Design and implementation of a hardware firewall

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

September 2023

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, Yesterday September 2023

Abstract

Design and implementation of a firewall device based on Raspberry Pi. The firewall will monitor and encrypt network activities. It is designed for someone who would like to sacrifice some compatibility to pursue better security but still wants some balance between security and convenience. A sensitive target, like an investigative journalist, could be a potential user of this device.

Acknowledgements

This document is a draft version of a working thesis of Bin Yong.

Contents

Figures

Tables

Chapter 1

Introduction

Hacking and surveillance tools are widely deployed The goal of this work is to increase the chance of survival of the user from hacking, evasdropping, and digital fingerprinting. The device works as a strict firewall which limits network activities and applies encryption standards to increase the difficulty of eavesdropping, and also to reduce the attack surface of fingerprinting. The device run a TLS proxy to supervise and manage TLS connections. It uses a screen to display selected real-time internet activities.

Using a hardware as a firewall has several advantages. Firstly, hardware firewall can provide complete isolation between highly unsafe code, like a browser, and firewall software (fun fact: I was hacked repeatedly through personally harddened latest version of firefox while writing this thesis). This could also work as a mitigation of CPU/BIOS level threats: Firmware malwares, bootkits, and doubtable proprietray technologies like Intel ME and the AMD PSP. Second, it also provide convenience to the user: the configuration of this portable device is applied to protect everything behind it, so people do not need to do the time consuming configuration work on different softwares and operating systems one by one.

Even things that could not be configured is under the restriction of the firewall. In example, people cannot untrust a built-in trusted root certificate from iPhone via Settings app.

Chapter 2

Hardware

2.1 Theory of the cost

Commercial firewalls are usually more expensive than development boards that contains the same level of hardware functioality. One one hand, It's acceptable because they provided addtional values throught their software and services and all of the development and maintainance of the product, the service, and the company itself are costing money. On the other hand, it also indicates that an open-source solution like the tool introduced in this paper would reduce lower bound of the cost of accessing a hardware firewall.

2.2 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 a firewall devices are around the price 250.74 EURO, and most of them are more expensive than the price of a Raspberry Pi 4B (68.79 EURO) selling on the same website. Only one exception: TP-LINK ER605. It's price is 55.5 euro which is still higher than the price of a Raspberry P 3B on the same website. After seeking into the details of this exception, its central management design and the corresponding cloud-based controller both showing that it is desigend for a totally different threat model and the its exceptional price could be explained, because it is under the control of that company and when people are motivated to spend money on a hardware firewall, their requirement to information security usually do not want to give full control of their firewall to a company. As a conclusion, 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, 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 Choosing Raspberry Pi

The historical reason why Raspberry Pi is used is because it is the only single board computer (SBC) I have when this firewall was started as a hobby project to solve the aggarsive hack problem that I was facing. The decision is unchanged after the start of this thesis mainly because the time costs on trying new SBCs: Since its first release on 2012, Raspberry Pi has upgraded its hardware models and software for multiple times, has accumluated a lot of working projects, and has maintained an active user community with a large number of experinced users behind it. All of them shows that using Raspberry Pi, I could cost less time on fixing the bugs of the SBC itself or on waiting for the answers from the community, because preivous users has asked and answered many questions and has found and fixed many problems. Besides that, its large number of users could also help on spreading the firewall project and help on pointing out security flaws of its new upgrades.

Chapter 3

Operating system

Due to its security-focused nature, OpenBSD would be a great choice when buiding a firewall. However, as a portable device, it is designed to work as a USB network device but OpenBSD does not contain a device mode in its USB stack. Linux provides USB gadget mode. Using the combination of ECM and RNDIS mode, most of Linux, Windows, BSD and MacOS could be supported. FreeBSD USB stack can run in device mode and provided 3 virtual network iterface templates but none of them works with Microsoft Windows[Project, 2023]. Considering the large market share of Microsoft Windows, linux is decided as the base OS of this firewall device.

Chapter 4

Packet filtering and forwarding

4.1 Design

As the general principles, all traffics passed thorugh should be checked to against evasdropping. A plaintext connection should be channeled through strong encryptions or be discarded if unable to ensure its integrity.

A modern browser can satisify most of needs to interact with internet for working. So, allowing HTTP and HTTPS connections could make most of works done when facing a hostile network environment. Thus, everyting unnecessray to make this done are blocked. The firewall rules should be carefully made to filter out as much as possible and as early as possible, because everything on all layers of the Open Systems Interconnection model (OSI model) is protentialy vulnerable and the less network traffic being passed to the following components, the less chance a vulnerability can affect the firewall.

Proxies should be used to forward network traffics. There are advantages of using a proxy than forwarding packets like a router. First, a proxy can run under user level while packets forwarding run at kernel level. When there is a vulnerability, a userland one naturally to be less harmful than a kernel one. Second, proxies are easier and also safer to configure. It's hard to configure operatng system firewalls well. When the rules are not strict enough unexpected things may happen. In example, allowing any connection to 127.0.0.1 to success without a log, could cause potential problems if a malicious dhcp 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 the packets being forwarded to the remote actually used by the protocols we are expecting. 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 that it can understand. when you are configuring a DNS proxy, the chance that your misconfiguration makes an attacker able to make http request over this proxy is very rare. They have to hack the firewall or create tunnels over those protocols to make their reverse shell work, which increased the difficulty of their operations. Third, the packets from the proxy are encoded again by Linux network stack so the fingerprinting techniques that targeting TCP ad IP properties is useless to the devices under its protection.

4.2 Implementation

Nftables is integrated in modern linux systems. It can classify and filter network traffics. 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 from users to the firewall device for DNS requests, TCP 80, and TCP 443 are allowed for HTTP and HTTPS traffics. All other packets are filtered as early as possible.

