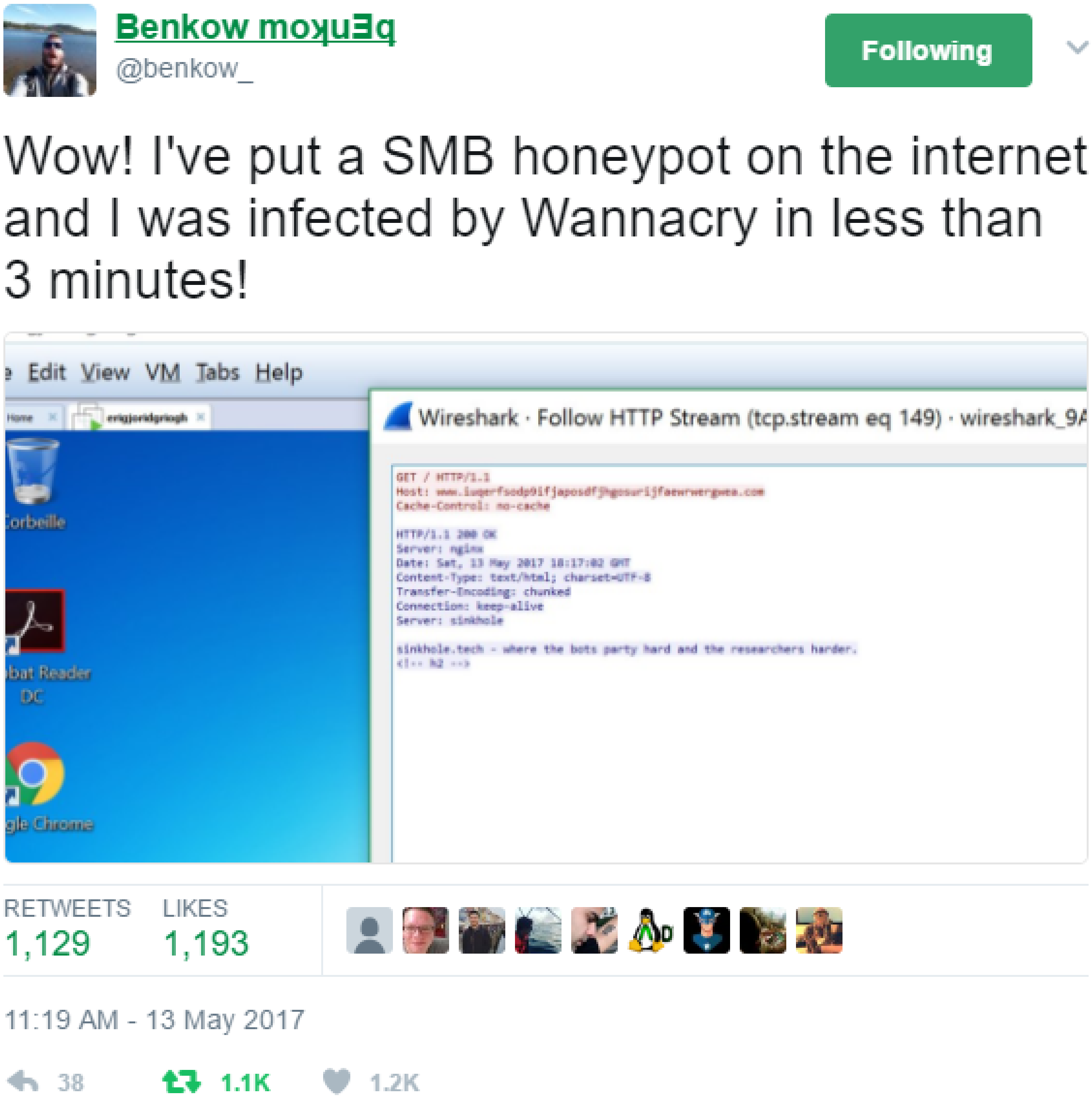
UIWIX – Evasive Ransomware Exploiting ETERNALBLUE

minerva-labs.com/post/uiwix-evasive-ransomware-exploiting-eternalblue

- Tweet
-

Last week everybody talked about the WannaCry ransomware, a non-evasive ransomware which exploited vulnerable servers to propagate, successfully infecting anything from digital billboards to the Russian interior ministry. Here at Minerva we took part in the global effort against evil, releasing a free vaccination tool, explaining how you may vaccinate in enterprise-scale.

