Weaponized Disk Image Files: Analysis, Trends and Remediation

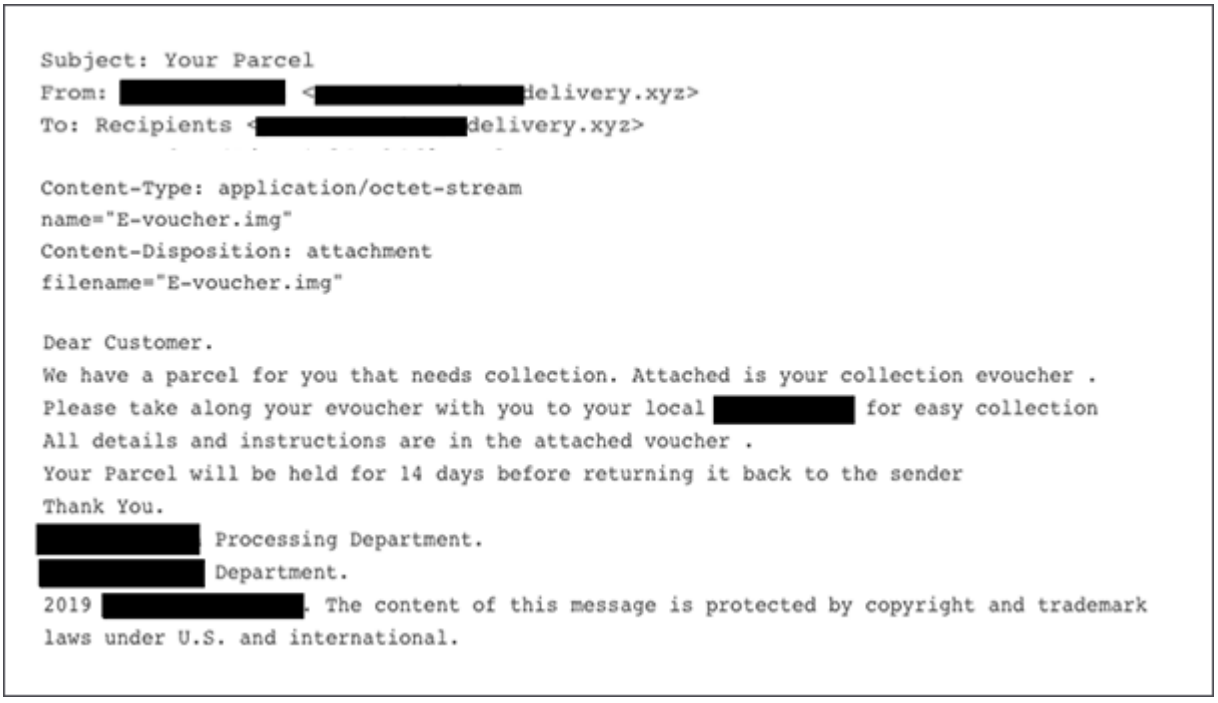
By Guillermo Taibo

Archived: 2026-04-05 19:50:10 UTC

Throughout 2019 and the beginning of 2020, the CrowdStrike® Falcon Complete™ team continuously observed a spike in the delivery of weaponized disk image files. Files such as ISO and IMG were sent to infect systems with the goal of delivering remote access trojans (RATs) as well as a few other malware variants. We've identified that these files are typically delivered via phishing campaigns as an attachment or link — a malicious URL in the body of the email or within crack software downloads. Cyber criminals have been taking advantage of built-in Windows capabilities to mount disk image files once they are opened by the end user. There are multiple disk image file formats, but we have seen ISO and IMG files being abused the most. A disk image is essentially a virtual copy of a physical disk that houses all of the files and requires that it be mounted in order to access its contents. The advantages of using disk images, combined with the easy access to purchasing RATs, make this a preferred and effective method for cybercriminals. In this blog, I dissect a campaign that uses this method to compromise a system, providing insight into what the CrowdStrike Falcon®Complete team has observed since 2019. I will also provide step-by-step remediation along with recommendations for how to implement this approach in your network.

Parcel-themed Phishing Email Scenario

The chain starts with a simple email containing a disk image file (.IMG) to socially engineer the victim into viewing the contents. The message seems to be coming from a worldwide package delivery company.



```
Subject: Your Parcel
From: [redacted] <[redacted]delivery.xyz>
To: Recipients <[redacted]delivery.xyz>

Content-Type: application/octet-stream
name="E-voucher.img"
Content-Disposition: attachment
filename="E-voucher.img"

Dear Customer.
We have a parcel for you that needs collection. Attached is your collection evoucher .
Please take along your evoucher with you to your local [redacted] for easy collection
All details and instructions are in the attached voucher .
Your Parcel will be held for 14 days before returning it back to the sender
Thank You.
[redacted] Processing Department.
[redacted] Department.
2019 [redacted]. The content of this message is protected by copyright and trademark
laws under U.S. and international.
```

Figure 1. Phishing contents sample. The delivery company did not send this email.

The attachment in this sample is only 2MB, which raises a flag immediately as disk images are typically larger in size.

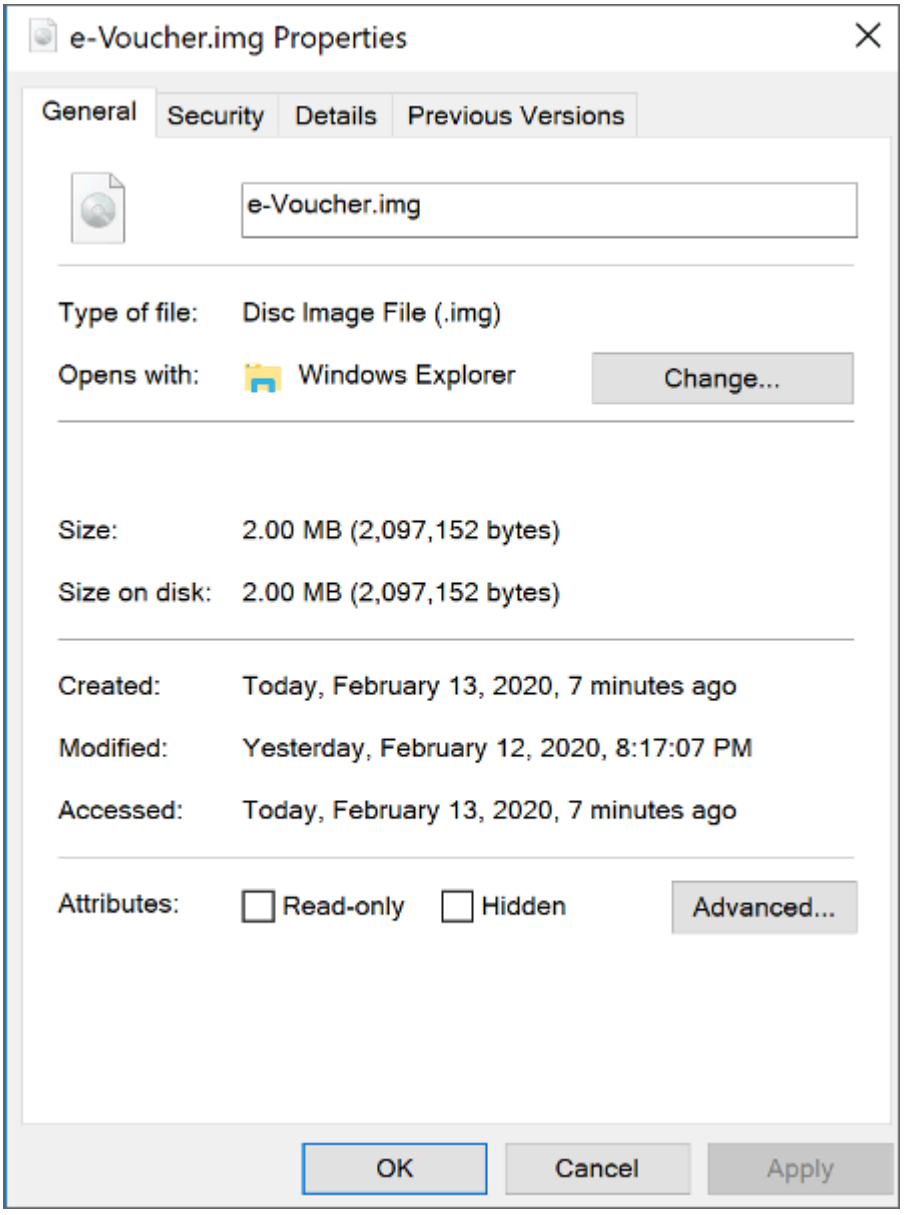


Figure 2. IMG file properties

Double-clicking on the file allows Windows 8 and Windows 10 to mount the IMG file natively to the next available drive. This sample uses a PDF icon as a disguise.