Different proxies and RPC services are created to check connections, to forward traffcs, and to apply encryptions. Details are included in the chapters of the corresponding protocol. See figs. 7.2, 8.1 and 8.2.

Chapter 5

Dynamic Host Configuration Protocol (DHCP)

5.1 Introduction

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

5.1.1 The issue

A client can expose its host name to the network from the 'client identifier' option or 'sname' field in the protocol. When in the same local network, the host name exposed by the DHCP protocol and other protocols can be used to locate the devices of the victim. If the name of a computer is straightforward enough, like "Yongbin's MacBoook Pro", the network administrator will be able to know the name of the owner of the computer from the first glance. The hardware address exposed to the network can also expose more information than just the address. Hardware address or Media Access Control (MAC) address can be used to reversly lookup the manufacturer and the manufacturer can help identify the owner. In example, the network administrator may lookup the MAC address of the a device to find out the manufacturer is company A and if the company A only sell its products in Country B and if C is the only person come from Country B then the chance that the device belongs to C would be high.

5.2 Design

Firewall should randomize both host name and MAC address. The randomized computer name should look normal to prevent being identified by potential attackers.

5.3 Implementation

NetworkManager is used to manage physical connections of the firewall. By adding 'ethernet.cloned-mac-address=random' and 'wifi.cloned-mac-address=random' to the 'connection' section of

NetworkManager.conf, the MAC address randomization is done by NetworkManager. The 'ifconfig interface link random' command can also randomize the MAC address of a interface manually.

Host names of the firewall device is random generated while starting the system. The relevant code is inside rc.local. The name is crafted to look normal. The pattern of default computer name of Microsoft Windows and common names like iPad are used. Since the implementation of the client of the protocol could also be vulnerable, the combination of AppArmor and dhclient to mitigate this problem.

Dnsmasq is used to assign 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 6

Domain Name System (DNS)

6.1 Introduction

DNS is a protocol that translates the human readable domain names to the ip addresses of the remote server. The default configuration of many devices are to use plaintexts to do requests, which could easily be monitored by simply recording the packets and be attacked by methods like DNS spoofing.

6.2 Design

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

6.3 Implementation

6.3.1 Outgoing requests

A name resoving service is created to resolve domain name queries from firewall serices. It is a local RPC service that resolve domain name queries via public DNS-over-HTTPS (DoH) services. DoH can mitigate both passive surveillance and DNS spoofing attacks[Hoffman and McManus, 2018]. DoT (DNS-over-TLS) is another DNS encryption standard that can do almost the same as DoH. The reason why use DoH instead of DoT is that DoT uses a unique server port number, 853, that can be easily filtered and identified as an encrypted DNS connection, while DoH, with the help of using the same port number and protocol as HTTPS, makes it much harder to be identified and filtered than DoT.

6.3.2 Inside firewall

A dnsmasq is started locally to resolve everything to localhost to make the firewall proxy to be applied as system-wide.

6.3.3 Incoming requests

A dnsmasq is started to resolve everything to the firewall device to make the firewall proxy to be applied to devices behind it.

Chapter 7

Transport Layer Security (TLS)

7.1 Management of certificates

7.1.1 Introduction

All modern major operating systems managed a collection of trused certificates issued by certificate authorities (CAs) they choose. Those certificates are used to prove the validity of a public key. When makeing a TLS connection, local applications check the path of the certificate provided by remote with local trusted collection to ensure secure connections.

The issue

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

Exsting solutions

SSL-pinning can prevent MITM attacks from a trusted CA when the user know the remote endpoint should use the certificate from another authority. However, configuring SSL-pinning to all the enspoints is hard and time consuming to average users and without a basic trusted environment, people unable to know whether the configuration has done correctly. Even if the configuration is correct, it is still not strange for an enspoint to switch to another CA. Thus, to a firewall, this technique cannot be used as a general solution of the issue that can be applied to all endpoints.

Removing suspecious CAs from the trusted list could protect people from MITM attacks initiated by those CA. However, some systems and devices do proide the function of untrusting a CA. In example, iPhone needs a computer to enable such configuration and most of Android variants.

7.1.2 Design

The firewall should trust only the CAs that are being widely used in the internet. People should be able to further decide which certificates to trust. One can consider untrust following when making such decisios: the CAs of the place you are staying, the CAs of the place you come from, and their enemies and allies.

7.1.3 Implementation

Since SSL-pinning can only be used on a small number of enspoints to an andvanced 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 certificate of all outgoing TLS connection remote points are checked by TLS proxy.

7.2 Mismatch between the changing best current practice and the unchanged implementation

7.2.1 Problem

The mismatch problem has been existed everywhere whenever a public protocol standard updated a fix to a problem. There are delays between a deficiency of a protocol being pubished and the majority of the implementation fixed the problem. A living example is that in the latest version of python 3.12.1 released on December 2023, as part of its builtin 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].

7.2.2 Solution

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

7.3 Proxy

MITM attack

Despite using MITM attack to insepect or protect 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 functioality. The reason is that when the firewall itself is compromised, the ability to decrypt TLS sessions will cause a disasteer to the devices behind it: the attacker will be able to easily modify the encrypted data transfered over a MITM proxy while the devices behind the firewall will alert nothing about the modification. However, if the firewall could not

do MITM attack at all, the compromised still unable to see and modify encryptions, because the certificate verification of the devices behind this firewall is still working.

Filtering certificates

The key_share extension sometimes makes the certificate invisible to the proxy. Thus, in order to filter certificates, the proxy handshakes with remote server with a sepearate connection to verify the certificate used by the remote host. A cache of trusted endpoints is used to reduce delay caused by certificate check connections.

Checking connections

The proxy should parse handshake packets. It should check algorithms provided by ClientHello and ServerHello, to ensure strong encryption. Strenghten the connection by modifing handshake packets without MITM is not possible because the HMAC of all handshake packets is used for encryption.

