

EXHIBIT 11

FILED UNDER SEAL

ZeroAccess

ZeroAccess

James Wyke

SophosLabs UK

Abstract

ZeroAccess is a sophisticated kernel-mode rootkit that is rapidly becoming one of the most widespread threats in the current malware ecosystem. ZeroAccess' ability to run on both 32-bit and 64-bit versions of Windows, resilient peer-to-peer command and control infrastructure and constant updates to its functionality over time show that ZeroAccess is a modern threat capable of thriving on modern networks and modern Operating Systems.

In this paper we will explore the ZeroAccess threat; from the distribution mechanisms used to spread it, through the installation procedure, memory residence and payload. We examine how ZeroAccess works and what its ultimate goal is.

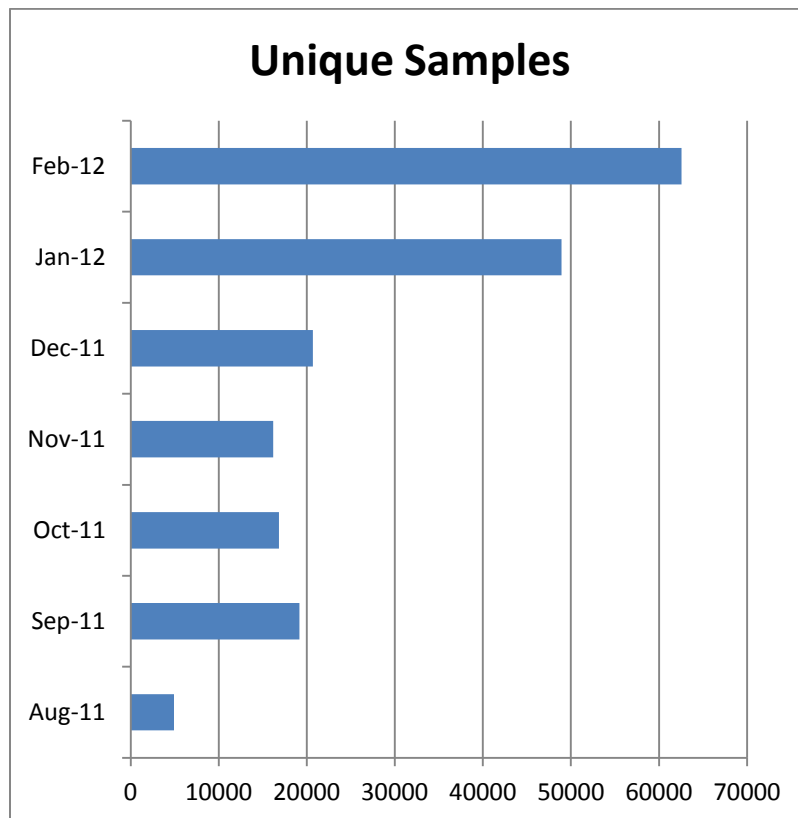
Contents

ZeroAccess.....	4
Distribution	5
Exploit Packs	5
Social Engineering.....	5
Dropper	7
Installation.....	8
Pseudo Random Domain Generator.....	11
32-Bit Installation	12
64-Bit Installation	15
Memory Residence	16
Rootkit.....	16
Payload.....	18
Conclusion	21
References.....	22
Appendix.....	23
P2P RC4 Key	23
RSA Public key.....	23

ZeroAccess

ZeroAccess is a dangerous threat that has been circulating for several years. SophosLabs has recently seen the number of machines infected with ZeroAccess increase sharply as there has been a proliferation of samples appearing in the wild.

The following graph shows the upwards trend in the number of unique ZeroAccess related samples seen by SophosLabs over the last 7 months:



In the time that ZeroAccess has been in the wild there have been a number of revisions, with modifications to its functionality, infection strategy and its persistence mechanisms on an infected machine. However, the core purpose has remained: to assume full control of the machine by adding it to the ZeroAccess botnet and to monetize the new asset by downloading additional malware.