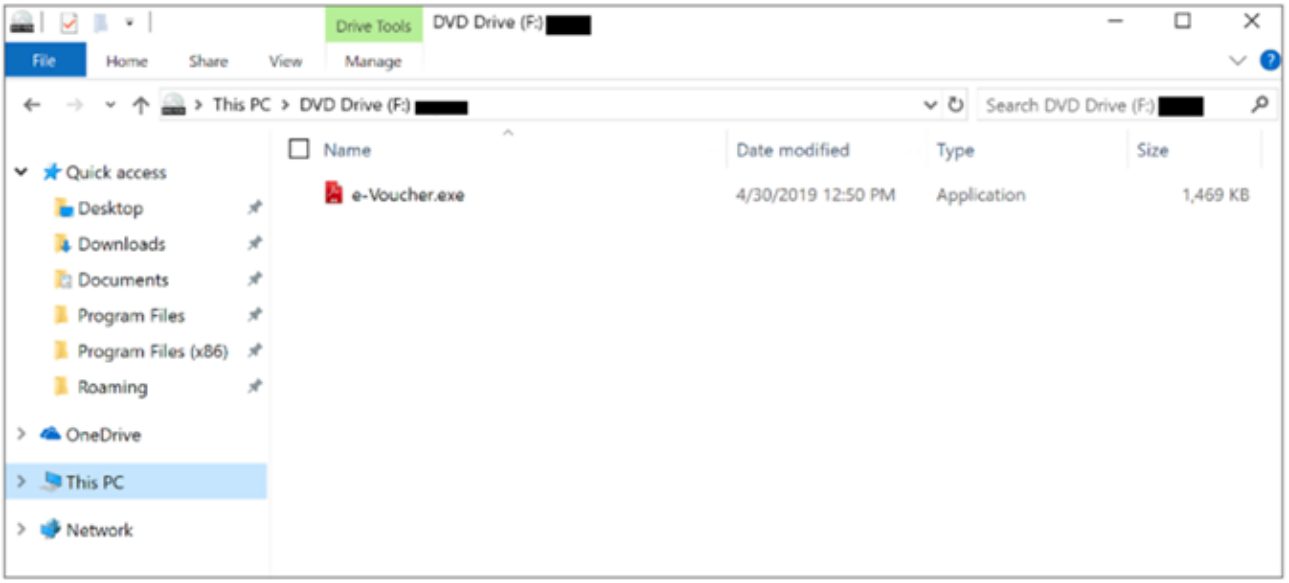


Figure 3. IMG file mounted on disk

Analysis

Exeinfo PE identified the binary as a compiled AutoIT script version 3. AutoIT is a scripting language used to automate Windows GUI tasks. Cybercriminals would first compile these scripts into an executable using the Aut2Exe compiler and further convert it into a disk image file to then distribute it widely in campaigns.

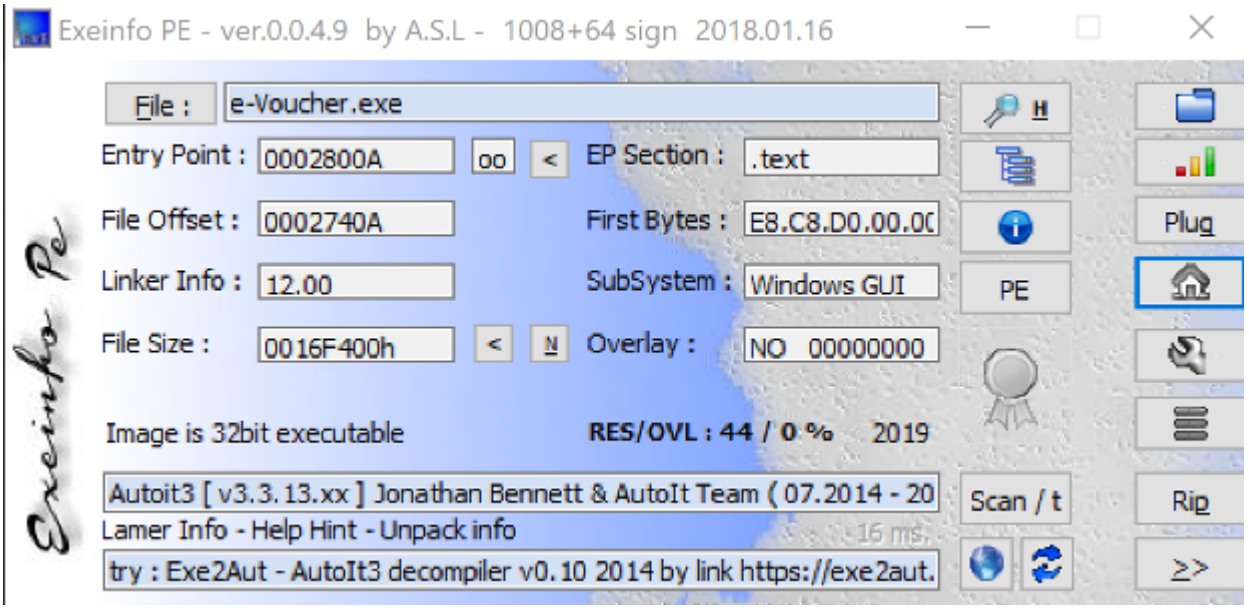


Figure 4. Exeinfo PE against binary `e-voucher.exe`

Dumping the rcdata resource and reviewing the strings shows AU3!, a common string seen in AutoIT-developed scripts.

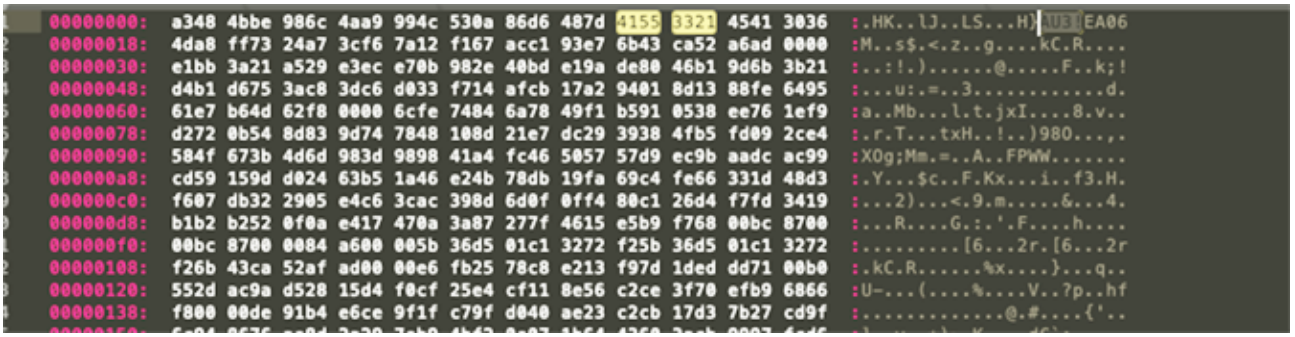


Figure 5. Hexdump of `e-voucher.exe`

The AutoIT script is obfuscated, and it is used as a dropper to eventually load the NanoCore RAT on the intended system.

```
9722        For $i = "1" To $split["0"]
9723            Global $1747249050 = 2027215461
9724            Global $te8lpwcrhn = 3379056
9725            For $e = 0 To 1560902
9726                Ptr(3224935 - 245547 * 2742726 + 3724992)
9727                If $1747249050 = 1232751179 Then
9728                    $xor = BitXOR($char, $len)
9729                    ExitLoop
9730                EndIf
9731                If $1747249050 = 2027215461 Then
9732                    $char = Asc($split[$i])
9733                    $1747249050 = 1232751179
9734                    IsBinary("McC7UeHGFxDVSgQKBjMnjOiWLaxIJsBUUUlLuTZ8ojIGRoGsWE")
9735                EndIf
9736            Next
9737            For $ii = "0" To $len - "1"
9738                $xor = BitXOR($xor, $len + $ii)
9739            Next
9740            $result &= ChrW($xor)
9741        Next
9742        Return $result
9743    EndFunc
9744
9745    fbywmpinbsfprdblltgdzxowl("PnPUnattend")
9746    Local $edsvgrzlrlfz = DllStructGetData(gcysqxsctorrzquq("dusmtask1", "8"), 1)
9747    $edsvgrzlrlfz &= DllStructGetData(gcysqxsctorrzquq("bdechangepin2", "8"), 1)
9748    $edsvgrzlrlfz &= DllStructGetData(gcysqxsctorrzquq("aadWamExtension3", "8"), 1)
9749    $edsvgrzlrlfz = swydxtrwncfvpukruyyjvmtphe($edsvgrzlrlfz, "hwnglongpcoiftynieblwrqseblfkkwvfvbhnizgvvfanyqbrn")
9750    ocxigeqqkiqejflnuwokj(True)
9751    startup("DataExchangeHost.exe", "AppVEntSubsystems64", "+", "")
```