Implementation

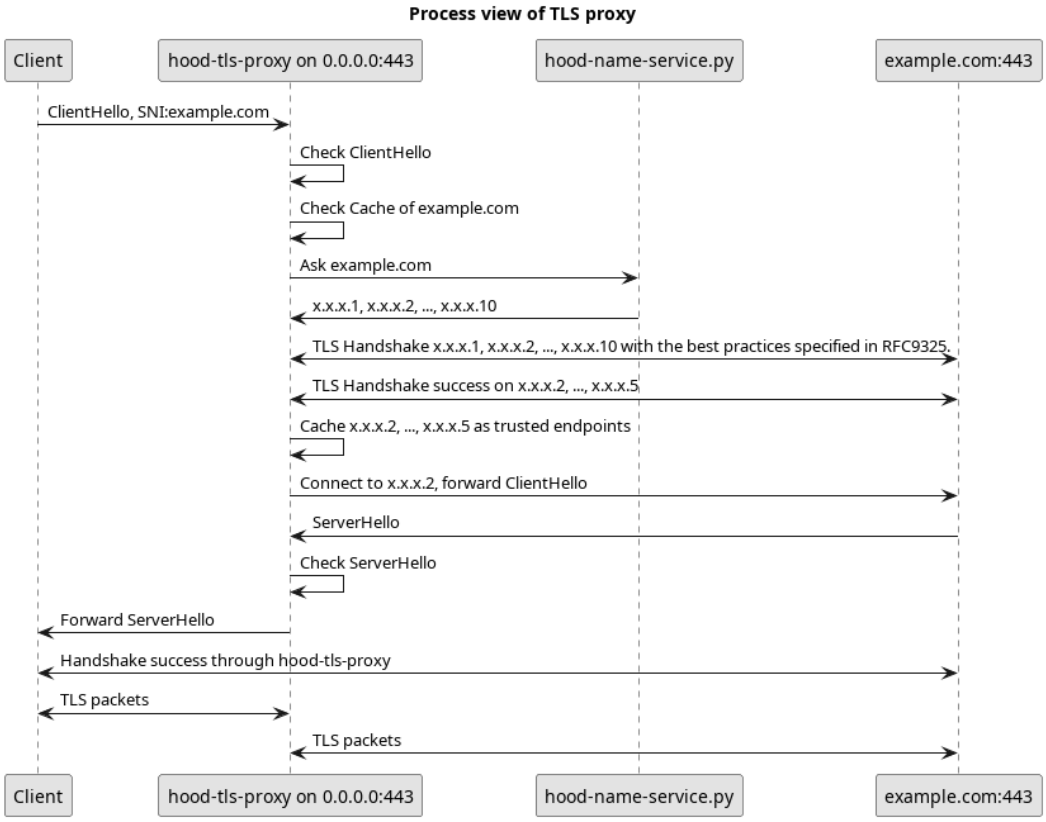

Figure 7.1. Process view of TLS proxy

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

ClientHello Upon receiving ClientHello from a client, the proxy at first check the version of the protocol, and the suppor_versions extension to see if the client supports to establish 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, proxy checks ciphersuites field, if it contains none of the value listed in 7.1 [Sheffer et al., 2022], it will be considered unable to be safe and discarded. If the ClientHello passed all of the security checks, the proxy will extract the Server Name Indication (SNI) from ClientHello to dertermine the actual destination of the request.

- 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

Table 7.1. Allowed cipher suites values

Name resolving and trused endpoint The proxy at first check if there is any cached trused endpoints available, if not it uses a RPC call to hood-name-service.py to resolve the domain name. Then TLS handshakes following the the same restrictions to the ClientHello check is 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 validate certificate of the hostname from a trusted CA of the firewall.

ServerHello 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 to use unsafe, TLS 1.0 or the cipher suite RSA_WITH_NULL_MD5 for example, the connection will be blocked.

