log prefix "[HOOD D]" flags all drop
    meta l4proto udp ct state {established, related} log prefix "[
        ↪ HOOD A]" flags all accept
    ct state {established, related} accept
    icmp type {destination-unreachable} icmp code {frag-needed}
        ↪ accept
    log prefix "[HOOD D]" flags all drop
  }
  chain forward {
    type filter hook forward priority filter; policy drop;
    log prefix "[HOOD D]" flags all drop
  }
  chain output {
    type filter hook output priority filter; policy drop;
    oif lo accept
    ip daddr 127.0.0.1/8 log prefix "[HOOD D]" flags all drop
    ip saddr 127.0.0.1/8 log prefix "[HOOD D]" flags all drop
    meta l4proto udp ct state {established, related} log prefix "[
        ↪ HOOD A]" flags all accept
    ct state {established, related} accept
    icmp type destination-unreachable icmp code {frag-needed} accept
    log prefix "[HOOD D]" flags all drop
  }
}

table ip6 filter {
  chain ingress {
    type filter hook input priority filter; policy drop;
    log prefix "[HOOD D]" flags all drop
  }
  chain prerouting {
    type filter hook input priority filter; policy drop;
    log prefix "[HOOD D]" flags all drop
  }
  chain input {
    type filter hook output priority filter; policy drop;
    log prefix "[HOOD D]" flags all drop
  }
  chain forward {
```

```
    type filter hook forward priority filter; policy drop;
    log prefix "[HOOD D]" flags all drop
  }
  chain output {
    type filter hook output priority filter; policy drop;
    log prefix "[HOOD D]" flags all drop
  }
  chain postrouting {
    type filter hook forward priority filter; policy drop;
    log prefix "[HOOD D]" flags all drop
  }
}

table bridge filter {
  chain ingress {
    type filter hook input priority filter; policy drop;
    log prefix "[HOOD D]" flags all drop
  }
  chain prerouting {
    type filter hook prerouting priority filter; policy drop;
    log prefix "[HOOD D]" flags all drop
  }
  chain input {
    type filter hook output priority filter; policy drop;
    log prefix "[HOOD D]" flags all drop
  }
  chain forward {
    type filter hook forward priority filter; policy drop;
    log prefix "[HOOD D]" flags all drop
  }
  chain output {
    type filter hook output priority filter; policy drop;
    log prefix "[HOOD D]" flags all drop
  }
  chain postrouting {
    type filter hook postrouting priority filter; policy drop;
    log prefix "[HOOD D]" flags all drop
  }
}

table netdev filter {
}


table arp filter {
  chain input {
    type filter hook input priority filter; policy drop;
```

```
  }
  chain output {
    type filter hook output priority filter; policy drop;
  }
}
```

Appendix B

The attacks encountered during the time I was working on this thesis

B.1 Malicious hardware

Normal attackers will use the network, Bluetooth, or WiFi for communications between malware and the host. For example, casting the screen to another device via WiFi or sending keyboard inputs via Bluetooth. Those kinds of attacks can be detected by an RF detector. RF shielding fabric, RF detectors, and signal jammers could be used to fight against such kinds of attacks.

However, there are still other methods. Some even without wireless communications. Despite I had already bought an RF detector to alert me of wireless communications between unknown hardware, unknown attackers still managed to monitor my progress with some modified hardware. Take the devices that I am using to develop this project for example. Raspberry 4B has a USB chip that can communicate with the power source. If the charger is specially crafted with the ability to forward the USB connection to the remote endpoint via power lines, then, with the help of the malware installed on the Pi itself, data can be leaked silently via power lines even without any network connections to the computer. The same story can also happen to the portable screen that I am using now. I have found two counter measurements to this kind of attack: One is to use a USB-C to DC adapter when connecting a USB-C charger to the laptop. Another is to tape the two pins in the middle of the male USB-A port to prevent data communications.

Many of my SD card readers, USB sticks, and an external CD/DVD drive have their firmware modified by unknown person. The DVD drive I bought from Amazon `https://www.amazon.de/-/en/gp/product/B0BLGQZ66D/` has its firmware changed. Following is the detection result from the host computer. Vendor changed to Initico Corporation, and the serial number changed to 000000000000000000000000. Usually, if they just want to install a backdoor to the device firmware, they do not have to change the vendor and the serial number. The fact that they are doing this shows that they are letting me know it on purpose. This act has two possible intentions: one is to deliver a warning because it is telling me that they do not care if I can find

out, another is to let me waste my time on this endless fight against infections and make me unable to focus on the main goal of my life, because if they do not let me notice this change, I may do nothing about it. The same story also happened to my SD card reader and USB sticks. The serial number of my SD card reader changed to ABCDEF0123456789AB.