Figure 6. Snippet of obfuscated AutoIt script

Beginning on line 9746 in Figure 6, we can see the following three resources: `dusmtask1 bdechangepin2 aadWamExtension3` The script merges these three resources and passes the key "hwnglongpcoiftynieblwrqseblfkkwvfvbhnizgvvfanyqbrn" as the second parameter to the function swydxtrwncfvpukruyyjvmtphe(). To decrypt, it creates a hash using CryptCreateHash with this key. Consequently, it then uses the function CryptDeriveKey and creates a separate key from the results of CryptCreateHash. Finally, CryptDecrypt is used to decrypt the resource.

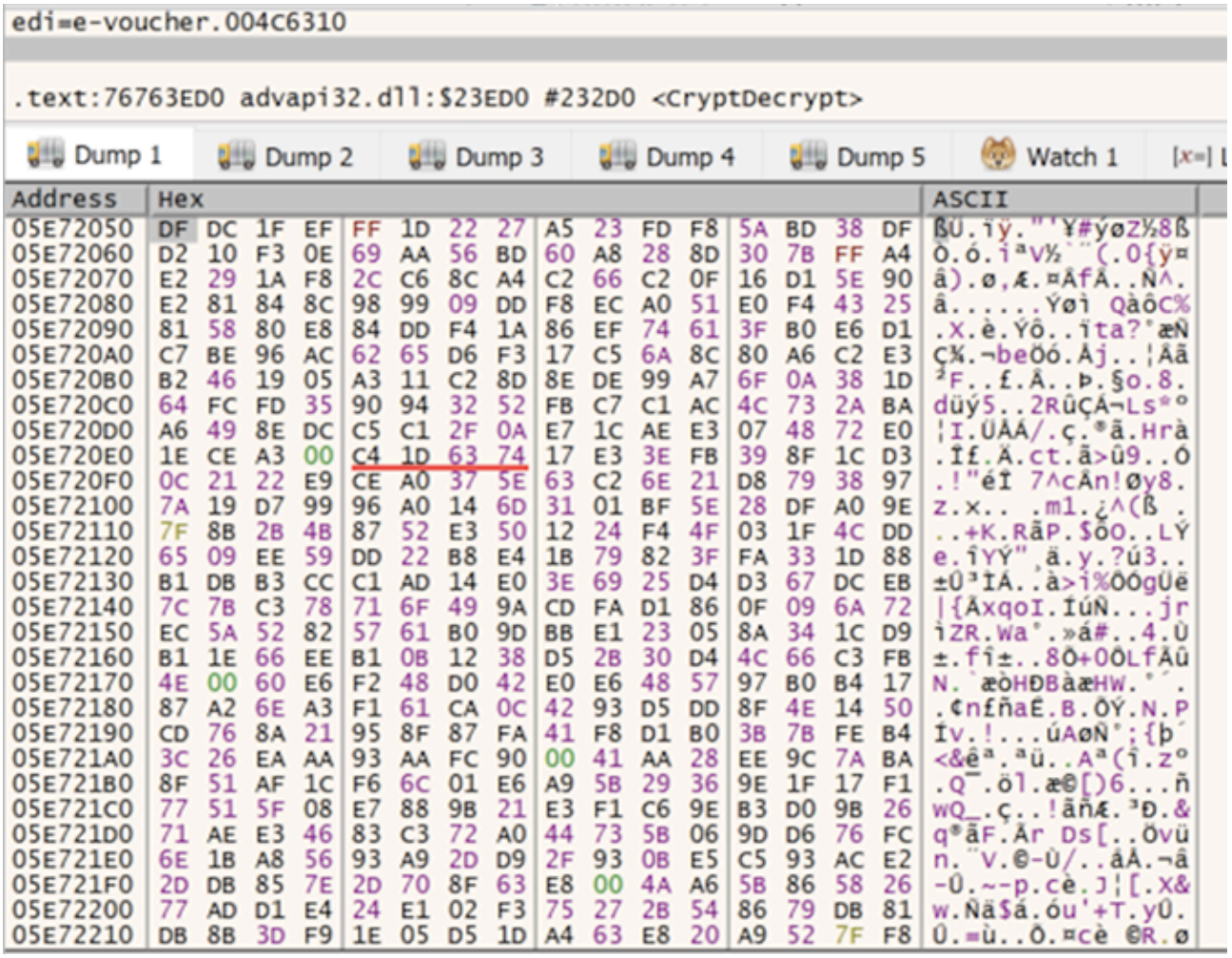


Figure 7. Encrypted stream prior to CryptDecrypt

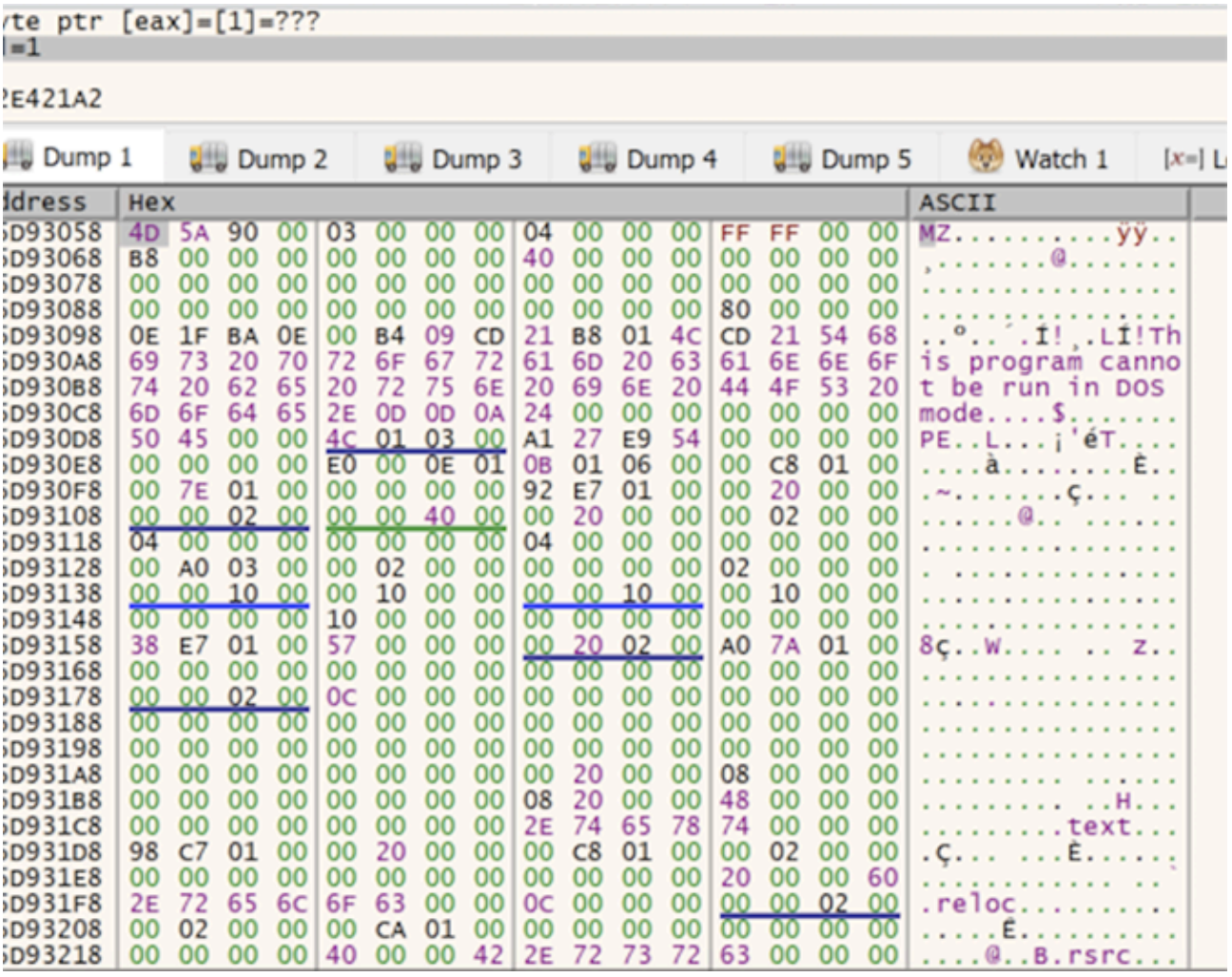


Figure 8. Contents decrypted after CryptDecrypt returns

Once the contents are decrypted, it will then use the CreateProcessW function to spawn the legitimate process RegAsm.exe in a suspended state using the process creation flag `0x00000004` (`CREATE_SUSPENDED`)