7.4 Process view of traffics

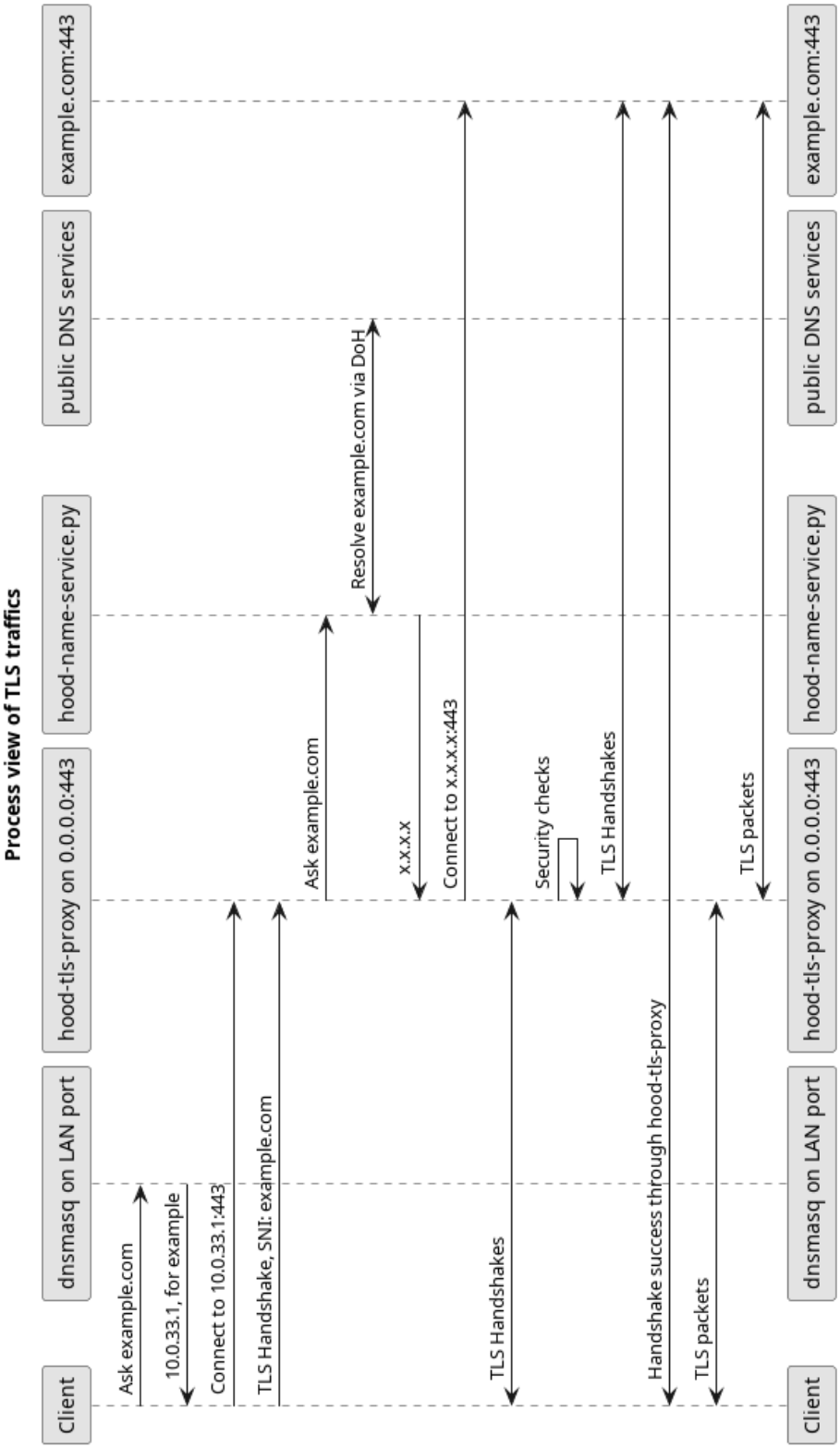

Figure 7.2. Process view of TLS traffics from user

Chapter 8

Plaintext connections

8.1 HTTP

8.1.1 Implementation

hood-http-handler.py is the proxy created to filter and protect HTTP plaintext connections. The proxy check the host name with a known list of OCSP (Online Certificate Status Protocol) and CRL (Certificate Revocatoin List) servers. Only the connections to those servers are allowed to be plaintext because those connections usually to be OCSP over HTTP connections, which have no reason to be protected. The connections to other servers are tunneled via TLS protocol through hood-tls-proxy service. See figs. 8.1 and 8.2.

