Vulnerability Assessment and Exploitation Write-Up

Machine: Chrome – TryHackMe

Prepared by: Melisa Sarıtaş

Table of Contents

1. Introduction

Google Chrome is a web browser that stores saved credentials in an SQLite database file called "Login Data", located within the user's profile directory. To protect sensitive information, Chrome encrypts these credentials using Windows' Data Protection API (DPAPI). DPAPI ensures that only the specific user who saved the credentials can decrypt them by generating a master key, which is further secured using the user's Windows password. Since DPAPI encryption is tightly linked to a Windows user account, decrypting the credentials requires access to both the encrypted master key and the corresponding authentication data.

This challenge[1] provides a collection of pre-captured files, a pcapng file containing network traffic from a domain environment. The goal is to analyze the file and extract the necessary components for decrypting the Chrome credentials. Since authentication data may be transmitted over the network, the pcapng file could reveal encryption keys or hashes needed to bypass DPAPI protection.

To solve the challenge, Kali Linux[2] is used as the host machine due to its extensive set of built-in security tools. It is downloaded the challenge file from the page[1] and transferred it to the Kali Linux machine. Once the files were available, it is extracted to access the pcapng file.

2. Solution Overview

The solution consists of six steps: analysis of the packets, decompiling the executable, decryption of the encrypted files, cracking the DPAPI password, decryption of the master key, and login credentials. Each of these steps will be explained in detail in the following sections.

2.1. Analysis of the Packets

Open the pcapng file with wireshark[3], which is a tool already installed in Kali Linux. Look at the packets in general. It can be seen that there are lots of smb packets. Extract them by navigating "*file>export>objects>SMB".* There are smb shares available which are *"transfer.exe"* and *"encrypted_files"*. Click the button "*save all"* for further analysis.

2.2. Decompiling the Executable

Open *"transfer.exe"* with the default archive manager. There are files named *".text"* and *".reloc"*. Open the *".text"* file with a text editor and investigate the content. Notice references to the .NET Framework[4], indicating that the executable is built using this technology. These findings are shown in Figure 1.

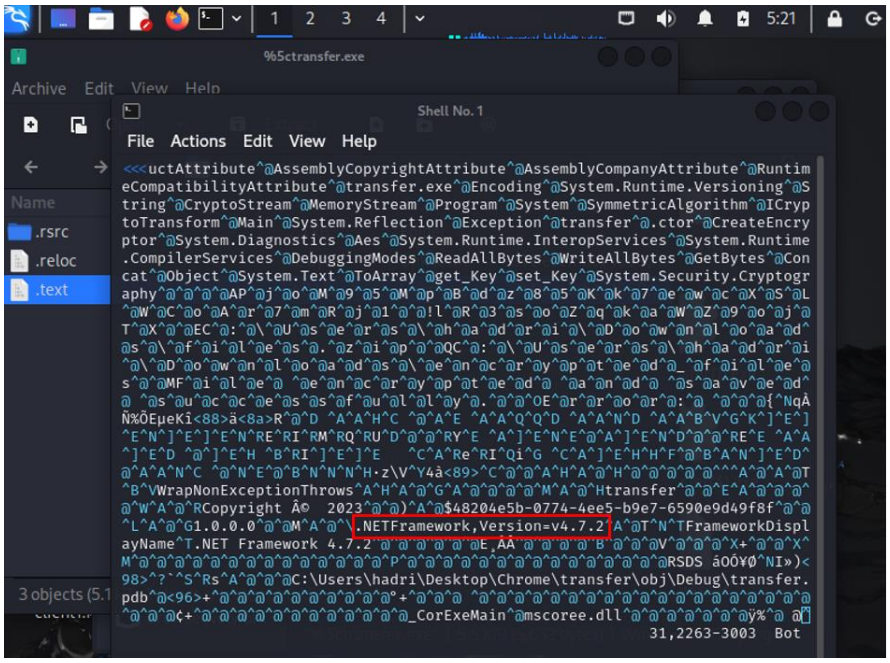

Figure 1: The content of *.text*

It will be used ILSpy[5], a decompiler, to see what is written inside of the file. Use Command 1 to download the necessary tools.

sudo apt-get install -y dotnet-sdk-8.0

dotnet tool install --global ilspycmd

Command 1

Run Command 2 to decompile the file.

ilspycmd ./%5ctransfer.exe

Command 2

From the output, take note of the AES IV and Key to use for the next step. The output is shown in Figure 2.

```
File  Actions  Edit  View  Help
assembly: Guid("48204e5b-0774-4ee5-b9e7-6590e9d49f8f")]
assembly: AssemblyFileVersion("1.0.0.0")]
assembly: TargetFramework(".NETFramework,Version=v4.7.2", FrameworkDisplayName = ".NET Framework
.7.2")]
assembly: AssemblyVersion("1.0.0.0")]
ublic class Program

        private static void Main()
        {
                try
                {
                        byte[] bytes = Encoding.UTF8.GetBytes("PjoM95MpBdz85Kk7ewcXSLWCoAr7mRj1");
                        byte[] bytes2 = Encoding.UTF8.GetBytes("lR3soZqkaWZ9ojTX");
                        string path = "C:\\Users\\hadri\\Downloads\\files.zip";
                        byte[] array = File.ReadAllBytes(path);
                        byte[] bytes3;
                        using (Aes aes = Aes.Create())
                        {
                                aes.Key = bytes;
                                aes.IV = bytes2;
                                ICryptoTransform transform = aes.CreateEncryptor(aes.Key, aes.IV);
                                using MemoryStream memoryStream = new MemoryStream();
                                using (CryptoStream cryptoStream = new CryptoStream(memoryStream,
ransform, CryptoStreamMode.Write))
                                {
                                        cryptoStream.Write(array, 0, array.Length);
                                }
                                bytes3 = memoryStream.ToArray();
                        }
                        string path2 = "C:\\Users\\hadri\\Downloads\\encrypted_files";
```

Figure 2: The content of the decompiled file

2.3. Decryption of the Encrypted Files

There is a file called *"encrypted_files"*. Decrypt it using cyberchef[6] and aes hashes that is found from the previous step. Choose UTF8 for the Key and IV, enter the values and set the input and output as Raw. The CyberChef interface is shown in Figure 3.