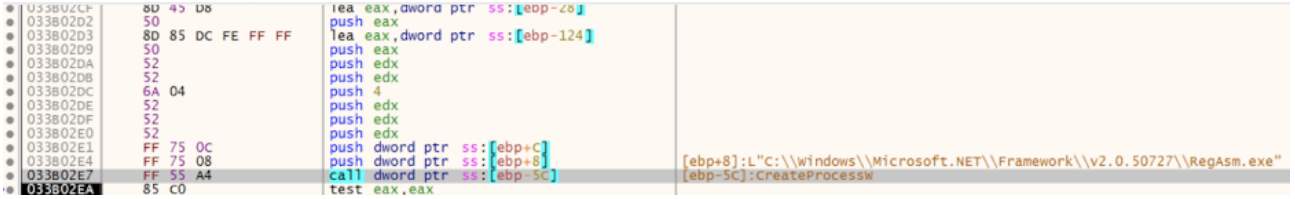


Figure 9. x32dbg debugger CreateProcessW function starts RegAsm.exe in suspended state

Shortly after, it proceeds to allocate memory space for the malicious payload that was decrypted earlier. This memory region is created with memory protection of `0x40` (`PAGE_EXECUTE_READWRITE`)

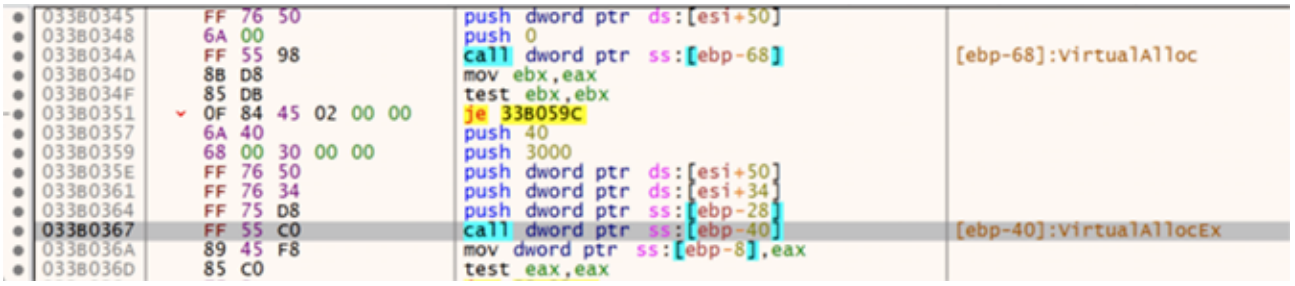


Figure 10. x32dbg debugger VirtualAllocEx allocating memory space

Last, the WriteProcessMemory call is seen to finally write the contents into this newly created memory region.

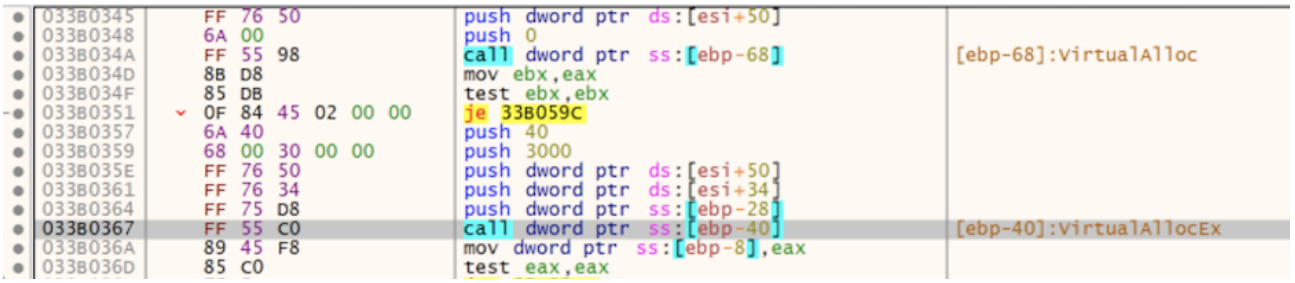


Figure 11. x32dbg debugger WriteProcessMemory function writing into memory region

Inspecting `RegAsm.exe` using ProcessHacker shows the memory region `0x400000` that was created earlier filled with the payload. The sample is using a well-known technique to hollow out `RegAsm.exe` and inject its payload.

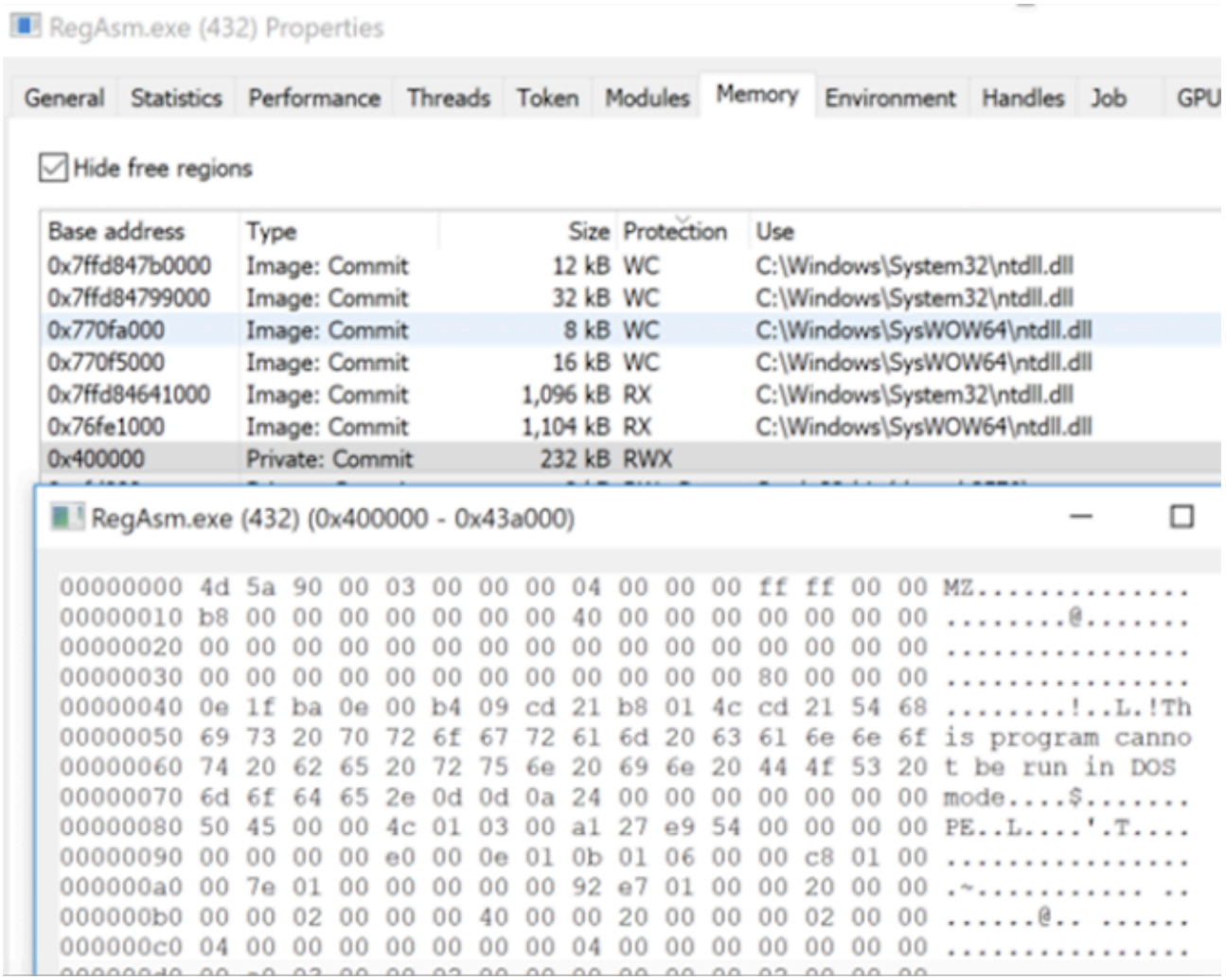


Figure 12. ProcessHacker showing memory region injected with malicious code

After dumping the malicious code out of memory, we can confirm that it is a .NET built binary packed with Eazfuscator.