Primarily ZeroAccess is a kernel-mode rootkit, similar in ethos to the TDL family of rootkits. It uses advanced techniques to hide its presence, is capable of functioning on both 32 and 64-bit flavors of Windows from a single installer, contains aggressive self defense functionality and acts as a sophisticated delivery platform for other malware.

Distribution

Infection vectors for ZeroAccess are very similar to other high profile malware families currently circulating in the wild. Although not entirely comprehensive the main distribution methods for ZeroAccess can be split into two categories: Exploit Packs and social engineering.

Exploit Packs

ZeroAccess has become an increasingly popular payload to the various Exploit Packs currently on the market, in particular BlackHole [1]. An exploit pack typically comes as a series of php scripts that are stored on a web server under the control of the attacker. When a victim's browser accesses the loaded website the server backend will attempt to exploit a vulnerability on the target machine and execute the payload. Exploit packs usually contain a great many different exploits targeting applications commonly found on Windows PCs such as Internet Explorer, Acrobat, Flash and Java.

Traffic is driven to websites hosting Exploit Packs through a variety of means. A common method is through the use of compromised sites. Legitimate sites that have been compromised by the attacker (often through stolen FTP credentials or SQL injection) are used to both host the exploit packs themselves and as redirectors to the main attack site. Typically, small amounts of JavaScript code are inserted into pages of a compromised website that will send the user to the attack site. Ad servers have also been compromised in this way which can result in widespread infection very quickly if the ads are served to high profile websites. SEO (Search Engine Optimisation) techniques are used to drive compromised websites up search engine rankings, increasing the traffic that gets sent to the attack site. We have also seen this delivery method initiated through email. An email is spammed out containing a link that when clicked sends the victim to a compromised website hosting an Exploit Pack. Exploit Packs as an infection vector for ZeroAccess are very effective and usually requires no input from the victim other than browsing to an apparently legitimate website or clicking an innocuous-seeming link.

Social Engineering

The second main infection vector for ZeroAccess is through a variety of social engineering techniques. At the heart of these is the goal of convincing a victim into running an executable that they should not. The lure is often a piece of illicit software such as a game or a copyright protection bypassing tool such as a crack or keygen. These Trojanised files are placed on upload sites and on torrents and given filenames designed to trick the unwary into downloading and running them.

The following is an example of a file purporting to be a keygen for DivX Plus 8.0 for Windows. The file would be placed onto upload sites or offered as a torrent. The file is in fact an NSIS self extractor that contains the advertised keygen program but also contains an encrypted

ZeroAccess

7zip file. When executed the self extractor unpacks the keygen program to "%Profile%\Application Data\Keygen.exe" and executes it:

[fig_1.png]



But in the background the 7zip file is dropped, extracted and the single file inside (the ZeroAccess dropper) is executed. By observing API calls the 7zip password can be ascertained:

[fig_2.png]

```
CreateProcess(NULL, ""C:\DOCUME~1\support\LOCALS~1\Temp\7za.exe  
"C:\DOCUME~1\support\LOCALS~1\Temp\al.7z" -aoa -oC:\DOCUME~1\s  
-p~@523js@vBz99432@t9 ""
```

Here is an example where the lure was a copy of the game Skyrim [2]. Again the installer is an NSIS archive. This time a file is dropped to "%Profile%\Application Data\skyrimlauncher.exe" and a screen is shown that purports to be the game installer:



[fig_3.png]

While once again in the background an encrypted 7Zip file is dropped, extracted and the contents executed, installing ZeroAccess.

Dropper

ZeroAccess droppers have changed as the rootkit itself has evolved. Currently, droppers are usually packed with one from a group of complex polymorphic packers.

These packers are a typical example of the protection measures that modern malware employs to both hinder analysis and to attempt to avoid detection by security tools. They are updated several times a day and are always checked against AV scanners before they are released into the wild. These packers contain a great many anti-emulation and anti-debug techniques designed to defeat emulators inside AV engines and to make analysis inside a controlled environment more difficult. The dropper has recently been using hardware breakpoints as part of its unpacking routine which makes attaching a kernel debugger to the target system (necessary to analyse the kernel-mode components) more challenging.