```
lsusb -v -s 001:005

BBus 003 Device 005: ID 13fd:0840 Initio Corporation INIC-1618L SATA
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               2.00
  bDeviceClass            0
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0        64
  idVendor           0x13fd Initio Corporation
  idProduct          0x0840 INIC-1618L SATA
  bcdDevice            1.14
  iManufacturer           1 Generic
  iProduct                2 External
  iSerial                 3 000000000000000000000000
  bNumConfigurations      1
  Configuration Descriptor:
    bLength                 9
    bDescriptorType         2
    wTotalLength       0x0020
    bNumInterfaces          1
    bConfigurationValue     1
    iConfiguration          0
    bmAttributes         0xc0
      Self Powered
    MaxPower                2mA
    Interface Descriptor:
      bLength               9
      bDescriptorType       4
      bInterfaceNumber      0
      bAlternateSetting     0
      bNumEndpoints         2
      bInterfaceClass       8 Mass Storage
      bInterfaceSubClass    2 SFF-8020i, MMC-2 (ATAPI)
      bInterfaceProtocol   80
      iInterface            0
      Endpoint Descriptor:
        bLength             7
        bDescriptorType     5
        bEndpointAddress 0x81  EP 1 IN
```

```
        bmAttributes            2
          Transfer Type            Bulk
          Synch Type               None
          Usage Type               Data
        wMaxPacketSize     0x0200  1x 512 bytes
        bInterval               0
      Endpoint Descriptor:
        bLength                 7
        bDescriptorType         5
        bEndpointAddress     0x02  EP 2 OUT
        bmAttributes            2
          Transfer Type            Bulk
          Synch Type               None
          Usage Type               Data
        wMaxPacketSize     0x0200  1x 512 bytes
        bInterval               0
Device Qualifier (for other device speed):
  bLength                10
  bDescriptorType         6
  bcdUSB               2.00
  bDeviceClass            0
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0        64
  bNumConfigurations      1
Device Status:     0x0001
  Self Powered
```

B.2 Sounds

Another attack that can bypass a computer without a wireless device, is to use AI / ML to identify the sounds of keyboard hits. The detected types can be sent to remote via power lines or mobile phones. The sound could even be recorded from the room of a neighbor which makes this attack more stealthy than other methods. I use cardboards to extend the pillar of the key cap to shorten the key travel to lower the volume of the sound of the key hit to counter this attack but I am uncertain about the effect because I have no attack tools to test this. AliPay also used to use sound waves to transmit data between phones and vending machines. My Raspberry Pi recently started to emit strange noises from the speakers on the screen, which could be because of the same kind of technology being used by hackers.

B.3 Direct physical access

A lot of my storage medias were stolen, including USB sticks, MicroSD cards, and CDs. A police told me that he does not trust me and he failed to find the thief. Sometimes they also

swap my storage media with another one with the same look but with torjan installed while I'm absent.

And even further: the Google account of the email of my previous GitHub Account was stolen. I have two-factor authenciation (2FA) enabled and the Google Authenticator was installed on an iPhone that only for 2FA authenticators. The iPhone does not have a SIM card inserted and only uses internet for updates. However, when the Google account was stolen, both the recovery email and the recovery phone were changed, but I received nothing to alert me. This kind of attack is unlikely to happen to ordinary people and my ordinary status makes it hard to believe.

Glossary

B.4 AppArmor

AppArmor is a Linux Security Module implementation of name-based mandatory access controls. AppArmor rules can confine program's access to files and POSIX 1003.1e capabilities.

B.5 CDC-ECM

It is the Ethernet Control Model (ECM) Devices subclass of Universal Serial Bus (USB) Communication Device Class (CDC). It's specification is available on the website `http://www.usb.org`. It provides a virtual Ethernet link to host OS but it is not supported by most of Apple products.

B.6 dhclient

Dhclient is a Linux command line Dynamic Host Configuration Protocol client.

B.7 dmesg

Dmesg is a linux command line tool that can print or control the kernel ring buffer.

B.8 dnsmasq

Dnsmasq is a software that provides network infrastructure for small networks: DNS, DHCP, router advertisement and network boot.

B.9 dtbo file

dtbo means devicetree blob for overlay [Project, 2023a]. It is the compact binary representation of the devicetree used for modifing the kernel's living devicetree [kernel development community, 2024]. A devicetree is a tree data structure with nodes that describe the devices in a system. [Limited, 2023]

B.10 hood

"hood" is the name of this firewall project.

B.11 nftables

Nftables is the modern Linux kernel packet classification framework. It is available in Linux kernels >= 3.13.

B.12 Puppeteer