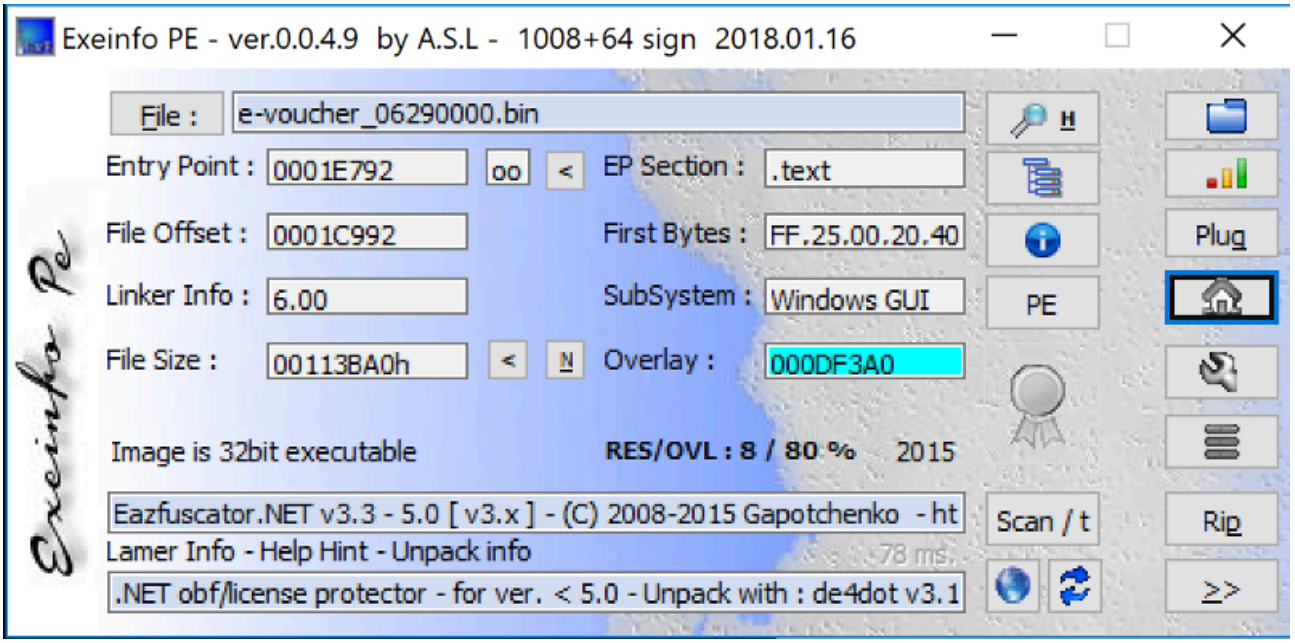


Figure 13. Exeinfo displaying packer information on dumped process

Running de4dot against this copy is able to deobfuscate to see readable strings.

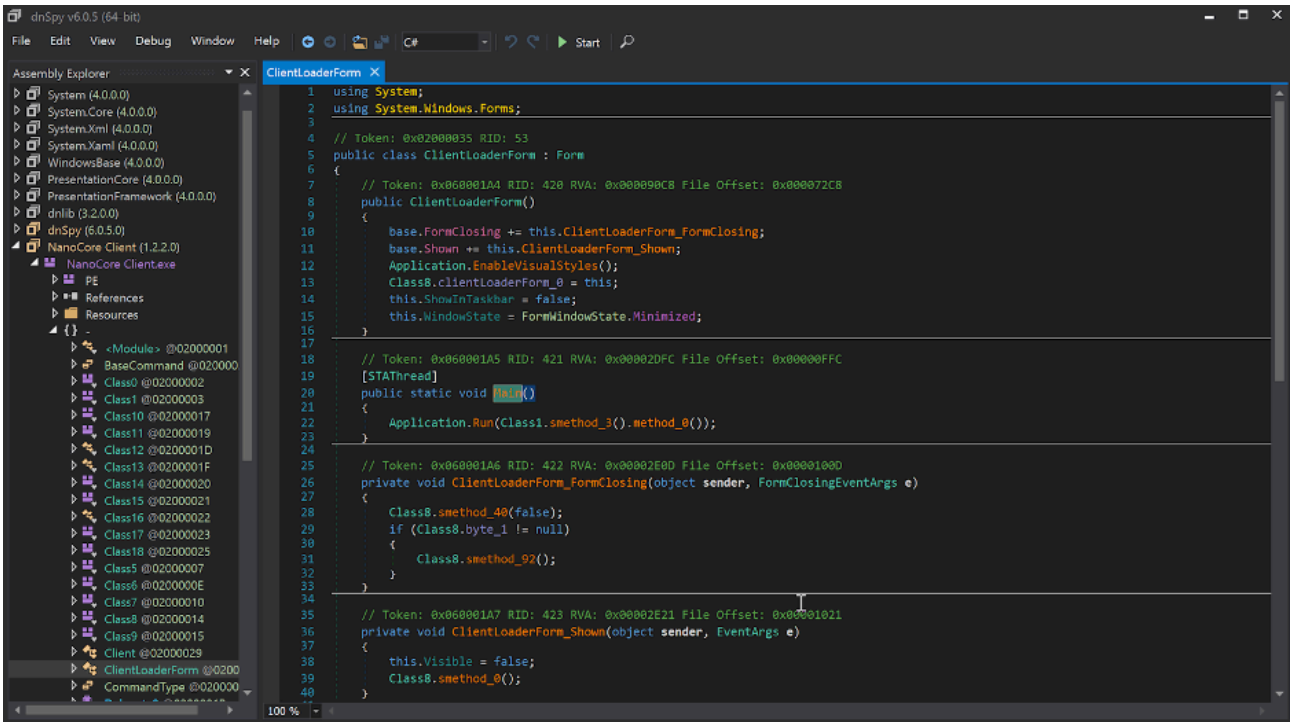


Figure 14. DnsSpy after deobfuscation

The malware then proceeds to drop a copy of itself to the path `C:\Users\username\PasswordOnWakeSettingFlyout\DataExchangeHost.exe` In addition, it creates persistence by using a URL shortcut in the StartUp folder that points to the copy of NanoCore RAT to survive reboot. A malicious VBS script named `AppVEntSubsystems64.vbs` is also dropped in the same directory where `DataExchangeHost.exe` resides.

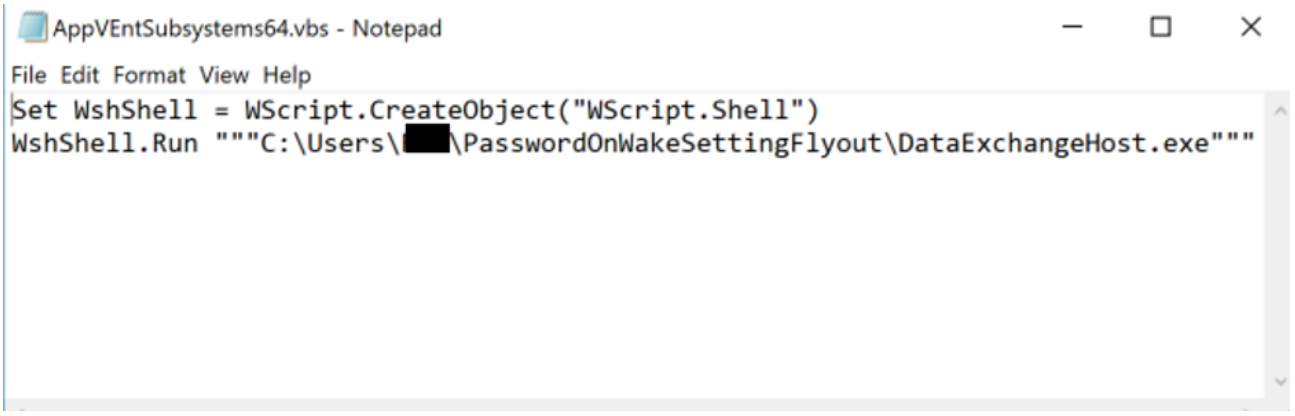


Figure 15. VBS script contents

The Falcon Complete Team has seen variations of the script above being obfuscated with the same ultimate goal such as in Figure 16.