WannaCry drew attention to other threats exploiting the very same SMB vulnerability (MS-17-010) using the Shadow Brokers' ETERNALBLUE-DOUBLEPULSAR combination. Unlike WannaCry, there have been no reports on the number of machines infected by the UIWIX ransomware, neither about the "revenues" generated. We assume that it is a direct result of a single major difference between WannaCry and the UIWIX ransomware family used in these threats. WannaCry did not try to evade detection and some researchers reported that their honeypots were infected only three minutes after they were deployed.



Benkow moʞuƎq
@benkow_

Wow! I've put a SMB honeypot on the internet and I was infected by Wannacry in less than 3 minutes!

Wireshark · Follow HTTP Stream (tcp.stream eq 149) · wireshark_9A

```
GET / HTTP/1.1
Host: www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com
Cache-Control: no-cache

HTTP/1.1 200 OK
Server: nginx
Date: Sat, 13 May 2017 10:17:02 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Server: sinkhole

sinkhole.tech - where the bots party hard and the researchers harder.
<!-- h2 -->
```

RETWEETS 1,129 LIKES 1,193

11:19 AM - 13 May 2017

38 1.1K 1.2K

Tweet about honeypots infected within 3 minutes

UIWIX however employed basic evasion techniques to stay under the radar:

Catalin Cimpanu
@campuscodi

Follow

The ransomware exists, but nobody has seen any payload/hash. UIWIX has been active for at least 8 days now. "Abilities" still unconfirmed

Security Alert: Uiwix Ransomware Is Here and It Can Be Worse Than Wannacry

Cyber criminals are flooding the market with ransomware

RETWEETS 3 LIKES 6

2:54 PM - 14 May 2017

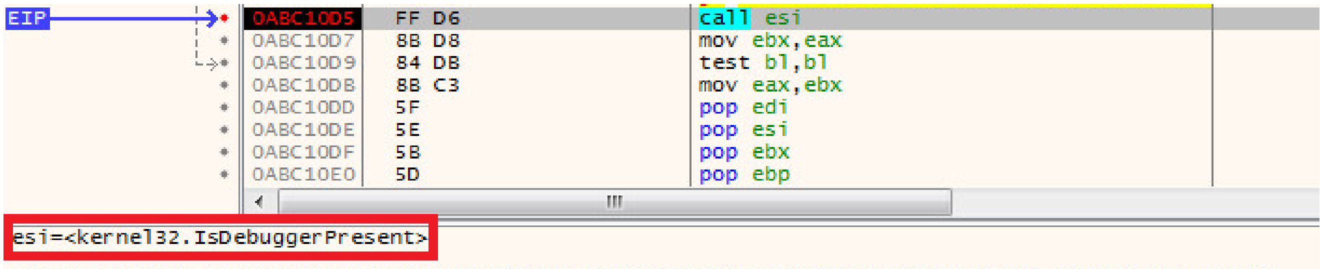
2 3 6

Tweet about the difficulty in obtaining a UIWIX sample

In this blog post, we describe how the UIWIX ransomware bypasses existing security defenses to target endpoints.

A Step-By-Step Analysis of How UIWIX Evades Detection

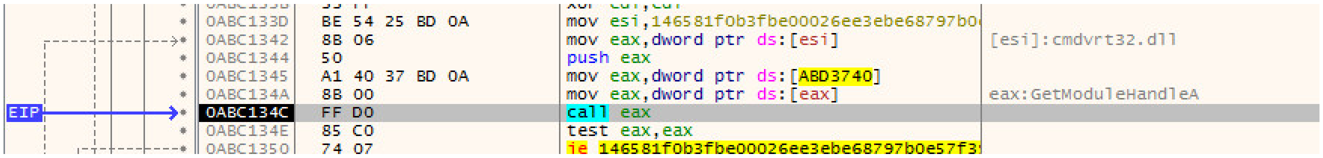
UIWIX did not invent any new technique, they relied on simple known techniques – starting with a direct test for the presence of a debugger:



```
EIP  0ABC10D5  FF D6   call esi
     0ABC10D7  8B D8   mov ebx,eax
     0ABC10D9  84 DB   test bl,bl
     0ABC10DB  8B C3   mov eax,ebx
     0ABC10DD  5F      pop edi
     0ABC10DE  5E      pop esi
     0ABC10DF  5B      pop ebx
     0ABC10E0  5D      pop ebp

esi=<kernel32.IsDebuggerPresent>
```

