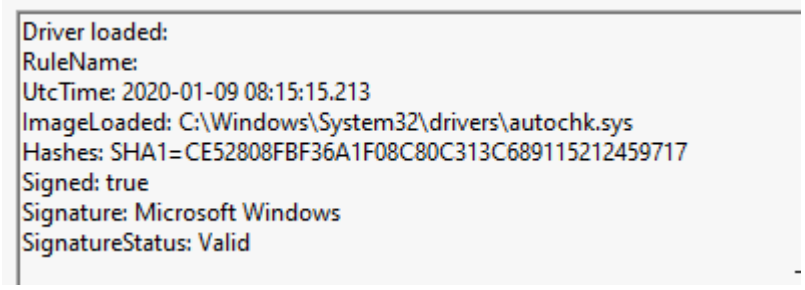
APT27 ZXShell RootKit module updates

Published: 2020-01-13 · Archived: 2026-04-05 20:11:12 UTC

Within the toolset of the different APT groups, one of the most interesting elements and the one that generally worries the most, are their capabilities in Ring0, generally RootKit/Bootkit type threats that act with the maximum level of privileges.

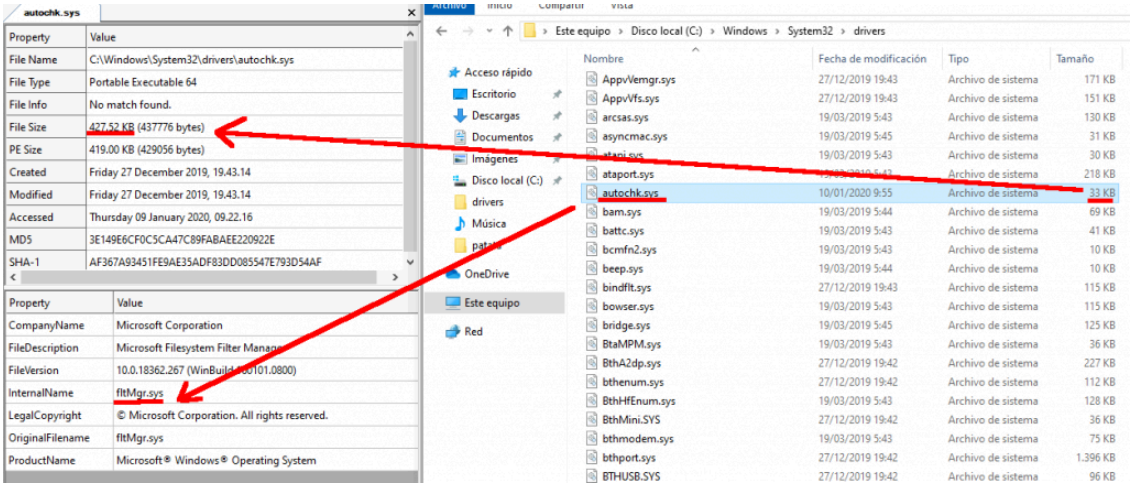
An example of this type of threats is the RootKit module of ZxShell RAT used by Emissary Panda (APT27), of which there is a relatively recent sample (Uploaded to Virustotal since 2019-09-21 17:59:39) that is also correctly signed, so it can be loaded in the latest version of Windows 10 and is perfectly functional as far as we have been able to check.



Sysmon DriverLoaded event

A complete analysis of this threat can be found made by the analyst Ori Damari (@0xrepnz) in his blog (https://repnz.github.io/posts/autochk-rootkit-analysis/). After analyzing this threat and describing its capabilities, he has rewritten the source code from a sample of this threat uploaded in 2018 to Virustotal, and published it in GitHub, which greatly facilitates the analysis of newer versions. As he describes in his blog, the capabilities of this Rootkit are basically the following:

- **File Redirection** – Redirect malicious files to benign files. If you try to call CreateFile() to open a malicious file you'll get a handle to a benign file.



