**Case Study 2 – Pen-testing Analysis**

**Media storage service**



Project Members & Student Numbers:

* Francisco Marcó (4467752)
* Sava Vasilev (4663438)
* Kaloyan Andreev (4408020)

Elaboration Date: 13/06/2022

Group 10

Table of content

[1. Introduction 2](#_Toc106045728)

[2. Reconnaissance 2](#_Toc106045729)

[2.1 Docker Instance 9](#_Toc106045730)

[2.2 Network Diagram 11](#_Toc106045731)

[3. Open ports & Vulnerabilities 11](#_Toc106045732)

[4. Services 12](#_Toc106045733)

[5. Brute-force attack 13](#_Toc106045734)

[6. Social Engineering 14](#_Toc106045735)

[7. Conclusion 14](#_Toc106045736)

[Sources 15](#_Toc106045737)

# Introduction

The following document aims to explain and narrate the steps taken to perform the penetration testing on Group 06, per our agreement. It will go in depth about the findings of the test, possible security risks, possible services the network might be running, brute-force attacks results and possible social engineering approaches.

The client has agreed for their infrastructure to be tested with, and no malicious activity was carried during the test.

The test was carried on a Kali Linux VM, connected through the VPN to the client’s infrastructure.

# Reconnaissance

During this preliminary phase, the goal was to investigate the infrastructure. Specifically:

* **IP Ranges**
* **IP Addresses of important IT infrastructure elements**

Firstly, it was noticed that the OpenVPN client had assigned us an IP for the Local Desktop machine the VM was running on.

Imagen que contiene Diagrama

Descripción generada automáticamente

After that, we determined the IP address of the “attacker” machine.

Texto

Descripción generada automáticamente

This revealed the existence of the 10.0.2.0/24 network. Due to the given netmask and broadcast address, we can determine how big the network is and where to look for other hosts.

Now, after knowing our own IP, we decided to do a Nmap scan of the 10.0.8.1 host (given how it’d be normal to have a .1 host due to us being .2).

Texto

Descripción generada automáticamente

With this information, we determined that this host was running a web server. We connected to this address with a web-browser and were welcomed to the pfSense login screen.

To further map out the network, we ran a Nmap ping sweep scan on both the 10.0.2.0/24 and 10.0.8.0/24 network.

Texto

Descripción generada automáticamente

This proved unsuccessful, as no hosts were found by the scan. Because it was extremely unlikely the server was empty, we ran some more tests.

Texto

Descripción generada automáticamente

Checking our IP routes, we found another new network. After researching, we determined it was a Docker instance as 172.17.0.0/16 is the default range used by Docker. It also showed us that there’s a host at 10.0.2.2 which serves as our gateway to its network range.

Following this line of thought, we ran a Nmap scan to determine more about this host.

Texto, Carta

Descripción generada automáticamente

This showed us the host’s OS (Windows) and its open port. Some ports caught our attention, mainly port 9100, which had a strange service description.

Because so far we hadn’t found any web-applications or other services, we started scanning the network using a different tool: netdiscover.

Tabla

Descripción generada automáticamente

Luckily, now we found two more hosts in the 10.0.2.0/24 network. We ran scans for each to determine OS and open ports.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

These two scans revealed the same open ports as the 10.0.2.2 host. After researching, we found some info about possible services running on these but it was not clear what services and how to access them.

Because we hadn’t found much information, we decided to use a different tool to scan these hosts. Specifically, we were interested in knowing more about the ports open on these hosts. We decided to use LEGION, a tool provided in the Kali Linux installation.

Firstly, we ran the scan on 10.0.2.3:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Now, we got a lot more of information about the open ports. Despite most of them being used by default system operations, there were some that caught our attention:

* **Port 27036 — Valve Steam In-Home Streaming Service**
* **Port 58795 — Node.js Express framework**

Finally, we had gotten a hint of a web-application on the network. We ran the same scan on the other hosts after.

* **10.0.2.2**Interfaz de usuario gráfica, Texto

  Descripción generada automáticamente Interfaz de usuario gráfica, Aplicación

  Descripción generada automáticamente
* **10.0.2.4:** Interfaz de usuario gráfica, Texto

  Descripción generada automáticamente

We noticed how there was a big similarity between the open ports between hosts. All of them were running the “Valve Steam In-home Streaming” service, an unknown “jetservice” (port 9100) service, and both 10.0.2.2 and 10.0.2.3 had and http port open for a JavaScript framework.

However, there was no port 80 or 443 for a web-server or web-application (besides pfSense). After researching, and using tools like nslookup and telnet, we determined that the current state of the architecture does not have a live web-application. This final finding concluded the Reconnaissance phase.

## 2.1 Docker Instance

It was mentioned before that a Docker Instance was found during the Reconnaissance phase. This lead to some research, which lead us to find that our Kali Linux VM had also been assigned an IP in the 172.17.0.0/16 network: 172.17.0.1.

After some more research, we ran a scan of our domain sockets (specifically TCP packets) to see how our machine communicates with other hosts, networks, and services. To do so we ran the command “ss” and found the following network activity related to the found Docker Instance.