Elementary test for the presence of a debugger

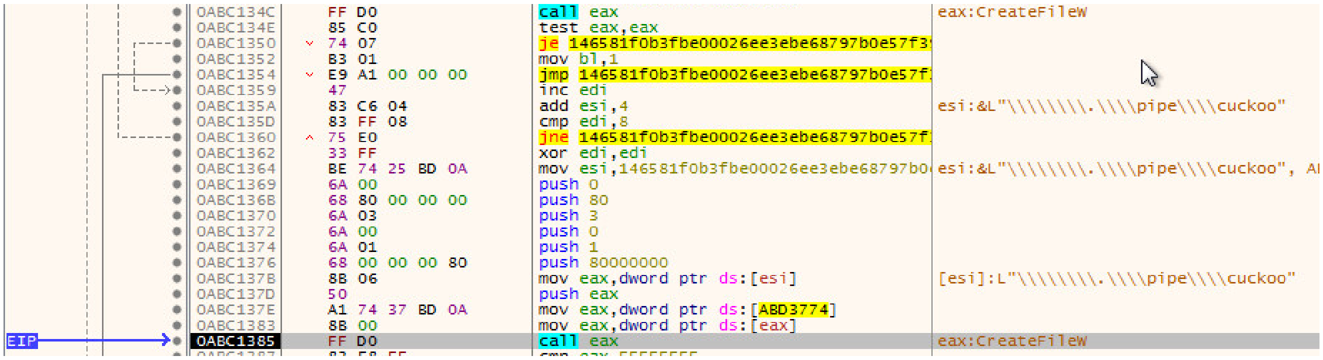
Later moving to detect different sandbox solutions, UIWIX checks the loaded modules against a black list a list of DLLs (see full list below):



```
     0ABC133D  BE 54 25 BD 0A  mov esi,146581f0b3fbe00026ee3ebe68797b0
     0ABC1342  8B 06           mov eax,dword ptr ds:[esi]               [esi]:cmdvrt32.dll
     0ABC1344  50              push eax
     0ABC1345  A1 40 37 BD 0A  mov eax,dword ptr ds:[ABD3740]
     0ABC134A  8B 00           mov eax,dword ptr ds:[eax]               eax:GetModuleHandleA
EIP  0ABC134C  FF D0           call eax
     0ABC134E  85 C0           test eax,eax
     0ABC1350  74 07           je 146581f0b3fbe00026ee3ebe68797b0e57f3
```

UIWIX tests if a DLL related to COMODO's sandbox is loaded

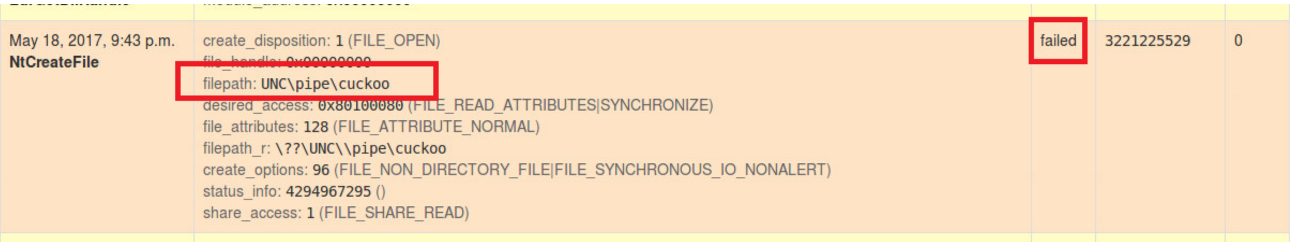
Afterwards, the ransomware tests if a Cuckoo sandbox pipe is present:



```
     0ABC134C    FF D0           call eax                                  eax:CreateFileW
     0ABC134E    85 C0           test eax,eax
     0ABC1350  v 74 07           je 146581f0b3fbe00026ee3ebe68797b0e57f3
     0ABC1352    B3 01           mov bl,1
     0ABC1354  v E9 A1 00 00 00  jmp 146581f0b3fbe00026ee3ebe68797b0e57f
     0ABC1359    47              inc edi
     0ABC135A    83 C6 04        add esi,4                                 esi:&L"\\\\\\\\.\\\\pipe\\\\cuckoo"
     0ABC135D    83 FF 08        cmp edi,8
     0ABC1360  ^ 75 E0           jne 146581f0b3fbe00026ee3ebe68797b0e57f
     0ABC1362    33 FF           xor edi,edi
     0ABC1364    BE 74 25 BD 0A  mov esi,146581f0b3fbe00026ee3ebe68797b0   esi:&L"\\\\\\\\.\\\\pipe\\\\cuckoo", A
     0ABC1369    6A 00           push 0
     0ABC136B    68 80 00 00 00  push 80
     0ABC1370    6A 03           push 3
     0ABC1372    6A 00           push 0
     0ABC1374    6A 01           push 1
     0ABC1376    68 00 00 00 80  push 80000000
     0ABC137B    8B 06           mov eax,dword ptr ds:[esi]                [esi]:L"\\\\\\\\.\\\\pipe\\\\cuckoo"
     0ABC137D    50              push eax
     0ABC137E    A1 74 37 BD 0A  mov eax,dword ptr ds:[ABD3774]
     0ABC1383    8B 00           mov eax,dword ptr ds:[eax]
EIP  0ABC1385    FF D0           call eax                                  eax:CreateFileW
```