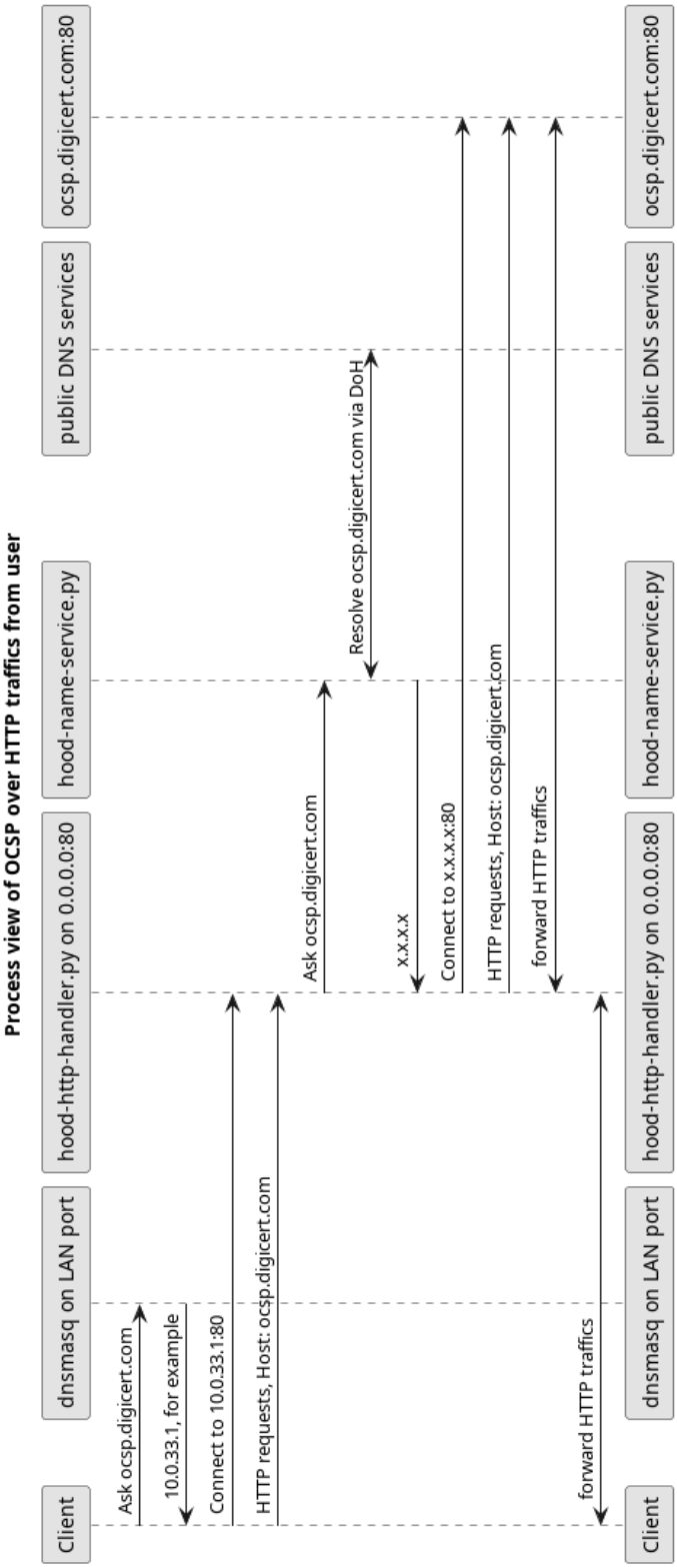

Figure 8.1. Process view of OCSP over HTTP traffics from user

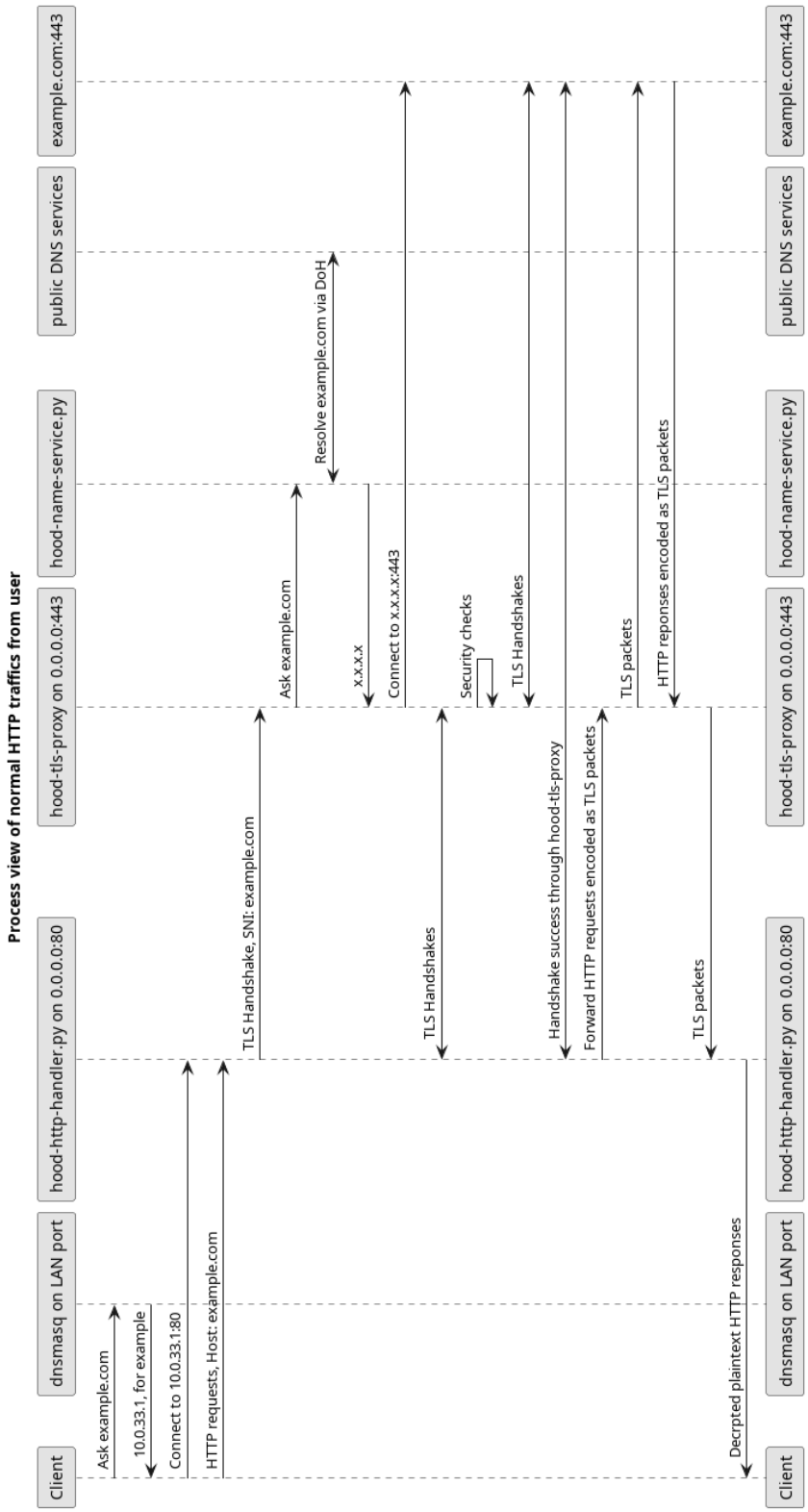

Figure 8.2. Process view of normal HTTP traffics from user

Chapter 9

Date time synchornization

9.1 Introduction

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

9.1.1 The issue

A problem of using a 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 synchornization mechanism is a must for this firewall.

Many of operating systems are pre-configured to use a NTP server which having 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) makes NTP a protocol that can be easily identified and targeted. Various attacks can be done to the protocol itself[Kwon et al., 2023]. The different in NTP behavior between different operatiing systems has been used in OS fingerprinting and tethering detection osandtether. Not to mention that the operating system being exposed by the domain name can also help an attacker to target the NTP client being used by the device.

9.2 Design

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

9.3 Implementation

The tool is implemented in a python script named hood-timesync.py. It accepts two command line arguments as listed in table 9.1. It at first try to receive a HTTP response through TLS connection from the domain name specified by host argument. If it failed, the reason could be the website is down or the system time is too different from the actual time and made a vailid 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 attemept. 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 success after the second attempt. An Epoch time is used as an 'achor' tim. Based on the assumption that time only moves forward, any time before this value will be rejected.

Name	Type	Default	Descriptoin
–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 9.1. Comand line arguments of hood-timesync.py

Chapter 10

Behavioral simulation

10.1 Introduction

In previous chapters, the fact that network activities can be used to detect the user has been proved in different protocols. However, the efforts to reduce the leak of the information on those protocols are not enought when facing a more sophisticated network analyzation. In example, the connections to Windows update servers and App Store servers can be used to detect Windows and Apple devices citeposandtether, but the firewall has no reason to stop those connections. Even if the firewall blocked all OS specific connections, the domain leaked from TLS handshakes can also tell the evasdroppers what websites a person is watching and then may infer what the person is doing. Thus, the leak of the information seems to be inevitable.