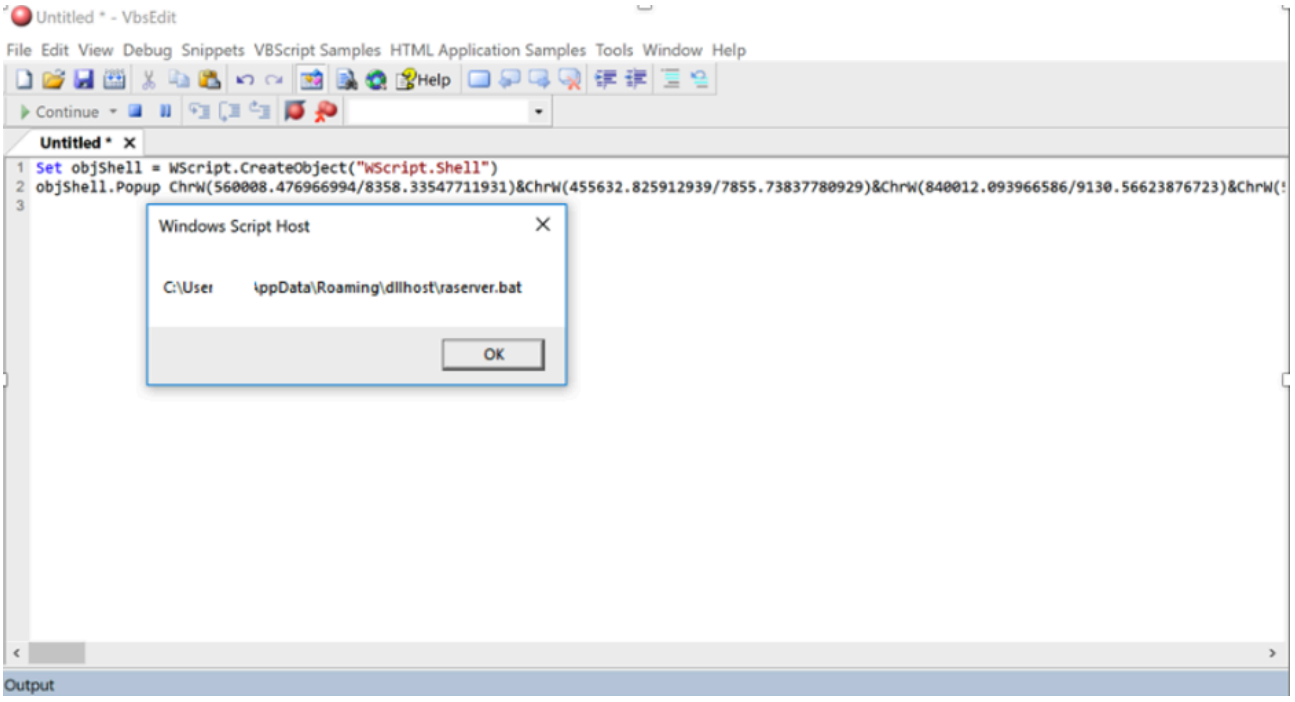


Figure 16. VbsEdit debugging obfuscated script

A copy of `RegAsm.exe` is dropped onto disk and is added to the Run key to boot on user logon, as seen in Falcon's Process Tree viewer. Falcon also logs the network connection used as the C2 in this sample, as seen in Figure 17.

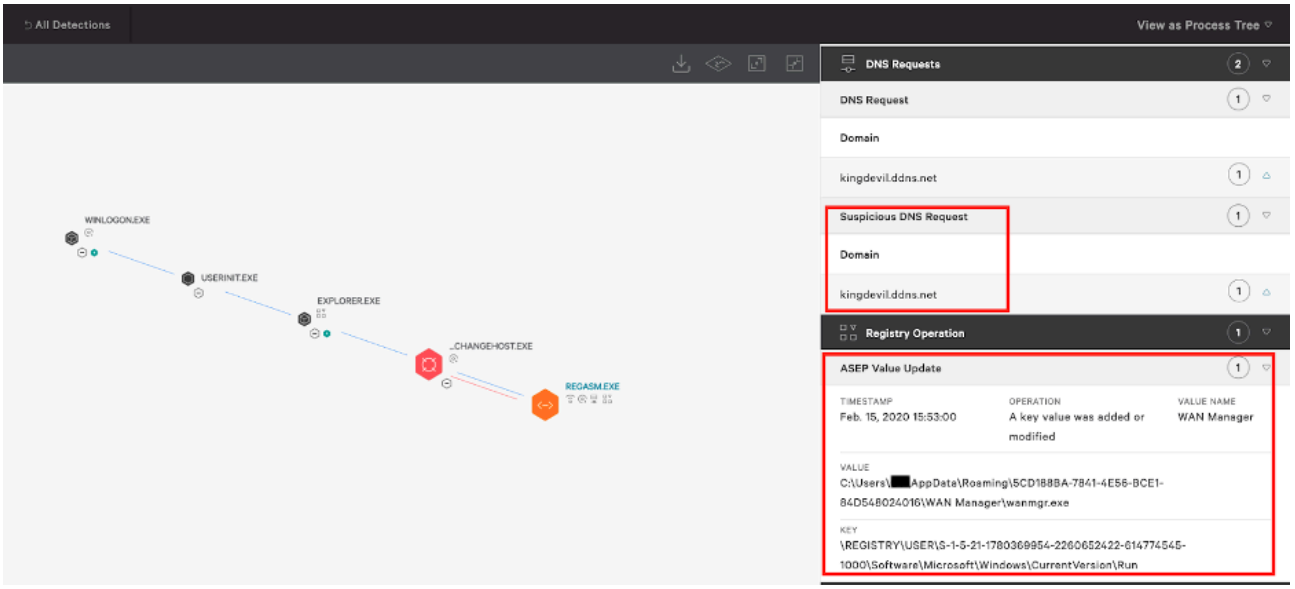


Figure 17. Falcon Process Tree displaying Registry Operations and DNS request

The functionality of NanoCore RAT has been covered heavily, so this blog will not focus on it. Figure 18 shows the same detection in Falcon's UI but this time being prevented after running the same sample with the detection and prevention settings set to "Aggressive."

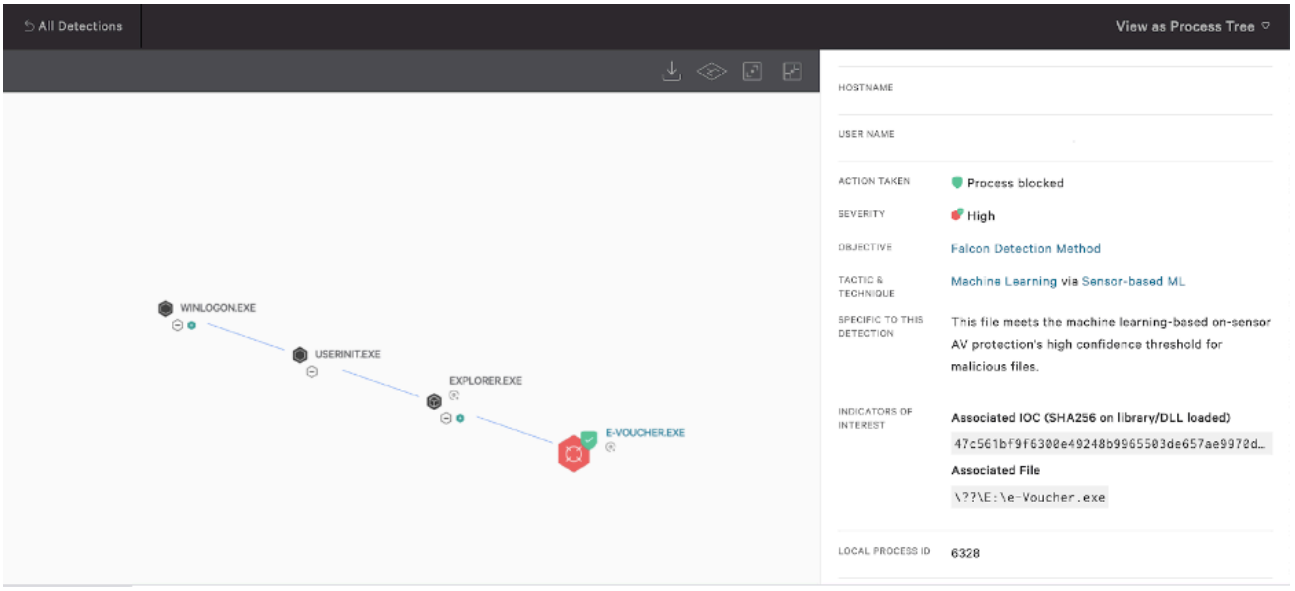


Figure 18. Prevention policy enabled

Remediation:

Remediation Difficulty

The remediation can be summarized in the following steps:

1. Identify and confirm detection originates from a virtual mounted drive:
    - Find the location of the disk image where it resides
    - Unmount the virtual drive
    - Remove the IMG from disk
2. Terminate the injected process
3. Remove the registry entry
4. Remove related directories and files

STEP 1: Identify and Remove the Mounted Disk Image

In order to identify, confirm and remove the IMG file that was mounted, we first use the class Win32_CDROMDrive from WMI in Figure 19 to provide us with information on what is currently mounted, along with the drive letter and the volume name.

```
C:\> runscript -Raw=```powershell gwmi -class win32_cdromdrive```

Caption                      Drive Manufacturer              VolumeName
-------                      ----- ------------              ----------
Microsoft Virtual DVD-ROM E:    (Standard CD-ROM drives) usps
NECVMWar VMware SATA CD01 D:    (Standard CD-ROM drives)
```

Figure 19. Output of WMI command

Now that we've identified what's mounted, we are using the PowerShell `Get-DiskImage` cmdlet to get the objects associated with the IMG file which will indicate where this file resides on disk.

```
C:\> runscript -Raw=```powershell get-diskimage -devicePath \\.\cdrom1```


Attached           : True
BlockSize          : 0
DevicePath         : \\.\CDROM1
FileSize           : 2097152
ImagePath          : C:\Users\[redacted]\Downloads\e-Voucher.img
LogicalSectorSize  : 2048
Number             : 1
Size               : 2097152
StorageType        : 1
PSComputerName     :
```

Figure 20. Output of Powershell `Get-DiskImage` command

Use the image path obtained from the output received on the previous command to unmount this virtual disk. If the process is actively running, terminate it first. Also, you first need to unmount this disk or else you will not be able to remove it.

```
C:\> runscript -Raw=```powershell Dismount-DiskImage -ImagePath C:\Users\[redacted]\Downloads\e-Voucher.img```
```

Figure 21. Unmounting IMG file using Dismount-DiskImage

STEP 2: Terminate the Injected Process

From Falcon's Process Tree, we discovered the injected RegAsm.exe process was running under the process ID 4952. Proceed to terminate this process using the built-in "kill" command using the process ID discovered.

```
C:\> kill 4952

  Id Name   Start Time (UTC-5)    PagedMemorySize      CPU HandleCount Path
  -- ----   ------------------    ---------------      --- ----------- ----
4952 RegAsm 2/15/2020 10:56:38 AM        28987392 1.640625         456 C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe


Killed PID 4952
```

Figure 22. Terminated process output

STEP 3: Remove the Registry Entry

Next, we remove the registry entry that was created at infection by using the PowerShell command in Figure 23.

```
C:\> reg delete 'HKEY_USERS\S-1-5-21-1780369954-2260652422-614774545-1000\software\microsoft\windows\currentversion\run' "WAN Manager"
Deleted (HKEY_USERS\S-1-5-21-1780369954-2260652422-614774545-1000\software\microsoft\windows\currentversion\run.WAN Manager)
```

Figure 23. Deleting registry entry successfully

STEP 4: Remove Related Directories and Files

Last, we remove all remaining directories and files that were discovered during timeline analysis of the system.

```
C:\> rm 'C:\users'[redacted]'\PasswordOnWakeSettingFlyout' -force
Deleted 'C:\users\[redacted]\PasswordOnWakeSettingFlyout'
```

Figure 24. Removing artifacts from disk output

```
C:\users\[redacted]\appdata\roaming> rm '5CD188BA-7841-4E56-BCE1-84D548024016' -force
Deleted 'C:\users\[redacted]\appdata\roaming\5CD188BA-7841-4E56-BCE1-84D548024016'
```

Figure 25. Removing artifacts from disk output

```
C:\> rm 'C:\users\[redacted]\appdata\Roaming\Microsoft\Windows\Start Menu\Programs\startup\AppVEntSubsystems64.url'
Deleted 'C:\users\[redacted]\appdata\Roaming\Microsoft\Windows\Start Menu\Programs\startup\AppVEntSubsystems64.url'
```

Figure 26. Removing artifacts from disk output

This completes the remediation steps we execute to tackle such variants when discovered. Note that in this scenario, we've purposely turned off the prevention policy while leaving the detection policy turned on for illustrative purposes. Within the scope of our service, we've been able to observe Warzone, NanoCore and Agent Tesla RATs to be the most preferred by cybercriminals among others as seen in Figure 27.

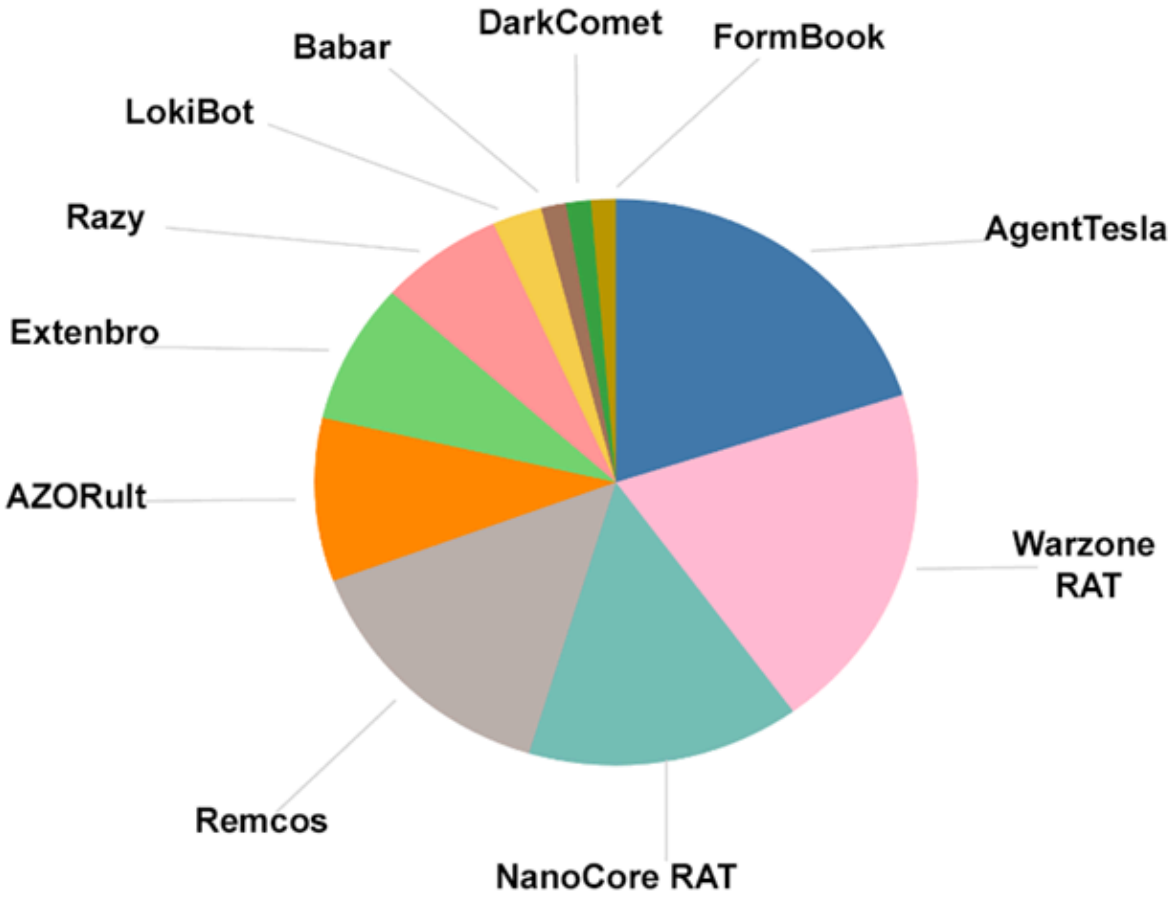


Figure 27. Malware family breakdown

The entry vector for these have primarily been phishing emails, where users download Torrent/Crack software onto their machines disguised as movies, games or music but that actually contains infected USB media.