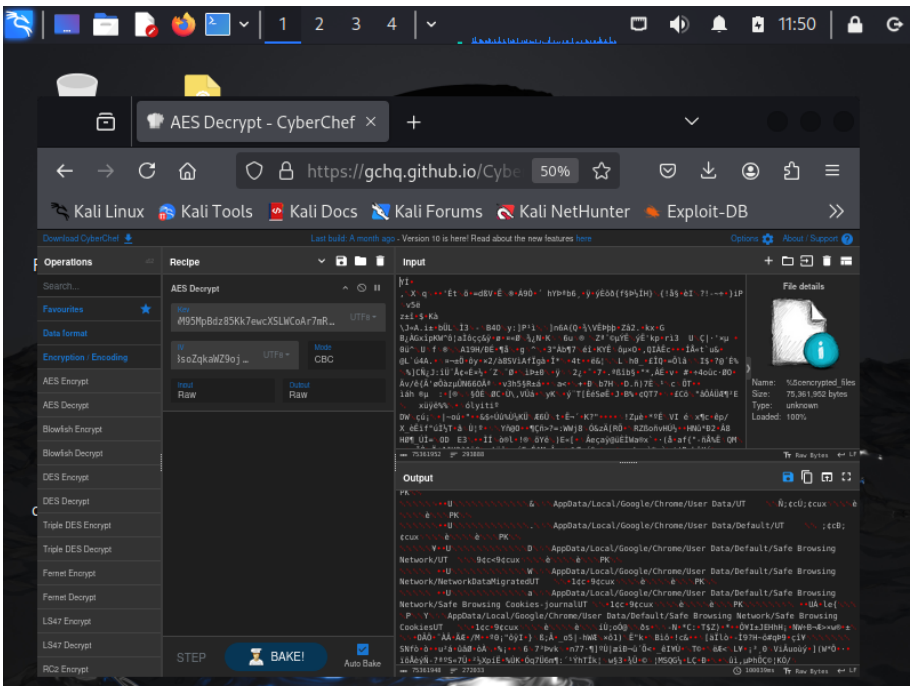

Figure 3: The CyberChef interface

Download the output and unzip it. There is a folder named AppData.

2.4. Cracking the DPAPI Password

To access the DPAPI master key, the user's DPAPI password must be found. This can be achieved using the dpapimk2john[7] command, which extracts the necessary hash for password cracking. Run Command 3 for the extraction of the necessary hash.

cd AppData/Roaming/Microsoft/Protect/S-1-5-21-3854677062-280096443-3674533662-1001

DPAPImk2john --sid="S-1-5-21-3854677062-280096443-3674533662-1001" --masterkey="8c6b6187-8eaa-48bd-be16-98212a441580" --context=local > hash.txt

Command 3

Note: If it gives the error *"Error: Please install PyCrypto package",* change the name of the package by: *sudo ln -s /usr/lib/python3/dist-packages/Cryptodome /usr/lib/python3/dist-packages/Crypto*

Crack the hash, which corresponds to the user's password, by running Command 4.

john hash.txt -wordlist="/usr/share/wordlists/rockyou.txt"

Command 4

The output will be: *bubbles*

Questions:

Q1) What is the first password that we find?

A: bubbles

2.5. Decryption of the Master Key

Carry the necessary files which are *"Login Data"* and *"Local State"* to the current directory. Find the preliminary key which is an intermediate cryptographic key by running Command 5.

pypykatz dpapi prekey password "S-1-5-21-3854677062-280096443-3674533662-1001" <user-password>

Command 5

The output will be 4 different possibilities as the preliminary key change over version of windows machine. Note these results and use for Command 6 to find the master key.

pypykatz dpapi masterkey -o key.json 8c6b6187-8eaa-48bd-be16-98212a441580 <pre-key>

Command 6

If the preliminary key is correct, the master key will be stored in the file *"key.json"*.

2.6. Login Credentials

Get the login data credentials with Command 7.

pypykatz dpapi chrome --logindata './Login Data' ./key.json './Local State'

Command 7

Find the urls with Command 8.

strings './Login Data' | grep -i http

Command 8

Questions:

Q1) What is the URL found in the first index? Fully defang the URL

A: hxxps[://]mysecuresite[.]thm/

Q2) What is the password found in the first index?

A: Sup3rPaS$w0rd1

Q3) What is the URL found in the second index? Fully defang the URL

A: hxxps[://]worksite[.]thm/

Q4) What is the password found in the second index?

A: Sup3rSecuR3!

3. References

[1] https://tryhackme.com/room/chrome
[2] https://www.kali.org/
[3] https://www.wireshark.org/
[4] https://dotnet.microsoft.com/en-us/download/dotnet-framework
[5] https://github.com/icsharpcode/ILSpy
[6] https://gchq.github.io/CyberChef/