- **Network Connection Hiding** – Hide network connections from tools like netstat,proceshacker…

We found interesting to analyze the differences between the 2018 version and the most recent 2019 version in order to try to identify new capabilities or changes in its capabilities. After comparing both samples using the GitHub source code, we have been able to see that most of the functions are identical, except for 5 of them (including the Driver's entrypoint):

Line	Address	Name	Address 2	Name 2	Ratio	BBlocks 1	BBlocks 2	Description
00000	000114b4	NetHookTcpDriver	00011560	NetHookTcpDriver	1.000	6	6	Perfect match, same name
00001	000116b4	NetTcpDriverCompletionRoutine	00011760	NetTcpDriverCompletionRoutine	1.000	41	41	Perfect match, same name
00002	00011d1c	FsCreateFileHook	00011bc4	FsCreateFileHook	1.000	16	16	Perfect match, same name
00003	0001285c	FsGetRedirectionTarget	00012128	FsGetRedirectionTarget	1.000	11	11	Perfect match, same name
00004	00012998	FsAddFileRedirection	00012264	FsAddFileRedirection	1.000	20	20	Perfect match, same name
00005	00011b4c	NetHookNsiProxy	000119f4	NethookNsiProxy	1.000	4	4	Same cleaned up assembly or pseudo-code
00006	00011600	NetTcoDriverDeviceIoctlHook	000116ac	NetTcpDriverDeviceIoctlHook	1.000	9	9	Same cleaned up assembly or pseudo-code
00007	000128ec	FsAddIgnoredTarget	000121b8	FsAddIgnoredProcess	1.000	11	11	Same cleaned up assembly or pseudo-code
00009	00011178	AutocheckDEviceControl	000112e8	AutochkDeviceControl	1.000	20	20	Same rare KOKA hash
00010	00011158	AutoChkIrpDefaultDispatcher	000112c8	AutochkIrpDefaultDispatcher	1.000	1	1	Mnemonics and names

Identical functions in both versions

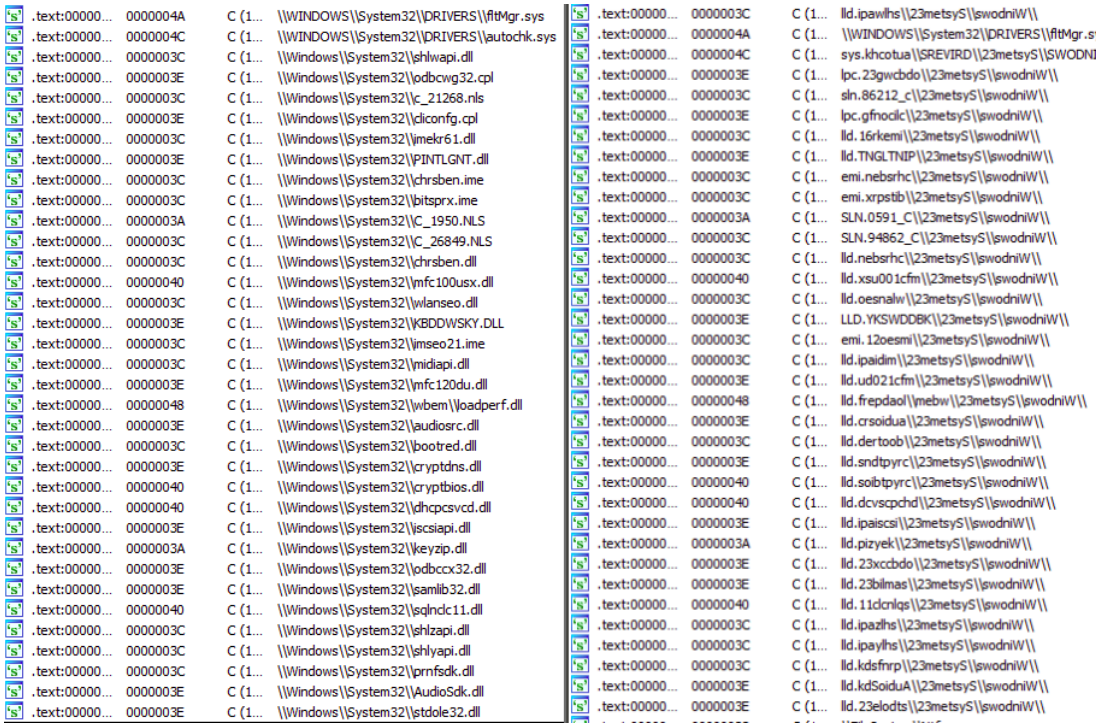
Line	Address	Name	Address 2	Name 2	Ratio	BBlocks 1	BBlocks 2	Description
00000	00011008	FsFreeFileRedirection	00011168	FsFreeFileRedirection	0.880	16	17	Perfect match, same name
00002	00011948	NetInitializeConnectionHider	00011008	NetInitializeConnectionHider	0.640	8	8	Perfect match, same name
00001	000112ac	DriverEntry	0001141c	DriverEntry	0.620	8	6	Perfect match, same name
00003	00011ee4	FsPutRedirectorHook	00011d8c	FsPutRedirectorHook	0.590	23	21	Perfect match, same name
00004	000120f0	FsInitializeFileRedirection	00011ecc	FsInitializeFileRedirection	0.130	71	4	Perfect match, same name