ZeroAccess

An interesting feature of ZeroAccess droppers is that a single dropper will install the 32-bit or the 64-bit version of the malware depending on which OS it is executed under.

Installation

At installation ZeroAccess will first ascertain if it is running on 32 or 64-bit Windows. This is achieved using the *ZwQueryInformationProcess* [3] API with *ProcessWow64Information* as the *ProcessInformationClass* parameter:

[fig_4.png]

```
push    eax
push    4
lea     eax, [esp+1D8h+IsX64]
push    eax
push    1Ah                                ; ProcessWow64Information
push    0FFFFFFFFh
call    ds:ZwQueryInformationProcess ; check if 64 bit
cmp     [esp+1D0h+IsX64], edi
jz      short x32
```

This is where the decision between 32 bit and 64-bit installation path is made. The installer will then attempt to give itself *SE_DEBUG_PRIVILEGE* [4] privileges using *RtlAdjustPrivileges*:

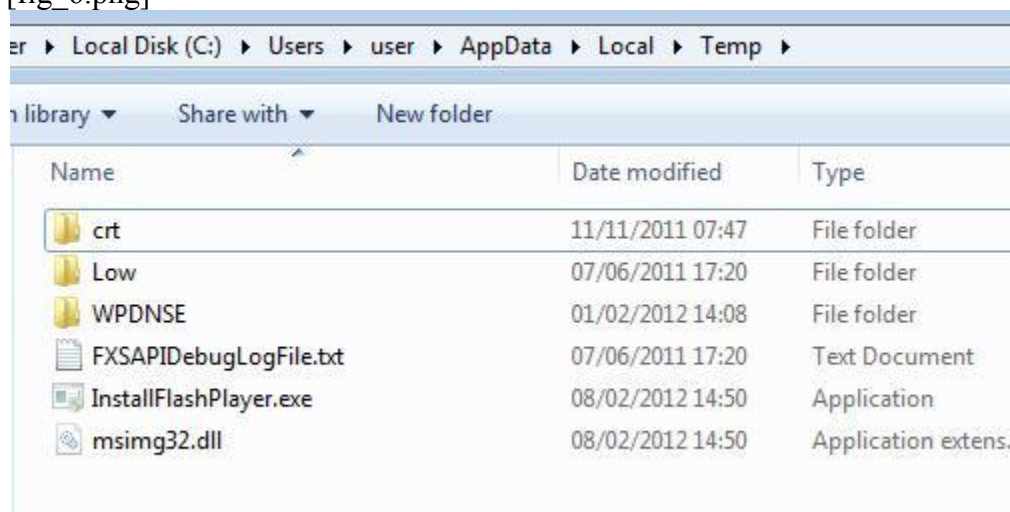
[fig_5.png]

```
mov     ebp, esp
sub     esp, 14h
push    ebx
push    esi
push    edi
lea     eax, [ebp+var_1]
push    eax                                ; enabled
xor     edi, edi
push    edi                                ; CurrentThread
push    1                                  ; enable
push    14h                                ; SE_DEBUG_PRIVILEGE
call    ds:RtlAdjustPrivilege
test    eax, eax
jl      short priv_adjust_failed
xor     eax, eax
jmp     done
```

If this is successful then installation will continue as normal, if the attempt fails (usually because the process has been executed by a normal user) then ZeroAccess will attempt another method of privilege escalation.