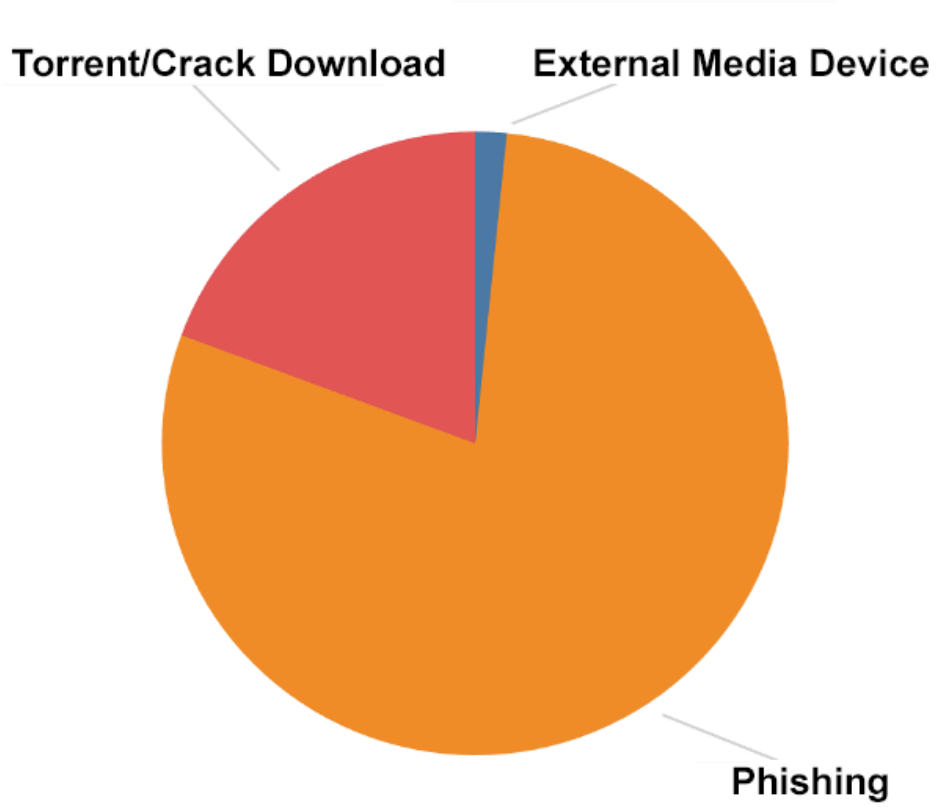


Figure 28. Entry vector breakdown

In regard to verticals, we've noticed these campaigns are widely spread across multiple verticals, with the hospitality sector being the most affected.

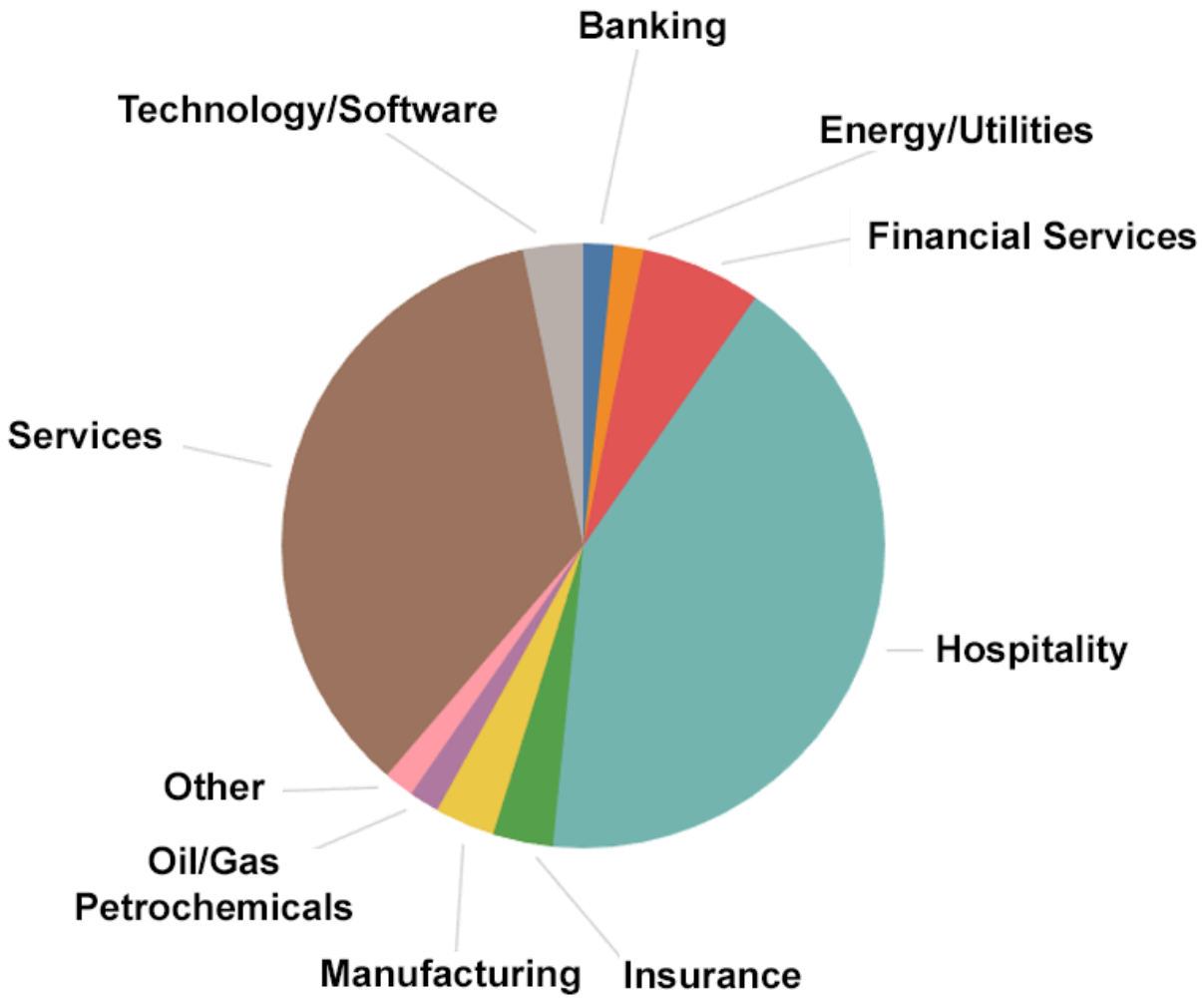


Figure 29. Affected verticals observed

Recommendations

1. Gain advanced visibility across your endpoints with anendpoint detection and response (EDR)solution such as theCrowdStrikeFalcon® platform.Turn onnext-gen antivirus (NGAV)preventative measures to stop malware.
2. Leverage a Layer 7 firewall that can perform deep packet inspection to examine the traffic and block P2P protocol types.
3. Observe inbound emails received during a short span of time to see the volume of disk image files being delivered as attachments. If applicable, block known disk images file types such as IMG, ISO, DAA, VHD, CDI, VMDK, etc., to reduce the attack surface.
4. Leverage a proxy to proactively block sites that are uncategorized/unknown, as we've seen new sites registered shortly before phishing campaigns are executed.
5. Incorporate a phishing awareness program internally, and routinely test employees with phishing test emails.

We've seen a shift toward cybercriminals using AutoIt and disk images to further achieve their objectives through various mass phishing campaigns. We believe this shift is primarily to evade detection from legacy AV software and bypass the email gateway, as most are not inspecting or blocking these file types, and no software is required to mount these disk images as Windows is able to natively mount them. We predict that in 2020, we will continue to see this trend as RATs become increasingly accessible to cybercriminals.

Additional Resources

- *Learn more about the CrowdStrike Falcon® platform by visiting the webpage.*
- *Learn how you can raise your organization's cybersecurity maturity to the highest level immediately with CrowdStrike Falcon® Complete™.*
- *Learn how you can take advantage of automated malware analysis and sandbox by visiting the CrowdStrike Falcon SandboxTM webpage.*
- *Learn how CrowdStrike combines automated analysis with human intelligence to enable security teams to get ahead of the attacker's next move by visiting the CROWDSTRIKE FALCON® INTELLIGENCETM webpage.*
- *Get a full-featured free trial of CrowdStrike Falcon® Prevent™ and learn how true next-gen AV performs against today's most sophisticated threats.*