Different functions

After analyzing the differences between this 5 funtcions, we have been able to observe that all the changes are focused on avoiding detections by slightly "obfuscating" some IOCs hardcoded as strings and code modification without impact in the capabilities on the driver…

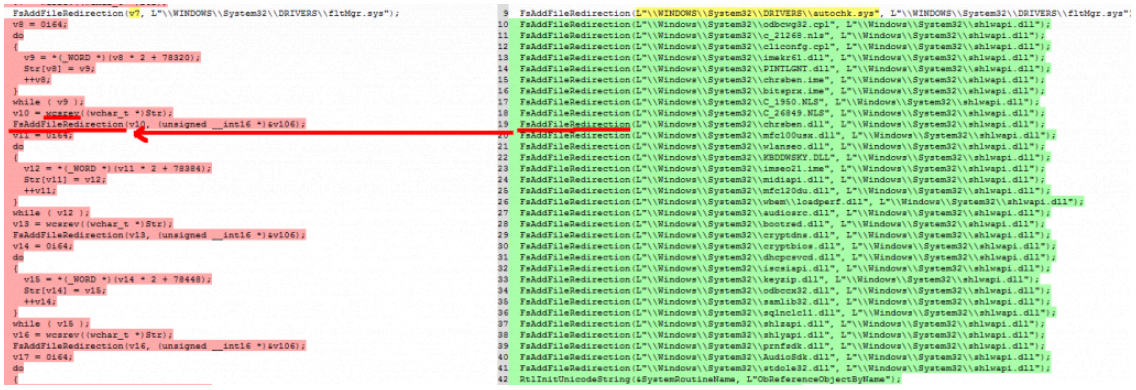
In total, there are three notable changes between the two versions:

- The first one basically consists in that they have reversed the list of strings that identify the files that the Driver hides by default when it is loaded:



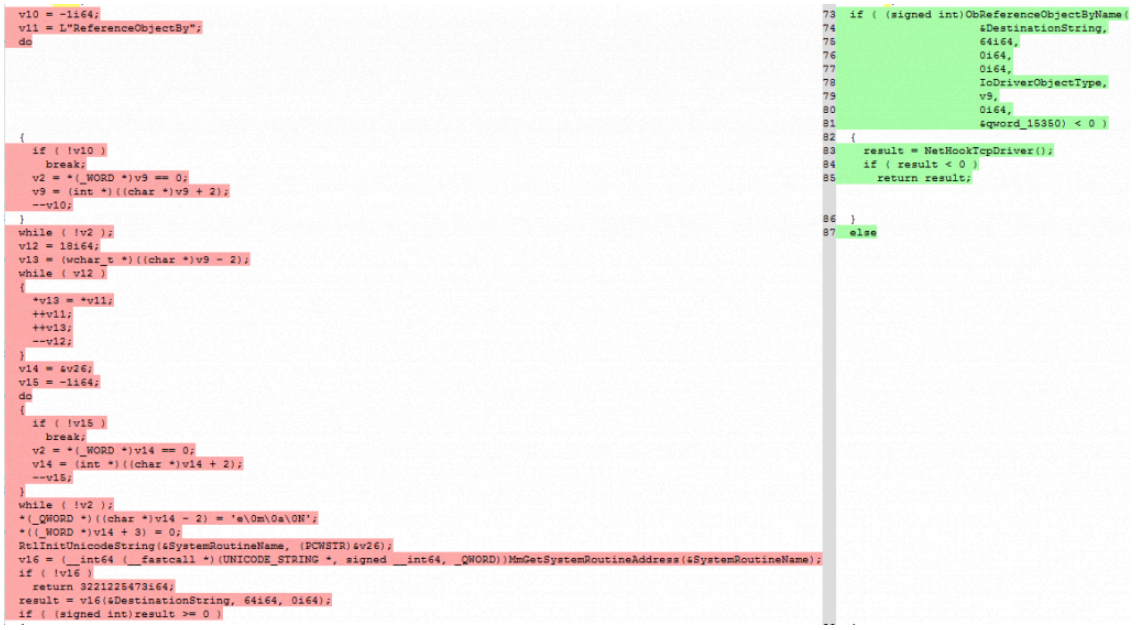
Old and New list of file names

At code level, the impact this has had is that the function that redirects these files, now uses the “wcrev” function that flips the strings before passing them to the function that hides the files:



New code (Red) and old code (Green)

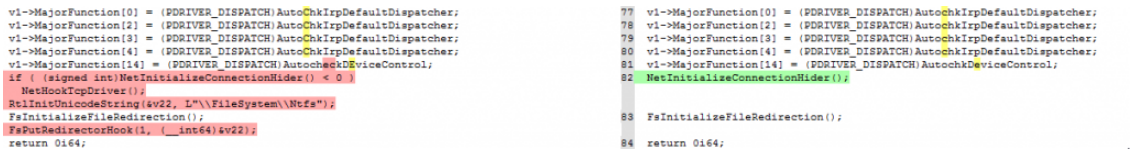
- Secondly, they have tried to disguise their use of the undocumented Microsoft API “ObReferenceObjectByName”, which is used to get the pointer to the different Driver_Object drivers they intend to hook in each case. Until now, they had the name of this function in their strings, and used it to resolve it by passing its name to the MmGetSystemRoutineAddress API which returns a pointer to it. Now they only keep part of the name, and complete the rest in a slightly more complex way before calling MmGetSystemRoutineAddress by building it from characters they store in the registers and other areas of the binary:



New code (Red) and old code (Green)

- Finally, they have moved part of the logic of some functions to another point, maintaining the same functionality. An example is the end of the driver entry function, where untill now, at the end they only called two functions that initialized the logic of hiding connections and redirecting files, and now, they

have extracted part of the logic of these functions and moved it right after each one of them, but without any impact on the capabilities and behavior of the Driver:



```
v1->MajorFunction[0] = (PDRIVER_DISPATCH)AutoChkIrpDefaultDispatcher;
v1->MajorFunction[2] = (PDRIVER_DISPATCH)AutoChkIrpDefaultDispatcher;
v1->MajorFunction[3] = (PDRIVER_DISPATCH)AutoChkIrpDefaultDispatcher;
v1->MajorFunction[4] = (PDRIVER_DISPATCH)AutoChkIrpDefaultDispatcher;
v1->MajorFunction[14] = (PDRIVER_DISPATCH)AutochkDEviceControl;
if ( (signed int)NetInitializeConnectionHider() < 0 )
  NetHookTcpDriver();
RtlInitUnicodeString(&v22, L"\\FileSystem\\Ntfs");
FsInitializeFileRedirection();
FsPutRedirectorHook(1, (__int64)&v22);
return 0i64;
```

```
77 v1->MajorFunction[0] = (PDRIVER_DISPATCH)AutochkIrpDefaultDispatcher;
78 v1->MajorFunction[2] = (PDRIVER_DISPATCH)AutochkIrpDefaultDispatcher;
79 v1->MajorFunction[3] = (PDRIVER_DISPATCH)AutochkIrpDefaultDispatcher;
80 v1->MajorFunction[4] = (PDRIVER_DISPATCH)AutochkIrpDefaultDispatcher;
81 v1->MajorFunction[14] = (PDRIVER_DISPATCH)AutochkDeviceControl;
82 NetInitializeConnectionHider();


83 FsInitializeFileRedirection();

84 return 0i64;
```

New code (Red) and old code (Green)

x64 Sample	42eab05c611bf24d86bb6c985caa2ad7380ed7d98340c7f08de9361be14dc244
x86 Sample	9b7c1e37d5f56cc0b5e5e22ce9805e237a189297e78405b9c392a0953b6e0321

Reader Interactions

Source: https://lab52.io/blog/apt27-rootkit-updates/