Tracing the Crimeware Origins by Reversing Injected Code

Reverse engineering
November 15, 2010 by **Giuseppe Bonfa**

Part 1: Introduction and De-Obfuscating and Reversing the User-Mode Agent Dropper
Part 2: Reverse Engineering the Kernel-Mode Device Driver Stealth Rootkit
Part 3: Reverse Engineering the Kernel-Mode Device Driver Process Injection Rootkit
Part 4:Tracing the Crimeware Origins by Reversing the Injected Code

In this final part we will trace the origins of the ZeroAccess rootkit. We will discover that the purpose of this rootkit is to set up a stealthy, undetectable and un-removable platform to deliver malicious software to victim computers. We will also see that ZeroAccess is being currently used to deliver FakeAntivirus crimeware applications that trick users into paying $70 to remove the antivirus. It could be used to delivery any malicious application, such as one that steals bank and credit card information in the future. Further analysis and network forensics supports that ZeroAccess is being hosted and originates from the Ecatel Network, which is controlled by the cybercrime syndicate RBN (Russian Business Network).

Let's take a look at the max++ DLL. It is injected into other processes address space by the kernel mode driver reversed in Part 3. Here are the hashes for this DLL:

> FileSize: 36.00 KB (36864 bytes)
>
> MD5: 4CE2F6BA808954FA7B1257D4C754D5B0
>
> SHA-1: E74EA961ADCA942623AE721283D33F6907A7C86D
>
> No VersionInfo Available.
>
> Resource: "TYPELIB".
>
> max++.00.x86.dll does not have an ET (Export Table) entire code is contained into DllEntryPoint() function.

This malicious DLL is injected via APC and ZwAllocateVirtualMemory by the second installed driver. Recall that this driver finds newly loaded processes via PsSetLoadImageNotifyRoutine.

```
35672E02                and     esp, 0FFFFFFF8h
35672E05                mov     eax, [ebp+fdwReason]
35672E08                sub     esp, 194h
35672E0E                push    ebx
35672E0F                xor     ebx, ebx
35672E11                sub     eax, ebx
35672E13                push    esi
35672E14                push    edi
35672E15                jz      short loc_35672E8C ; if (fdwReason == 1)
35672E17                dec     eax
35672E18                jnz     loc_35672F12    ; Exit
35672E1E                call    ds:ZwTestAlert  ; tests if current thread has been alerted
35672E24                push    [ebp+hLibModule] ; hLibModule
35672E27                call    ds:DisableThreadLibraryCalls
35672E2D                lea     eax, [esp+1A0h+WSAData]
35672E31                push    eax             ; lpWSAData
35672E32                push    202h            ; wVersionRequested
35672E37                call    ds:WSAStartup   ; load WinSock 2.2
35672E3D                mov     eax, offset unk_356761E0
35672E42                mov     edi, offset unk_35676200
35672E47                mov     esi, eax
35672E49                cmp     eax, edi
35672E4B                jnb     short loc_35672E5C
35672E4D
35672E4D loc_35672E4D:                          ; CODE XREF: DllEntryPoint+5B↓j
35672E4D                mov     eax, [esi]
35672E4F                add     esi, 4
35672E52                cmp     eax, ebx
35672E54                jz      short loc_35672E58
35672E56                call    eax             ; Call a different routine for each iteration
```

This DLL can be debugged by using OllyDbg 1.10 LoadDLL utility. As this component is the just a small feature of ZeroAccess, we won't go into as much detail on it as we did with the two drivers and the user-mode agent.

The above ZwTestAlert function tests whether the current thread has been alerted (and clears the alerted flag). It also enables the delivery of queued user APCs. NextDisableThreadLibraryCalls disables the DLL_THREAD_ATTACH and DLL_THREAD_DETACH notifications for the DLL. By disabling the notifications, the DLL initialization code is not paged in because a thread is created or deleted, thus reducing the size of the application's working code set. This use of DisableThreadLibraryCalls increases invisibility for the injected DLL.

The call EAX is placed inside a do-while that updates the value pointed by EAX in iteration. This involves in calling a different routine each time. Let's look at the next code block:

```
                call    eax             ; Call a different routine for each iteration

loc_35672E58:                           ; CODE XREF: DllEntryPoint+55↑j
                cmp     esi, edi
                jb      short loc_35672E4D

loc_35672E5C:                           ; CODE XREF: DllEntryPoint+4C↑j
                push    1
                call    sub_356752D7    ; Dll Stuff -> sub_356752D7(1)
                push    ebx             ; lpThreadId
                push    ebx             ; dwCreationFlags
                push    ebx             ; lpParameter
                push    offset StartAddress ; lpStartAddress
                push    ebx             ; dwStackSize
                push    ebx             ; lpThreadAttributes
                call    ds:CreateThread
                mov     handleThread, eax
                cmp     eax, 0FFFFFFFFh
                jnz     loc_35672F12
                mov     handleThread, ebx
                jmp     loc_35672F12    ; Exit
```

We see the API call completed. A thread is created, and we can inspect it by reaching the StartAddress argument. This new thread will invoke a couple of calls that will talk to a Command and Control (C&C) server. The C&C server will issue requests to websites that contain code to install further malicious code to be executed on the victim's machine.