10.2 Design

A service to simuluate different network activities is designed to solve ths 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 avoid the firewall to be compromised from browser vulnerabilities, browser-based simulation tools, like puppeteer, should not be used for implementation. It should also make plaintext traffics to attaract attentions from attackers and evasdroppers to make them less focused on the actual user traffics.

10.3 Implementation

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

10.3.1 Action files

Action files are in JSON format. An action file contains an array of objects. All objects in the array has a "type" field to let the parser know how to process it. Supported object types are descripted in tables 10.1 to 10.3.

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

Table 10.1. "type" field of action file objects

Field	Type	Descriptoin
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 10.2. "properties" object of action file

Field	Type	Descriptoin
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 10.3. "task" object of action file

10.3.2 HoodExecutor

HoodExecutor is a general purpose multi-thread task executor for python. It can run and schedule tasks that has dependency relationships and delay requirements in parallel. It has a thread pool of workers to execute tasks and a single thread for scheduling delayed tasks. It is requred to simulate the paralleled requests sent out from real world browsers if without third party HTTP client based on asyncio. It is also the cornerstone of fullfilling all the execution requirements of the tasks and actions in an action file.

10.3.3 Browser

A fake browser is creted to simuluate network activities of browsing is a general purpose multi-thread task executor for python. 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 webpage to extract

the resources to be loaded and load them in parallel just like a real browser. It uses headers that used by real browsers and uses different headers on different resoures just like a real browser. Cookiesare 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, fake browser will neither render the webpage [CVE-2023-6707, CVE-2023-6351, CVE-2023-6869, CVE-2023-5170] nor evaluate scripts [CVE-2023-6702, CVE-2023-5997, CVE-2023-5728, CVE-2023-5723] used by the webpage.

10.3.4 Hood action script

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, command, and argument. For example, "hood:browser:goto:http://www.example.com". "hood:browser:" is to refer browser namespace from hood namespace, "goto" is a command inside browser namespace to tell intepreter to use browser to load a webpage. "http://www.example.com" is the parameter of the "goto" command. In this example, it represents the url of the webpage to be loaded.

hood:browser:goto command

Its usage is "hood:browser:goto URL". Its effect is to use a fake browser (See section 10.3.3) to load a URL. When the special value "random_link" is used as the argument, actor randomly picks a link from the webpage currently loaded in the browser to go to, but nothing will be done if current webpage contains not links.

hood:loop command

Its usage is "hood:loop:CONDITION:COMMAND". It is used to declare a loop. Multiple conditions can be declared in 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 33

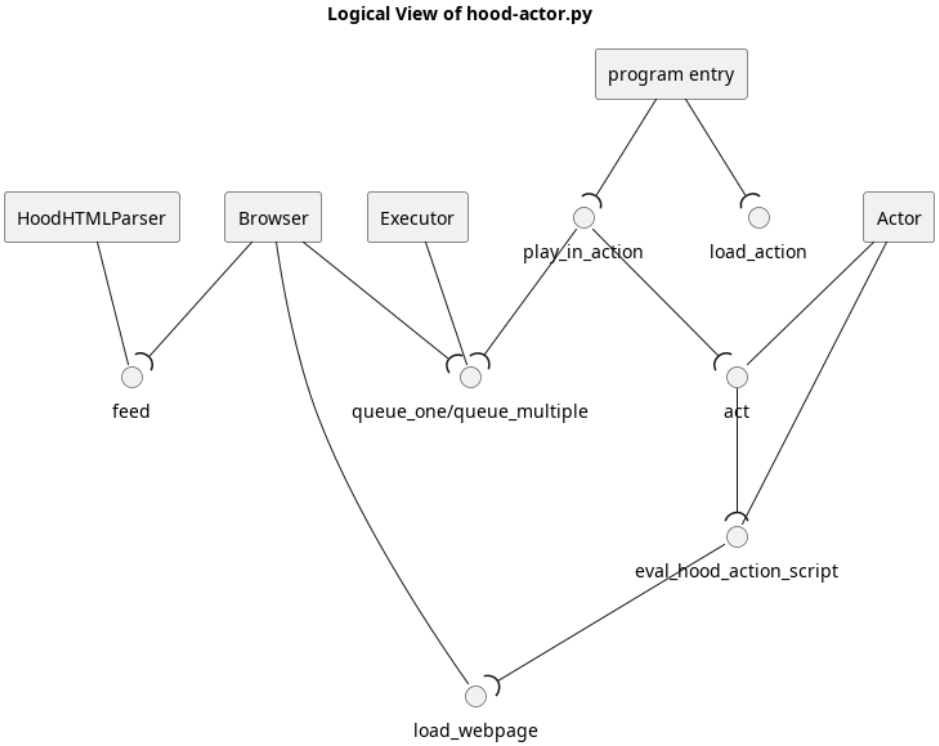

Figure 10.1. Logic view of hood-actor.py

Chapter 11

Dispatcher and firewall rules

Dispatcher is a shell script named 02-hood-dispatcher. It interacts with the changes of the network status or the system status to dynamically configure firewall rules and to start or stop netowrk services. It's callers are two sysrtemctl services: before-network.service (See appendix A.1), udev.service, and NetworkManager-dispatcher.service.

11.1 Initial state of firewall rules

The initial state of firewall rules (See appendix A.3) is basic because the deisgn 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 reject 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.

11.2 On udev add event

netdev filter

A default netdev filter will be added to both LAN and WAN interfaces. It has to be added dynamically because both of its ingress and egress filter requires the name of the device. The filter accepts only ARP and IP packets to passthrough. 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 to be filtered the less chance the packet to cause a problem.