ZeroAccess must elevate its privileges to install successfully, but in order to do this from a non-administrator account on UAC enabled versions of Windows, a UAC popup will appear. End users are more likely to be suspicious of a file they have just downloaded from the internet that they thought was an illegal keygen, crack or hacked version of a game; they may also be suspicious if an unknown exe file causes a UAC popup while the user is browsing the web (exploit pack infection vector). As a result the user may choose not to allow the program to proceed, thus ZeroAccess installation may fail. To bypass this possible problem ZeroAccess disguises itself by forcing the UAC popup to appear to come from a different, benign-seeming program. A clean copy of the Adobe Flash Installer (*InstallFlashPlayer.exe*) is dropped to a temporary directory and the DLL load order of Windows [5] is abused to ensure that ZeroAccess is loaded into the clean file's process address space when it is executed. By dropping a DLL called *msimg32.dll* (one of the DLL's that *InstallFlashPlayer.exe* imports) into the same directory as the Flash installer file, Windows will load this DLL in preference to the genuine *msimg32.dll* because Windows looks in the current directory before the system directory when loading DLL's:

[fig_6.png]



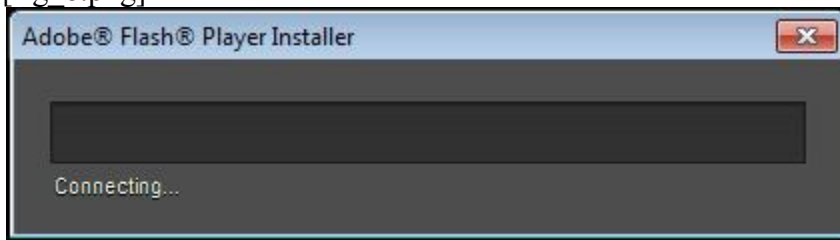
This means that the UAC popup now appears to be generated by the genuine Adobe Flash Installer which is much more likely to be authorized:



[fig_7.png]

The Flash installer will then continue while ZeroAccess silently infects the system in the background, even if Flash is already installed:

[fig_8.png]



ZeroAccess will next go about lowering security on the infected machine by disabling a number of Windows security related services. The Windows Firewall is turned off and updates will no longer be retrieved from Microsoft. The full list of services that it will attempt to disable is below:

- BFE (Base FilteringEngine Service)*
- iphlpvc (IP Helper service)*
- mpssvc (Windows firewall service)*
- WinDefend (Windows Defender service)*
- wscsvc (Windows Security Center Service)*
- WinHttpAutoProxySvc (Proxy Auto Discovery Service)*

Pseudo Random Domain Generator

During installation many ZeroAccess samples will report back to an IP address embedded inside the executable with information about the infected machine. This is carried out with an HTTP Get request with the “Host” field of the request set to a pseudo-randomly generated “.cn” domain.

The generated domain name does not exist and does not need to exist as it is never looked up and no attempt is made to connect to any URL on the generated domain. The domain uses the current date and a seed value and one domain will be generated per day:

[fig 10.png]

```

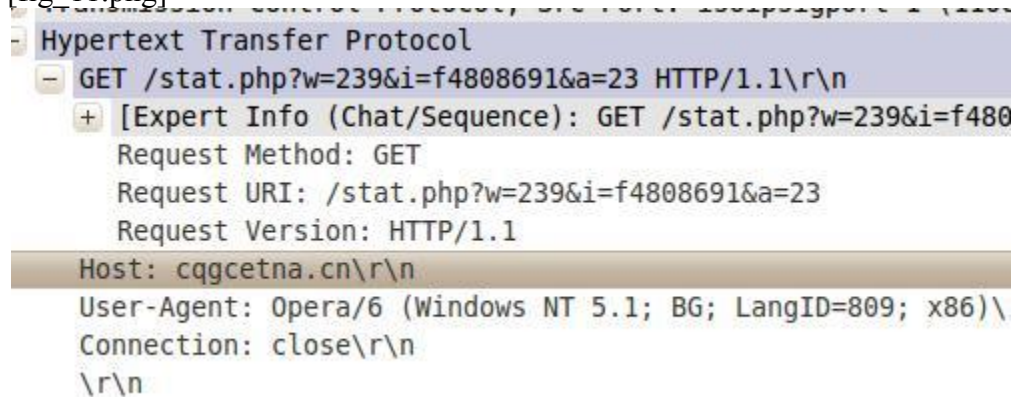
push    eax                                ; ]
call    ds:GetSystemTimeAsFileTime
lea     eax, [ebp+var_1C]
push    eax
lea     eax, [ebp+SystemTimeAsFileTime]
push    eax
call    ds:RtlTimeToTimeFields
mov     edi, ds:RtlComputeCrc32
push    6
lea     eax, [ebp+var_1C]
push    eax
push    0
call    edi ; RtlComputeCrc32
mov     esi, eax
push    6
lea     eax, [ebp+var_1C]
push    eax
xor     ebx, ebx
shld    ebx, esi, 8
push    87878787h
shl     esi, 8
call    edi ; RtlComputeCrc32
xor     edx, edx
xor     eax, esi
xor     edx, ebx

c_40268F:                                ; {
mov     esi, [ebp+arg_0]
mov     ecx, eax
and     ecx, 1Fh
mov     cl, byte ptr aCnqazwsxedcrfvtge;
inc     [ebp+arg_0]
mov     [esi], cl
mov     cl, 5
call    _allshr
mov     ecx, eax
or      ecx, edx
jnz     short loc_40268F
mov     eax, [ebp+arg_0]
pop     edi
pop     esi
mov     dword ptr [eax], 'nc.' |
non     ahv