Imagen que contiene Patrón de fondo

Descripción generada automáticamente

These findings didn’t clear things up however, as we were unable to properly assess and analyze the information gained. Many attempts were made to try and connect or reach these addresses or ports, but none were fruitful. But we weren’t close to giving up, so we decided to run more network scans to see if we could get something.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Unfortunately, this scan didn’t give us any new things to interact with or research. We had no luck figuring out the Docker Instance running on the server, so after using other different tools for analysis, we determined that we were unable to determine the services run in this Docker Instance.

## 2.2 Network Diagram

After fully mapping out the infrastructure, we devised a Network Diagram to have a clearer view of our discoveries and to illustrate the pen-testing reconnaissance phase.

Diagrama

Descripción generada automáticamente

We connect to the network using “Desktop”, where the OpenVPN client assigns us the IP 10.0.8.2. There, the pfSense Firewall lets our Kali Linux VM to access the 10.0.2.0/24 network.

# Open ports & Vulnerabilities

After the reconnaissance phase, we decided to turn our attention to the open TCP ports in these hosts. We’ve outlined the most relevant open ports which could pose a vulnerability if they were accessible by a malign agent.

|  |  |  |  |
| --- | --- | --- | --- |
| Host | Port | Service | Vulnerability |
| 10.0.8.1 | 53 | DNS. | Targeted attacks, buffer overflow, DoS. |
| 10.0.2.3; 10.0.2.4 | 9100 | Prometheus Monitoring / HP Printer. | Remote Firmware Update, remote usage, bypass print filters. (\*) |
| 10.0.2.3; 10.0.2.4 | 445 | TCP/IP MS Networking. | Trojans, Worms, Remote Code Execution, buffer overflow. |
| 10.0.2.3; 10.0.2.4 | 3389 | Remote Desktop. | Trojans, DoS, Remote Code Execution. |
| 10.0.2.3; 10.0.24 | 135 | Remote Procedure Call / Exchange client. | DoS, buffer overrun, Worms. |
| 10.0.2.2; 10.0.2.3; 10.0.2.4 | 27036 | Valve Steam  In-home streaming. | Privilege scalation. |

(\*) Applicable to HP Printer only

# Services

Because of the shortcomings on the reconnaissance phase, it was hard to determine exactly what services are being run in the infrastructure.

With the information gained during the analysis we can determine the following services running:

* **DNS / DHCP server (pfSense)**
* **Monitoring (Prometheus) / Printer (HP)**
* **Client / Server Application**
* **Steam In-home streaming**

However, doing this assessment based on port availability is tricky. For example, port 9100 is typically used by printers, but there’s an open-source monitoring solution called Prometheus which also makes use of it. However, port 9090, used by the Prometheus server, was not found on any host. Furthermore, the port used for the Client / Server Application, could also be used for a VPN connection instead.

# Brute-force attack

Considering the open ports on these hosts, we decided to attempt a brute-force attack using xHydra using the RDP protocol, as these were open to allow remote access to the hosts.

The attack was performed using a custom-made username list, which contained probable username options. It includes the names of our clients, Group 06, as it was a likely match for the username.

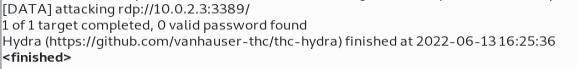
Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

As for the password list, we used the default list given in the /usr/share/wordlist/ directory on our Kali Linux installation. These were the results after running an attack on hosts 10.0.2.2, 10.0.2.3 and 10.0.2.4.

Texto

Descripción generada automáticamente



Texto

Descripción generada automáticamente

After these results, we determined that the passwords used for RDP connections are safe enough to not be figured out by a standard run of Hydra.

# Social Engineering

For our pen-testing purposes, we did not perform any Social Engineering. We felt it went away from our scope and could be interpreted as bad-faith.

However, Social Engineering is usually the most effective way to penetrate a network. Phishing is ever so common and dangerous around professional workspaces, and it’s more efficient if the attacker impersonates an authoritative figure. For example, if we reached to our client (Group 06) by email impersonating their tutor, we could get a chance at learning compromising information.

Usually, the weak link in an IT infrastructure is not a port, server, or machine. It’s ourselves. Nowadays, this problem is being addressed by many companies in the world by providing training and methods of recognizing phishing and malign emails.

# Conclusion

In conclusion, we have determined how our client’s infrastructure is composed and any vulnerabilities they have currently. However, we have failed in properly assessing all infrastructure components and services. We do not know if this is due to inexperience, lack of services, or both.

Despite the vulnerabilities we found, we do not think they would pose a major security risk for the infrastructure. It would only be the case if the mentioned ports were accessible from an outside network, but this test was carried as if the attacker already was inside and had full unregulated access.

As for security measures, we could suggest analyzing which ports are relevant to have open and which are not. Most could be closed or at the least investigated as they are used for fairly standard OS tasks.

# Sources

<https://www.speedguide.net/ports.php>

<https://www.cve.org/>

<https://book.hacktricks.xyz/network-services-pentesting/135-pentesting-msrpc>

<https://book.hacktricks.xyz/network-services-pentesting/pentesting-rdp>

https://book.hacktricks.xyz/network-services-pentesting/pentesting-dns