From string analysis we can obtain some valuable information:

```
a??C2cad97240_0:                        ; DATA XREF: .rdata:356765A0↓o
                unicode 0, <\??\C2CAD972#4079#4fd3#A68D#AD34CC121074\L\{FF1D3D65-8EB9>
                unicode 0, <-4347-B8C5-C2EC822C6CC2}>,0
aV              db 'ó',0                ; DATA XREF: .data:356780A4↓o
aD_0            db 'ñ',0
                dd offset a??C2cad97240_0 ; "\\??\\C2CAD972#4079#4fd3#A68D#AD34CC12107"...
                align 8
a??C2cad97240_1:                        ; DATA XREF: .rdata:356766F0↓o
                unicode 0, <\??\C2CAD972#4079#4fd3#A68D#AD34CC121074\L\{7D718C60-B241>
                unicode 0, <-4bcc-97C5-888755504 60C}>,0
aV_0            db 'ó',0                ; DATA XREF: .data:356780B0↓o
aD_1            db 'ñ',0
                dd offset a??C2cad97240_1 ; "\\??\\C2CAD972#4079#4fd3#A68D#AD34CC12107"...
                align 8
a??C2cad97240_2:                        ; DATA XREF: .rdata:35676700↓o
                unicode 0, <\??\C2CAD972#4079#4fd3#A68D#AD34CC121074\L\{76620BB4-6C08>
                unicode 0, <-49d2-AE35-480385646585}>,0
aV_1            db 'ó',0                ; DATA XREF: .data:356780A4↓o
aD_2            db 'ñ',0
                dd offset a??C2cad97240_2 ; "\\??\\C2CAD972#4079#4fd3#A68D#AD34CC12107"...
                align 8
a??C2cad97240_3:                        ; DATA XREF: .rdata:356767B0↓o
                unicode 0, <\??\C2CAD972#4079#4fd3#A68D#AD34CC121074\L\{F8E9DFD7-1EFD>
                unicode 0, <-4F4a-96B1-5A693CF787B9}>,0
aV_2            db 'ó',0                ; DATA XREF: .data:356780BC↓o
aD_3            db 'ñ',0
                dd offset a??C2cad97240_3 ; "\\??\\C2CAD972#4079#4fd3#A68D#AD34CC12107"...
```

All these entries as you can see are placed into the hidden NTFS volume that was analyzed previously. There are also other interesting strings like:

registrymachineSoftwareMicrosoftInternet ExplorerMain{F9197A7E-CE10-458e-85F8-5B0CE6DF2BBE}

The CLSID encoding is of great help during malware forensics. It can be used to determine univocally malware type and version. A quick search show us that this CLSID belongs to Trojan-Ransom.Win32.Digitala.b which is a downloader Aagent.

The above code blocks are executed if fdwReason is satisfied. Otherwise execution flows to another block of code that essentially acts as a cleanup routine.

This injected DLL serves the purpose of generating web redirections to malicious websites that contain FakeAntivirus software. Fake antivirus software (a.k.a misleading applications or rogue antivirus) is big business nowadays with Symantec reporting 43 million installation attempts from over 250 distinct programs between July 1, 2009, to June 30, 2010. With fake AV software costing the victim anywhere from $30 to $100, this is a lucrative earner for criminals.

The malicious URLs are:

- http://intensedive.com/install/setup.php?m=d310b08f1d6d&i=1&id=000069000
- http://intensedive.com/install/setup.php?m=d310b08f1d6d
- http://intensedive.com/updates/cleaner.dll?m=d310b08f1d6d

The IP address behind these domains is 94.102.63.99. From www.robtex.com we can see the following graph

AS29073 belongs to Ecatel Network which is a well known crimeware friendly ISP.

http://hphosts.blogspot.com/2009/11/crimeware-friendly-isps-ecatel-as29073.html

Ecatel is infamous for the massive hosting of malware and spambots, the most widely used IPs are:

> - 94.102.60.151
> - 94.102.60.152
> - 94.102.60.153
> - 94.102.60.182
> - 94.102.60.43
> - 94.102.60.77

Detailed information on Ecatel activities can be seen here:
http://www.sudosecure.net/archives/333

Often Ecatel was involved into fakeAV campaigns, and ZeroAccess drives to fake software download. From sudosecure.net we see a relation with the well-know cybercrime ring, RBN (Russian Business Network).

Posted: November 15, 2010

Author

Giuseppe Bonfa

VIEW PROFILE

Giuseppe is a security researcher for InfoSec Institute and a seasoned InfoSec professional in reverse-engineering and development with 10 years of experience under the Windows platforms. He is currently deeply focused on Malware Reversing (Hostile Code and Extreme Packers) especially Rootkit Technology and Windows Internals. He has previously worked as Malware Analyst for Comodo Security Solutions as a member of the most known Reverse Engineering Teams and is currently a consultant for private customers in the field of Device Driver Development, Malware Analysis and Development of Custom Tools for Digital Forensics. He collaborates with Malware Intelligence and Threat Investigation organizations and has even discovered vulnerabilities in PGP and Avast Antivirus Device Drivers. As a technical author, Giuseppe has over 10 years of experience and hundreds of published pieces of research.