
Design and implementation of a firewall device

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

Contents	vii
List of Figures	ix
List of Tables	xi
1 Introduction	1
1.1 Existing commercial hardwares	1
2 Operating system	3
3 Packet filtering and forwarding traffics	5
3.1 Design	5
3.2 Implementation	6
4 Dynamic Host Configuration Protocol (DHCP)	7
4.1 Introduction	7
4.1.1 The issue	7
4.2 Design	7
4.3 Implementation	7
5 Domain Name System (DNS)	9
5.1 Introduction	9
5.2 Design	9
5.3 Implementation	9
5.3.1 Outgoing requests	9
5.3.2 Inside firewall	9
5.3.3 Incoming requests	10
6 Transport Layer Security (TLS)	11
6.1 Implementation	11
7 Plaintext connections	13
7.1 HTTP	13
7.1.1 Implementation	13

8 Management of certificates	15
8.1 Introduction	15
8.1.1 The issue	15
8.1.2 Existing solutions	15
8.2 Design	16
8.3 Implementation	16
9 Date time synchronization	17
9.1 Introduction	17
9.1.1 The issue	17
9.2 Design	17
9.3 Implementation	17
10 Conclusions	19
10.1 Possible improvements for future	19
10.1.1 LSM and seccomp	19
10.1.2 Compile time hardening	19
10.1.3 Network stack fingerprinting	19
11 The attacks that I have encountered during the time I was working on this thesis	21
11.1 Malicious hardware	21
11.2 Sounds	21
12 Short title	23
12.1 The first section	23
12.2 The second, math section	23
12.3 third	23
A Some retarded material	25
A.1 It's over...	25
Glossary	27
Bibliography	29

Figures

Tables

Chapter 1

Introduction

The goal of this work is to increase the chance of survival of the user from hacking, evasdropping, and digital fingerprinting. The device will work as a strict firewall which limits network activities and will also apply encryption standards to increase the difficulty of eavesdropping, and also to reduce the attack surface of fingerprinting. The device will run a TLS proxy to harden TLS protocol. It will use 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 hardened latest version of firefox while writing this thesis). This could also work as a mitigation of CPU/BIOS level threats: Firmware malwares, bootkits, and doubtful proprietry technologies like Intel ME and the AMD PSP. Second, it will also provide convenience to the user: the configuration of this portable device will be 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 will be under the restriction of the firewall. In example, people cannot untrust a built-in trusted root certificate from iPhone via Settings app.

1.1 Existing commercial hardwares

Commercial firewalls are expensive. In switzerland, the price of a entry level firewall is more than twice of a raspberry pi. Raspberry Pi being used in this work could be replaced with some cheaper alternatives, making the cost will be even lower. Commercial firewalls also do not provide a screen to display network activities in real-time.

Chapter 2

Operating system

Due to its security-focused nature, OpenBSD would be a great choice when building 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 interface 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 3

Packet filtering and forwarding traffics

3.1 Design

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

A modern browser can satisfy 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 fire-wall 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 will be encoded again by Linux network stack so the TCP fingerprinting will be less useful to the devices under its protection.

3.2 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 traffics, and to apply encryptions. Details are included in the chapters of the corresponding protocol.

Chapter 4

Dynamic Host Configuration Protocol (DHCP)

4.1 Introduction

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

4.1.1 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.

4.2 Design

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

4.3 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 will be 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 will also tell 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 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.

5.2 Design

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

5.3 Implementation

5.3.1 Outgoing requests

A name resolving service is created to resolve domain name queries from firewall services. 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.

5.3.2 Inside firewall

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

5.3.3 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 6

Transport Layer Security (TLS)

6.1 Implementation

A TLS proxy is created to forward and protect TLS connections. The proxy will not decrypt TLS sessions, so that when the firewall itself is hacked will not cause a disaster to the user because the certificate verification is still working on the devices behind this firewall. If I give the proxy the ability to decrypt TLS sessions, when the firewall is hacked, the hacker can easily modify the encrypted data transferred over TLS while the devices behind the firewall will still consider the modified data are coming from a trusted connection. To do this, the proxy will parse handshake packets. It will check algorithms provided by ClientHello and ServerHello, if the negotiation result could not be using strong encryptions, the connection will be blocked. Altering handshake packets without MITM is not viable because the HMAC of all handshake packets is used for encryption. The key_share extension sometimes makes the certificate invisible to the proxy. Thus, in order to filter certificates, the proxy will handshake with remote server with a separate connection to verify the certificate used by the remote host. A cache of trusted endpoints is used to reduce delay caused by TLS handshakes used for certificate check from proxy.

Chapter 7

Plaintext connections

7.1 HTTP

7.1.1 Implementation

A proxy is 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 by TLS. The connections to other hosts will be converted to HTTPS connections. If a remote host accepts only HTTP connections, the connection attempt will be failed.

Chapter 8

Management of certificates

8.1 Introduction

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

8.1.1 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 things will be worse if without TLS.

8.1.2 Existing 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 endpoints is hard and time consuming to average users and without a basic trusted environment, it will be unable to certain whether the configuration has 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 of the issue that can be applied to all endpoints.

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

8.2 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.

8.3 Implementation

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

Chapter 9

Date time synchornization

9.1 Introduction

Network Time Protocol (NTP) is used by most of modern systems to synchornize the clock with a remote server.

9.1.1 The issue

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 and the unique domain names makes NTP a protocol that can be easily identified and targeted. Various attacks can be done to the protocol itself[Kwon et al., 2023]. 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.

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.

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. Usually, in order to hide, the attacks targeted to HTTP/HTTPS protocol do not want to make mistakes in the header of its attack vector.It's hard to say whether this request is made for time synchornization.

9.3 Implementation

Since TLS needs time to be correct to validate certificates, a plaintext HTTP request is made to do an initial synchornization. After the first step, the system clock is considered to be correct

enought for making a TLS connection. So the second step is to extract date time from the header of a HTTPS response.

Chapter 10

Conclusions

10.1 Possible improvements for future

10.1.1 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`.

10.1.2 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.

10.1.3 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.

Chapter 11

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

11.1 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.

11.2 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 be because of the same technology being used for hackers.

Chapter 12

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

12.1 The first section

12.2 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, \dots, 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

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

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

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

12.3 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()  $\longrightarrow$  cart;
7 insert(cart, item)  $\longrightarrow$  cart;
8 observers
```

```
9 amount(cart)  $\longrightarrow$  int;
10 transformers
11 delete(cart, item)  $\longrightarrow$  cart;
12
13 axioms
14 forall c: cart, i, j: item
15
16 amount(create()) = 0;
17 amount(insert(c,i)) = amount(c) + price(i);
18 delete(create(),i) = create();
19 delete(insert(c,i),j) =
20 if (i == j) c
21 else insert(delete(c,j),i);
22 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

Some retarded material

A.1 It's over...

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.

Paul Hoffman and Patrick McManus. 8. privacy considerations. In *DNS Queries over HTTPS (DoH)*, chapter 8. Internet Engineering Task Force (IETF), 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.

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/>.