Puppeteer is a Node.js library which provides a high-level API to control Chrome/Chromium over the DevTools Protocol. It can automate form submission, do UI testing, and program keyboard input to browsers.

B.13 RNDIS

The Remote Network Driver Interface Specification (RNDIS) is a Microsoft proprietary protocol. It provides a virtual Ethernet link to host OS.

B.14 seccomp

Seccomp is a computer security facility in the Linux kernel. seccomp allows a process to make a one-way transition into a "secure" state where it cannot make any system calls except exit, sigreturn, read and write to already-open file descriptors.

B.15 systemd-udevd.service

udev supplies the system software with device events, manages permissions of device nodes and may create additional symlinks in the /dev/ directory, or renames network interfaces.

The udev daemon, systemd-udevd.service(8), receives device uevents directly from the kernel whenever a device is added or removed from the system, or it changes its state. When udev receives a device event, it matches its configured set of rules against various device attributes to identify the device. Rules that match may provide additional device information to be stored in the udev database or to be used to create meaningful symlink names.

B.16 Wireshark

Wireshark is a data capturiong program that used for analyzing network protocols. It can present the captured packets as values of fields inside the data structures of the network protocol.

Bibliography

Yi-Chao Chen, Yong Liao, Mario Baldi, Sung-Ju Lee, and Lili Qiu. Os fingerprinting and tethering detection in mobile networks. In *IMC '143: Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 173–180, New York, NY, United States, November 2014. Association for Computing Machinery. doi: https://doi.org/10.1145/2663716.2663745.

Intel Corporation. Intel® converged security and management engine (intel® csme) security technical white paper rev1.5. Technical report, Intel®, October 2022. URL https://www.intel.com/content/dam/www/public/us/en/security-advisory/documents/intel-csme-security-white-paper.pdf. Online; accessed January 26, 2024.

Intel Corporation. What is intel® management engine?, September 2023. URL https://www.intel.com/content/www/us/en/support/articles/000008927/software/chipset-software.html. Online; accessed January 26, 2024.

Ralph Droms. Dynamic Host Configuration Protocol. RFC 2131, RFC Editor, March 1997. URL https://www.rfc-editor.org/rfc/rfc2131.

FORTINET. Security profiles. In *FortiOS - Administration Guide*, chapter 9, pages 1026–1029. June 2021. URL https://fortinetweb.s3.amazonaws.com/docs.fortinet.com/v2/attachments/137da60a-5f22-11ea-9384-00505692583a/FortiOS-6.4.0-Administration_Guide.pdf.

Python Software Foundation. ssl - tls/ssl wrapper for socket objects. In *Python 3.12.1 documentation*. December 2023. URL https://docs.python.org/3/library/ssl.html#ssl.create_default_context.

Paul Hoffman and Patrick McManus. DNS Queries over HTTPS (DoH). RFC 8484, RFC Editor, October 2018. URL https://www.rfc-editor.org/rfc/rfc8484.

The kernel development community. Devicetree overlay notes — the linux kernel documentation, January 2024. URL https://www.kernel.org/doc/html/latest/devicetree/overlay-notes.html. Online; accessed January 24, 2024.

Jonghoon Kwon, Jeonggyu Song, Junbeom Hur, and Adrian Perrig. Did the shark eat the watchdog in the ntp pool? deceiving the ntp pool's monitoring system. In *USENIX Security'23*, USENIX Security Symposium. USENIX, August 2023. URL https://www.usenix.org/system/files/usenixsecurity23-kwon.pdf.

Evangelos Ladakis, Lazaros Koromilas, Giorgos Vasiliadis, Michalis Polychronakis, and Sotiris Ioannidis. You can type, but you can't hide: A stealthy gpu-based keylogger. In *sixth European Workshop on Systems Security (EuroSec)*, European Workshop on Systems Security (EuroSec), April 2013. URL https://www.syssec-project.eu/m/page-media/3/ladakis-eurosec13.pdf.

Linaro Limited. Devicetree specification release v0.4, June 2023. URL https://github.com/devicetree-org/devicetree-specification/releases/tag/v0.4. Online; accessed January 24, 2024.

Kathleen Moriarty and Stephen Farrell. Deprecating tls 1.0 and tls 1.1. RFC 8996, RFC Editor, March 2021. URL https://www.rfc-editor.org/rfc/rfc8996.

Netfilter. Nftables families, July 2021. URL https://wiki.nftables.org/wiki-nftables/index.php/Nftables_families#netdev. Online; accessed February 02, 2024.

Android Open Source Project. Device tree overlays, August 2023a. URL https://source.android.com/docs/core/architecture/dto. Online; accessed January 24, 2024.

The FreeBSD Documentation Project. Chapter 28.usb device mode/usb otg. In *FreeBSD Handbook*, chapter 28. July 2023b. URL https://docs.freebsd.org/en/books/handbook/usb-device-mode/.

Yaron Sheffer, Peter Saint-Andre, and Thomas Fossati. Recommendations for secure use of transport layer security (tls) and datagram transport layer security (dtls). RFC 9325, RFC Editor, November 2022. URL https://www.rfc-editor.org/rfc/rfc9325.

W3Techs. Usage statistics of ssl certificate authorities for websites, January 2024. URL https://w3techs.com/technologies/overview/ssl_certificate. Online; accessed January 29, 2024.

Bin Yong. hood firewall Github repository, 2023. URL https://github.com/YongBinnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn/hood.
1