```

This DGA (Domain Generation Algorithm) system is used in various places throughout ZeroAccess where communication needs to take place over HTTP. If the exact same HTTP request is made with an incorrect “Host” field in the HTTP request then an empty response will be returned. In this way it is used as a pseudo-authentication system to verify that the server is only talking to a genuine ZeroAccess instance and not anything else such as security researchers or search bots. The locale and the architecture of the infected machine are also added to the get request in the “User-Agent” field:

[fig_11.png]



The rest of ZeroAccess installation is markedly different on 32 and 64-bit platforms.

32-Bit Installation

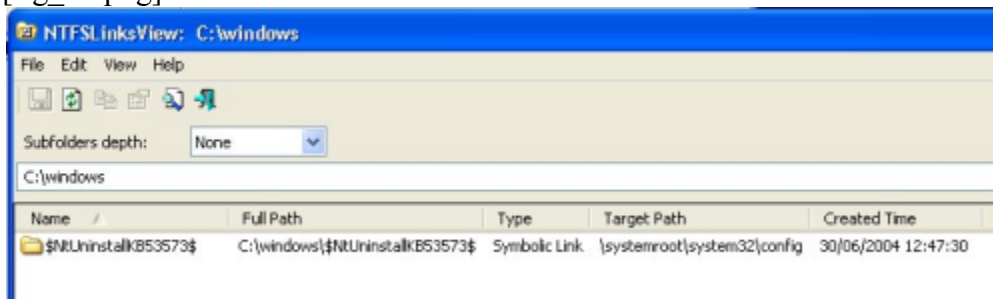
When installed under 32-bit Windows ZeroAccess will install a kernel-mode rootkit. To load its code into the kernel an existing driver will be overwritten on disk. The original driver file and any subsequent files downloaded by ZeroAccess will be stored in encrypted form on a part of the disk not normally accessible to other applications.

The stealth technique used by ZeroAccess to hide its files has changed over time. Previous versions created an entire hidden volume to store their files but recent versions have used a much simpler but no-less effective technique. A directory is created under “%systemroot%” with a name designed to look like a legitimate directory created when a Microsoft patch is installed. The directory name will be similar to: “c:\windows\NtUninstallKB35373\$” with the last five digits being specific to the victim machine. Files stored inside this folder are encrypted using a modified version of RC4 and to