11.3 On system startup

Upon system startup, dispatcher will be called by before-network.service. It will start two dmesg 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 iptables era, it was used for inititalizing iptables rules.

Since the udev event could be triggered at very early stage of the startup, so early that the nftables service may not available and adding rules may fail. Dispatcher try 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.

11.4 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. Devie path is the realpath 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 local 68 port to remote 67 port and to be received from remote 67 port to local 68 port for DHCP protocol.

LAN port

For LAN port network interfaces, the dispatcher at first pick 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 start. LAN subnet randomization could help increase the difficulty of a malicious code to detect the firewall from LAN port.

After randomization, following firewall rules is 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 local 67 port to remote 68 port and to be received from remote 68 port to local 68 port. For the dnsmasq instance.

Table 11.1. 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 the IP address of the LAN port for DNS requests.

11.5 On NetworkManager-dispatcher up event

Dispatcher only reacts to this event for the WAN port. It at first waits for DHCP to finish. After DHCP is done, dispatcher removes DHCP related accept rules from nftables and add a rule to accept outbound TCP connections from 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, dispatcher restarts the hood-network-services.service (appendix A.2). It will also check the existance of /do_upgrade or /run_once files to do system upgrade or to run the script.

11.6 On NetworkManager-dispatcher down event

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

Chapter 12

Installation

Chapter 13

Reasons of not to use

In most cases, the reason to not use something is to reduce the attack surface.

13.1 IPV6

13.2 WiFi

13.3 Bluetooth

13.4 GPU

13.5 Honeypot

Chapter 14

Conclusions

14.1 Possible improvements for future

14.1.1 LSM and seccomp

Use LSMs, like AppArmor, and seccomp to protect all as much process as possible. The programs that do not support seccomp can be changed by LD_LOAD_LIBRARY.

14.1.2 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.

14.1.3 Network stack fingerprinting

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

14.1.4 Further use to libcomposite

Libcomposite has the ability to make the device to 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 comptuer will be able to boot a live system from it. For a computer that 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.

Chapter 15

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

15.1 Malicious hardwares

Normal attackers will use network, bluetooth or wifi for communications between malware and the host. In example, casting the screen to another device via wifi or sending keyboard inputs via bluetooth. Those kind of attacks can be detected by a RF detector. RF sheilding fabrc, RF detectors, and signal jammers could be used to fight against such kind of attacks.

However, There are still other ways exist. Some even without wireless communications. Despite I have already bought a RF detector to alert me wireless communications between unknown hardwares, unknown attackers still managed to monitor my progress by modified hardwares. Take the devices that I am using to develop this project for example. Raspberry 4B has a usb chip that can communicate to 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 from 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 be happened to the portable screen that I am using now. I have found two counter measurements to this kind of attack: One is to use 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 male USB-A port to prevent data communications.

15.2 Sounds

Another attack that can bypass a rfkilled computer is to use AI / ML to identify the sounds of keyboard hits. The detected types can be send 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 because of the same technology being used for hackers.

Chapter 16

A chapter title which will run over two lines — it's for testing purpose

16.1 The first section

16.2 The second, math section

Theorem 1 (Residue Theorem). Let f be analytic in the region G except for the isolated singularities $a_1, a_2, \ldots, a_m$. If γ is a closed rectifiable curve in G which does not pass through any of the points a_k and if $\gamma \approx 0$ in G then

$$\frac{1}{2\pi i}\int_\gamma f = \sum_{k=1}^{m} n(\gamma; a_k)\mathrm{Res}(f; a_k).$$

Theorem 2 (Maximum Modulus). *Let G be a bounded open set in $\mathbb{C}$ and suppose that f is a continuous function on G^- which is analytic in G. Then*

$$\max\{|f(z)| : z \in G^-\} = \max\{|f(z)| : z \in \partial G\}.$$

16.3 A very very long section, titled "The third section", with a rather short text alternative (third)

Some Test

```
1 import IntSpec, ItemSpec;
2
3 sort cart;
4
5 constructors
6 create() ⟶ cart;
7 insert(cart, item) ⟶ cart;
8 observers
```

```
amount(cart) ⟶ int;
transformers
delete(cart, item) ⟶ cart;

axioms
forall c: cart, i, j: item

amount(create()) = 0;
amount(insert(c,i)) = amount(c) + price(i);
delete(create(),i) = create();
delete(insert(c,i),j) =
if (i == j) c
else insert(delete(c,j),i);
end
```

As you can easily see from the above listing Baresi et al. [2007a] define something weird based on the BPEL specification [Andrews et al., 2003].