The malware tests if the Cuckoo pipe is present

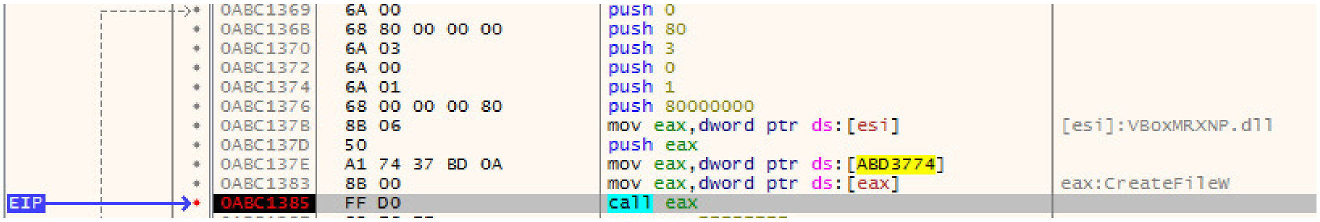
Ironically, the test for the Cuckoo pipe triggers both a signature and returns false even when executed in a Cuckoo sandbox:



May 18, 2017, 9:43 p.m. **NtCreateFile**	create_disposition: 1 (FILE_OPEN) file_handle: 0x00000000 filepath: UNC\pipe\cuckoo desired_access: 0x80100080 (FILE_READ_ATTRIBUTES|SYNCHRONIZE) file_attributes: 128 (FILE_ATTRIBUTE_NORMAL) filepath_r: \??\UNC\\pipe\cuckoo create_options: 96 (FILE_NON_DIRECTORY_FILE|FILE_SYNCHRONOUS_IO_NONALERT) status_info: 4294967295 () share_access: 1 (FILE_SHARE_READ)	failed	3221225529	0

Although executed in a Cuckoo, the test returns false

Now, UIWIX tests yet another list of DLLs, this time they are VM related:



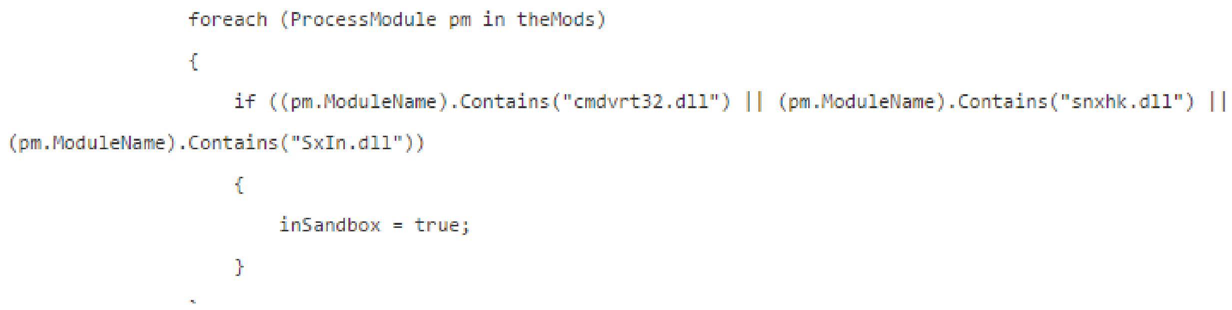
```
     0ABC1369  6A 00           push 0
     0ABC136B  68 80 00 00 00  push 80
     0ABC1370  6A 03           push 3
     0ABC1372  6A 00           push 0
     0ABC1374  6A 01           push 1
     0ABC1376  68 00 00 00 80  push 80000000
     0ABC137B  8B 06           mov eax,dword ptr ds:[esi]       [esi]:VBoxMRXNP.dll
     0ABC137D  50              push eax
     0ABC137E  A1 74 37 BD 0A  mov eax,dword ptr ds:[ABD3774]
     0ABC1383  8B 00           mov eax,dword ptr ds:[eax]       eax:CreateFileW
EIP  0ABC1385  FF D0           call eax
```