ZeroAccess

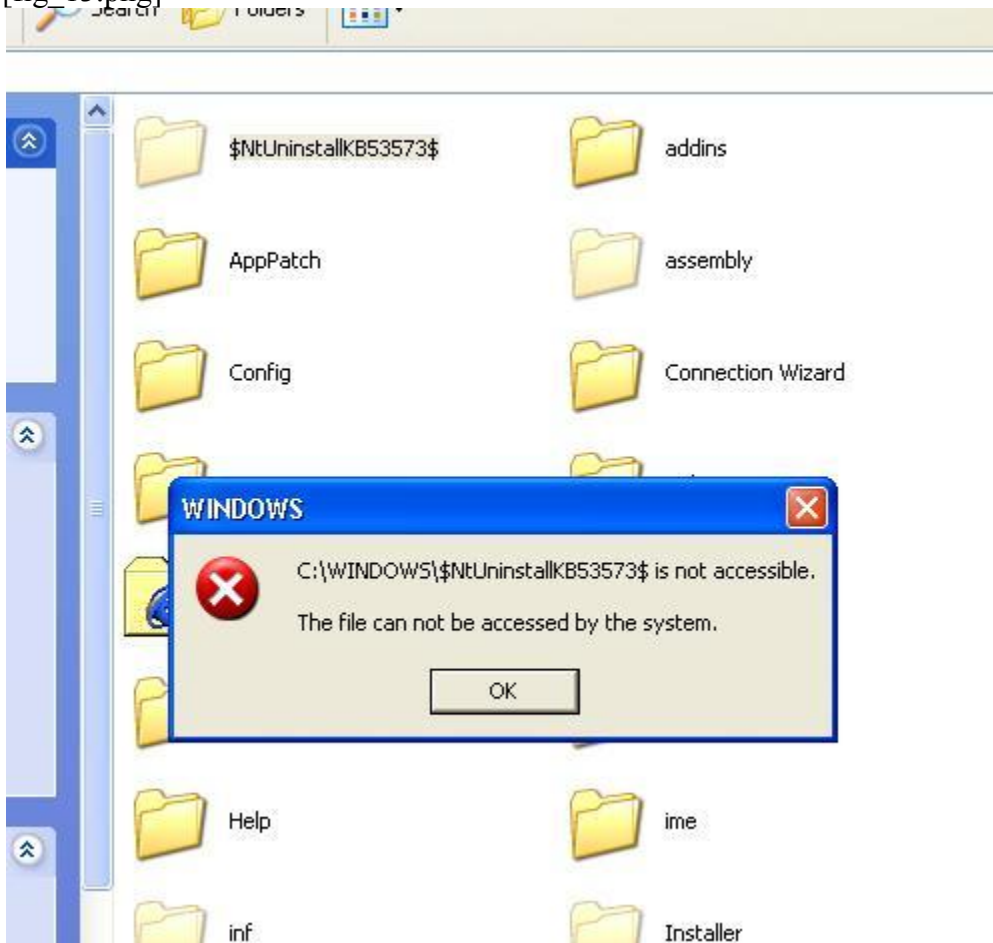
make the folder inaccessible to programs using standard Windows APIs it is made into a symbolic link pointing to “\systemroot\system32\config”:

[fig_12.png]



The ACL (Access Control List) on the directory is also changed so that the target of the link cannot be browsed to:

[fig_13.png]



However, if the raw disk is accessed or the directory is examined offline from a Linux OS the contents can be seen:

[fig_14.png]