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 11) 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 nftables.conf

```
1  #!/usr/sbin/nft -f
2
3  flush ruleset
4
5  table ip filter {
6    chain input {
7      type filter hook input priority filter; policy drop;
8      iif lo accept
9      ip daddr 127.0.0.1/8 log prefix "[HOOD D]" flags all drop
10     ip saddr 127.0.0.1/8 log prefix "[HOOD D]" flags all drop
11     meta l4proto udp ct state {established, related} log prefix "[HOOD A
           ]" flags all accept
12     ct state {established, related} accept
13     icmp type {destination-unreachable} icmp code {frag-needed} accept
```

```
14      log prefix "[HOOD D]" flags all drop
15    }
16    chain forward {
17      type filter hook forward priority filter; policy drop;
18      log prefix "[HOOD D]" flags all drop
19    }
20    chain output {
21      type filter hook output priority filter; policy drop;
22      oif lo accept
23      ip daddr 127.0.0.1/8 log prefix "[HOOD D]" flags all drop
24      ip saddr 127.0.0.1/8 log prefix "[HOOD D]" flags all drop
25      meta l4proto udp ct state {established, related} log prefix "[HOOD A
            ]" flags all accept
26      ct state {established, related} accept
27      icmp type destination-unreachable icmp code {frag-needed} accept
28      log prefix "[HOOD D]" flags all drop
29    }
30  }
31
32  table ip6 filter {
33    chain ingress {
34      type filter hook input priority filter; policy drop;
35      log prefix "[HOOD D]" flags all drop
36    }
37    chain prerouting {
38      type filter hook input priority filter; policy drop;
39      log prefix "[HOOD D]" flags all drop
40    }
41    chain input {
42      type filter hook output priority filter; policy drop;
43      log prefix "[HOOD D]" flags all drop
44    }
45    chain forward {
46      type filter hook forward priority filter; policy drop;
47      log prefix "[HOOD D]" flags all drop
48    }
49    chain output {
50      type filter hook output priority filter; policy drop;
51      log prefix "[HOOD D]" flags all drop
52    }
53    chain postrouting {
54      type filter hook forward priority filter; policy drop;
55      log prefix "[HOOD D]" flags all drop
56    }
57  }
58
59  table bridge filter {
```

```
60   chain ingress {
61     type filter hook input priority filter; policy drop;
62     log prefix "[HOOD D]" flags all drop
63   }
64   chain prerouting {
65     type filter hook prerouting priority filter; policy drop;
66     log prefix "[HOOD D]" flags all drop
67   }
68   chain input {
69     type filter hook output priority filter; policy drop;
70     log prefix "[HOOD D]" flags all drop
71   }
72   chain forward {
73     type filter hook forward priority filter; policy drop;
74     log prefix "[HOOD D]" flags all drop
75   }
76   chain output {
77     type filter hook output priority filter; policy drop;
78     log prefix "[HOOD D]" flags all drop
79   }
80   chain postrouting {
81     type filter hook postrouting priority filter; policy drop;
82     log prefix "[HOOD D]" flags all drop
83   }
84 }
85
86 table netdev filter {
87 }
88
89
90 table arp filter {
91   chain input {
92     type filter hook input priority filter; policy drop;
93     log prefix "[HOOD A]" flags all accept
94   }
95   chain output {
96     type filter hook output priority filter; policy drop;
97     log prefix "[HOOD A]" flags all accept
98   }
99 }
```

Glossary

Bibliography

Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. Business Process Execution Language for Web Services, Version 1.1. BPEL4WS specification, May 2003.

L. Baresi, D. Bianculli, C. Ghezzi, S. Guinea, and P. Spoletini. Validation of web service compositions. *IET Software*, 1(6):219–232, 2007a. doi: 10.1049/iet-sen:20070027. URL `http://link.aip.org/link/?SEN/1/219/1`.

Luciano Baresi, Domenico Bianculli, Carlo Ghezzi, Sam Guinea, and Paola Spoletini. A timed extension of WSCoL. In *Proceedings of the IEEE International Conference on Web Services (ICWS 2007)*, pages 663–670. IEEE Computer Society Press, July 2007b.

Domenico Bianculli and Carlo Ghezzi. Monitoring conversational web services. In *Proceedings of the 2nd International Workshop on Service-Oriented Software Engineering (IW-SOSWE'07), co-located with ESEC/FSE 2007*, pages 15–21, New York, NY, USA, September 2007. ACM Press.

Domenico Bianculli, Carlo Ghezzi, and Paola Spoletini. A model checking approach to verify BPEL4WS workflows. In *Proceedings of the 2007 IEEE International Conference on Service-Oriented Computing and Applications (IEEE SOCA 2007)*, pages 13–20. IEEE Computer Society Press, June 2007a.

Domenico Bianculli, Radu Jorca, Walter Binder, Carlo Ghezzi, and Boi Faltings. Automated dynamic maintenance of composite services based on service reputation. In *Proceedings of ICSOC'07, International Conference on Service-Oriented Computing*, volume 4749 of *Lecture Notes in Computer Science*, pages 449–455. Springer-Verlag, September 2007b.

Domenico Bianculli, Angelo Morzenti, Matteo Pradella, Pierluigi San Pietro, and Paola Spoletini. Trio2Promela: a model checker for temporal metric specifications. In *29th International Conference on Software Engineering (ICSE'07 Companion)*, pages 61–62, Los Alamitos, CA, USA, May 2007c. IEEE Computer Society. ISBN 0-7695-2892-9. doi: http://doi.ieeecomputersociety.org/10.1109/ICSECOMPANION.2007.79. Research Demo.

Domenico Bianculli, Paola Spoletini, Angelo Morzenti, Matteo Pradella, and Pierluigi San Pietro. Model checking temporal metric specification with Trio2Promela. In *Proceedings of International Symposium on Fundamentals of Software Engineering (FSEN 2007)*, volume 4767 of *Lecture Notes in Computer Science*, pages 388–395. Springer Verlag, April 2007d.

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.

FORTINET. Security profiles. In *FortiOS - Administration Guide*, chapter 9, pages 1026–1029. June 2021. URL `http://docs.freebsd.org/en/books/handbook/usb-device-mode/`.

Python Software Foundation. ssl - tls/ssl wrapper for socket objects. In *Python 3.12.1 documentation*. December 2023. URL `http://docs.freebsd.org/en/books/handbook/usb-device-mode/`.

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

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`.

David C. Luckham, Friedrich W. von Henke, Bernd Krieg-Brueckner, and Olaf Owe. *ANNA: a language for annotating Ada programs*. Springer-Verlag, New York, NY, USA, 1987. ISBN 0-387-17980-1.

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`.

The FreeBSD Documentation Project. Chapter 28.usb device mode/usb otg. In *FreeBSD Handbook*, chapter 28. July 2023. URL `http://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/rfc6176`.

Sean Turner and Tim Polk. Prohbiting secure socket layer (ssl) version 2.0. RFC 6176, RFC Editor, March 2011. URL `https://www.rfc-editor.org/rfc/rfc6176`.

1