Sample tests if in a virtual environment

Tracking the Evasion Techniques’ Source Code

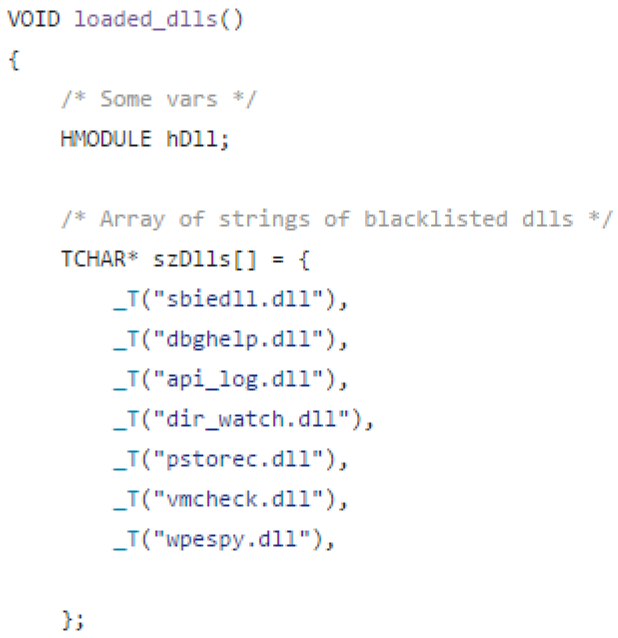
From our analysis, it is quite clear that the coders of this ransomware relied on existing lists of artifacts to create the above “DetectSandbox()” and “DetectVM()” functions.

We found some candidates for the source of the evasion techniques. In the image below, a snippet of code looks for sandbox solutions by the loaded DLLs:



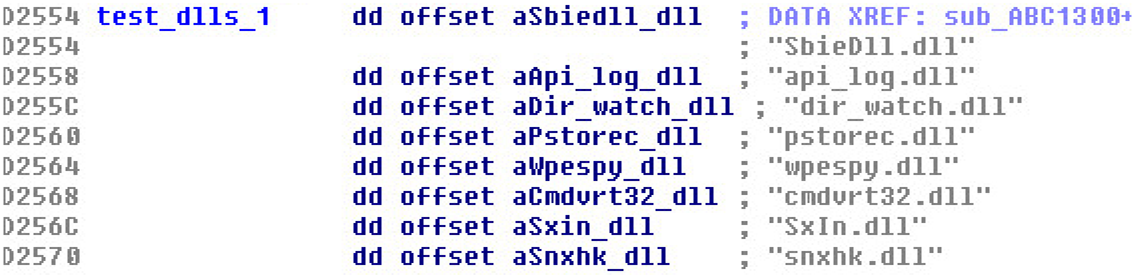
```
                foreach (ProcessModule pm in theMods)
                {
                    if ((pm.ModuleName).Contains("cmdvrt32.dll") || (pm.ModuleName).Contains("snxhk.dll") ||
(pm.ModuleName).Contains("SxIn.dll"))
                    {
                        inSandbox = true;
                    }
```

And in this source shows another list collected for the very same purpose:



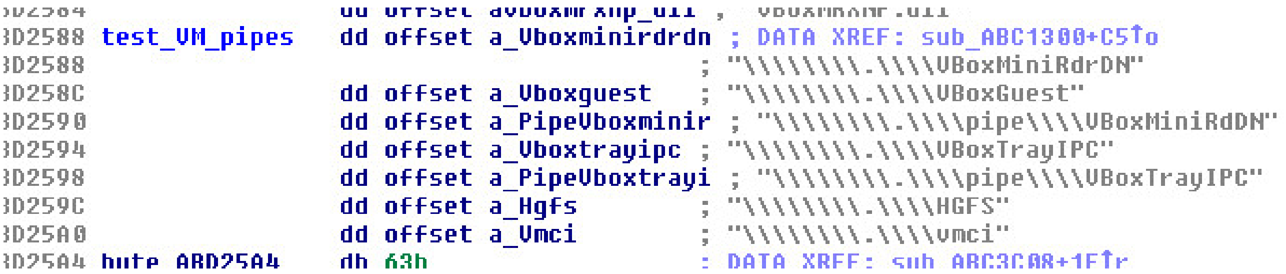
```
VOID loaded_dlls()
{
    /* Some vars */
    HMODULE hDll;

    /* Array of strings of blacklisted dlls */
    TCHAR* szDlls[] = {
        _T("sbiedll.dll"),
        _T("dbghelp.dll"),
        _T("api_log.dll"),
        _T("dir_watch.dll"),
        _T("pstorec.dll"),
        _T("vmcheck.dll"),
        _T("wpespy.dll"),

    };
```

It appears that those two lists were appended together in UIWIX (with dbghelp.dll and vmcheck.dll tested in a different function):



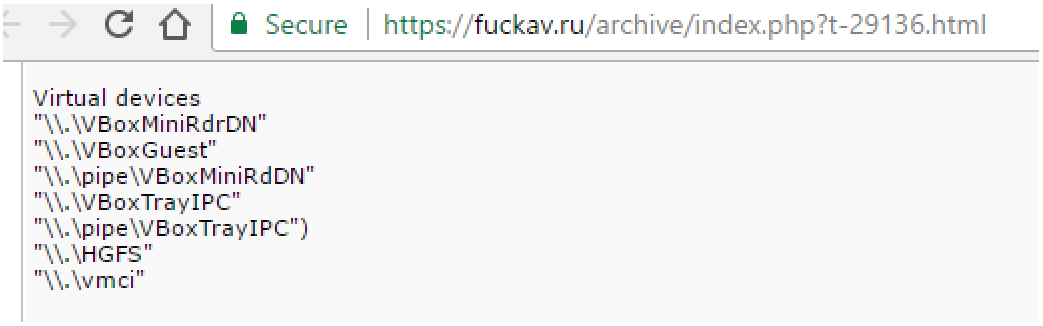
```
D2554 test_dlls_1     dd offset aSbiedll_dll  ; DATA XREF: sub_ABC1300+
D2554                                         ; "SbieDll.dll"
D2558                 dd offset aApi_log_dll  ; "api_log.dll"
D255C                 dd offset aDir_watch_dll ; "dir_watch.dll"
D2560                 dd offset aPstorec_dll  ; "pstorec.dll"
D2564                 dd offset aWpespy_dll   ; "wpespy.dll"
D2568                 dd offset aCmdvrt32_dll ; "cmdvrt32.dll"
D256C                 dd offset aSxin_dll     ; "SxIn.dll"
D2570                 dd offset aSnxhk_dll    ; "snxhk.dll"
```

Another interesting similarity is in the malware code section which tests for VM pipes:



```
3D2588 test_VM_pipes   dd offset a_Vboxminirdrdn ; DATA XREF: sub_ABC1300+C5↑o
3D2588                                         ; "\\\\\\\\.\\\\VBoxMiniRdrDN"
3D258C                 dd offset a_Vboxguest    ; "\\\\\\\\.\\\\VBoxGuest"
3D2590                 dd offset a_PipeVboxminir ; "\\\\\\\\.\\\\pipe\\\\VBoxMiniRdDN"
3D2594                 dd offset a_Vboxtrayipc ; "\\\\\\\\.\\\\VBoxTrayIPC"
3D2598                 dd offset a_PipeVboxtrayi ; "\\\\\\\\.\\\\pipe\\\\VBoxTrayIPC"
3D259C                 dd offset a_Hgfs         ; "\\\\\\\\.\\\\HGFS"
3D25A0                 dd offset a_Vmci         ; "\\\\\\\\.\\\\vmci"
3D25A4 byte_ABD25A4    db 63h                   ; DATA XREF: sub_ABC3CA8+1E↑r
```

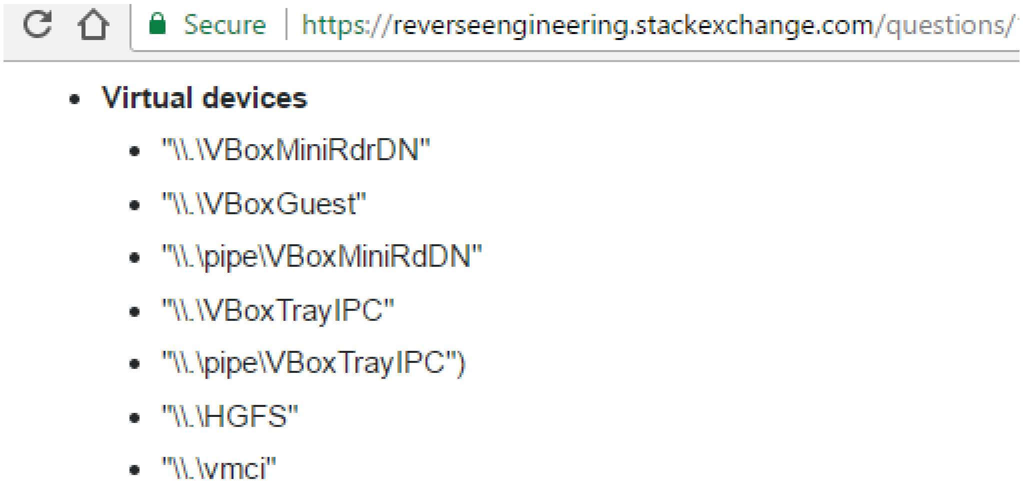
And this is how they appear in a Russian hacking forum called “FuckAV”:



Virtual devices
"\\.\VBoxMiniRdrDN"
"\\.\VBoxGuest"
"\\.\pipe\VBoxMiniRdDN"
"\\.\VBoxTrayIPC"
"\\.\pipe\VBoxTrayIPC")
"\\.\HGFS"
"\\.\vmci"

Note how the order of the artifacts is an exact match to the malware!

This list can also be found in legitimate websites:



- **Virtual devices**
  - "\\.\VBoxMiniRdrDN"
  - "\\.\VBoxGuest"
  - "\\.\pipe\VBoxMiniRdDN"
  - "\\.\VBoxTrayIPC"
  - "\\.\pipe\VBoxTrayIPC")
  - "\\.\HGFS"
  - "\\.\vmci"

Why Minerva Aces Against UIWIX

Minerva Anti-Evasion Platform creates a virtual reality that fools the malware, making it believe that it is in a hostile environment. Clever environmentally aware malware like UIWIX will avoid execution in a Minerva-protected endpoint as we make the malware believe it is in a VM or sandbox.

UIWIX is exploiting unpatched machines to execute its DLL without writing itself to the disk. Luckily, Minerva works against any type of evasive threat, including file-less attacks like this one.

IoC

Hashes

3860c2526fc8acf5366573cdeb0a292036398d3ee9e7d9764a60ec5d0812582a

146581f0b3fbe00026ee3ebe68797b0e57f39d1d8aecc99fdc3290e9cfadc4fc

Searched VM related DLLs

SbieDll.dll

api_log.dll

dir_watch.dll

pstorec.dll

wpespy.dll

cmdvrt32.dll

SxIn.dll

snxhk.dll

Searched Sandbox related DLLs

dbghelp.dll

vmcheck.dll

VBoxHook.dll

VBoxMRXNP.dll

Searched Sandbox Pipes

\\.\pipe\cuckoo

Searched VM Pipes

\\.\VBoxMiniRdrDN

\\.\VBoxGuest

\\.\pipe\VBoxMiniRdDN

\\.\VBoxTrayIPC

\\.\pipe\VBoxTrayIPC

\\.\HGFS

\\.\vmci

URLs

(as published by Lawrence Abrams in BleepingComputer)

hxxps://4ujngbdqqm6t2c53[.]onion[.]to

hxxps://4ujngbdqqm6t2c53[.]onion[.]cab

hxxps://4ujngbdqqm6t2c53[.]onion[.]nu

hxxps://4ujngbdqqm6t2c53[.]onion[.]to

hxxps://4ujngbdqqm6t2c53[.]onion[.]cab

hxxp://4ujngbdqqm6t2c53[.]onion