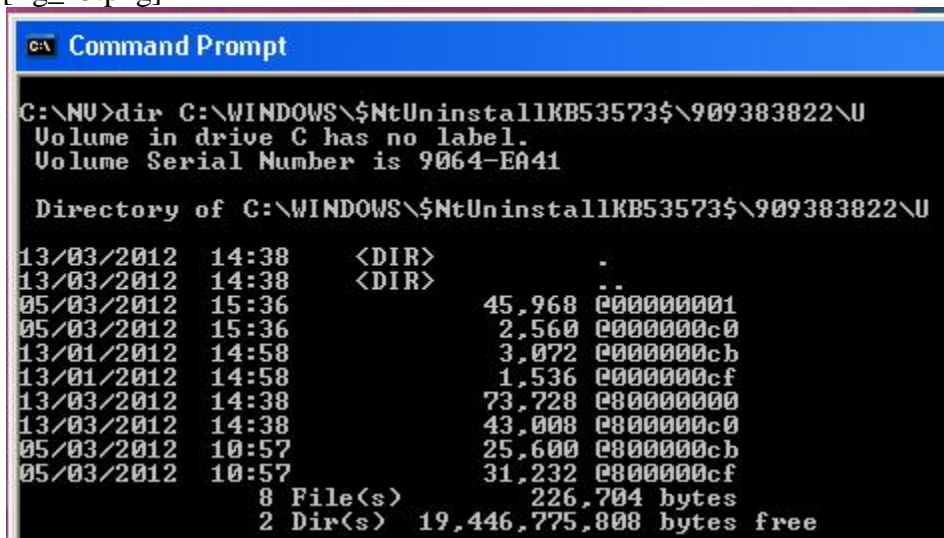
```

root@slax:/dev# ls /mnt/hda1/WINDOWS/\$NtUninstallKB53573\$/
1490692407* 909383822/
root@slax:/dev# ls /mnt/hda1/WINDOWS/\$NtUninstallKB53573\$/909383822/
\0*  loader.tlb*
root@slax:/dev# ls /mnt/hda1/WINDOWS/\$NtUninstallKB53573\$/909383822/U
\000000001* \00000000cb* \080000000* \08000000cb*
\0000000c0* \0000000cf* \0800000c0* \0800000cf*
root@slax:/dev# ls /mnt/hda1/WINDOWS/\$NtUninstallKB53573\$/909383822/L
akygdmgo*
root@slax:/dev#

```

If the symlink directory is ignored and the full path is provided the encrypted files can also be accessed:

[fig_15.png]



```

C:\>dir C:\WINDOWS\$NtUninstallKB53573$\909383822\U
Volume in drive C has no label.
Volume Serial Number is 9064-EA41

Directory of C:\WINDOWS\$NtUninstallKB53573$\909383822\U

13/03/2012  14:38    <DIR>          .
13/03/2012  14:38    <DIR>          ..
05/03/2012  15:36               45,968  000000001
05/03/2012  15:36               2,560  0000000c0
13/01/2012  14:58                3,072  0000000cb
13/01/2012  14:58                1,536  0000000cf
13/03/2012  14:38              73,728  080000000
13/03/2012  14:38              43,008  0800000c0
05/03/2012  10:57              25,600  0800000cb
05/03/2012  10:57              31,232  0800000cf
               8 File(s)              226,704 bytes
               2 Dir(s)  19,446,775,808 bytes free

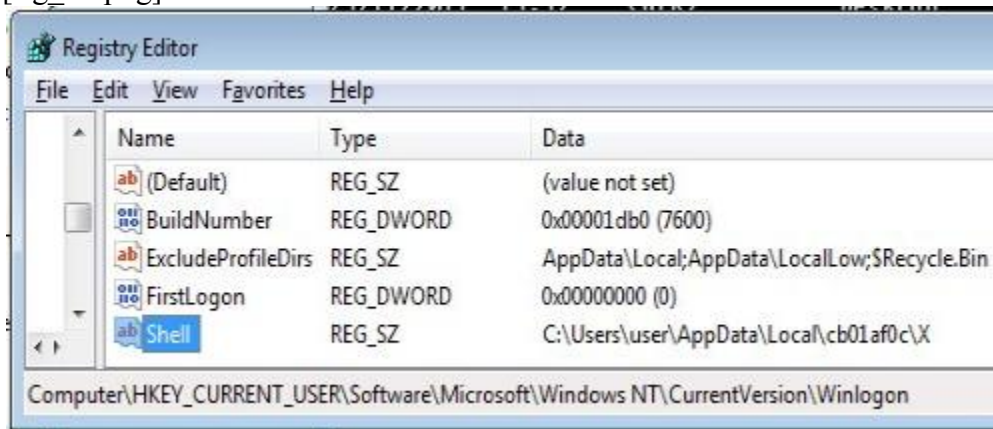
```

A special device is created that the rootkit uses to read and write into the hidden folder, decrypting and encrypting files on the fly, and the *LowerDeviceObject* of the *DR0* device of *\Driver\Disk* is hooked to hide the overwritten driver.

64-Bit Installation

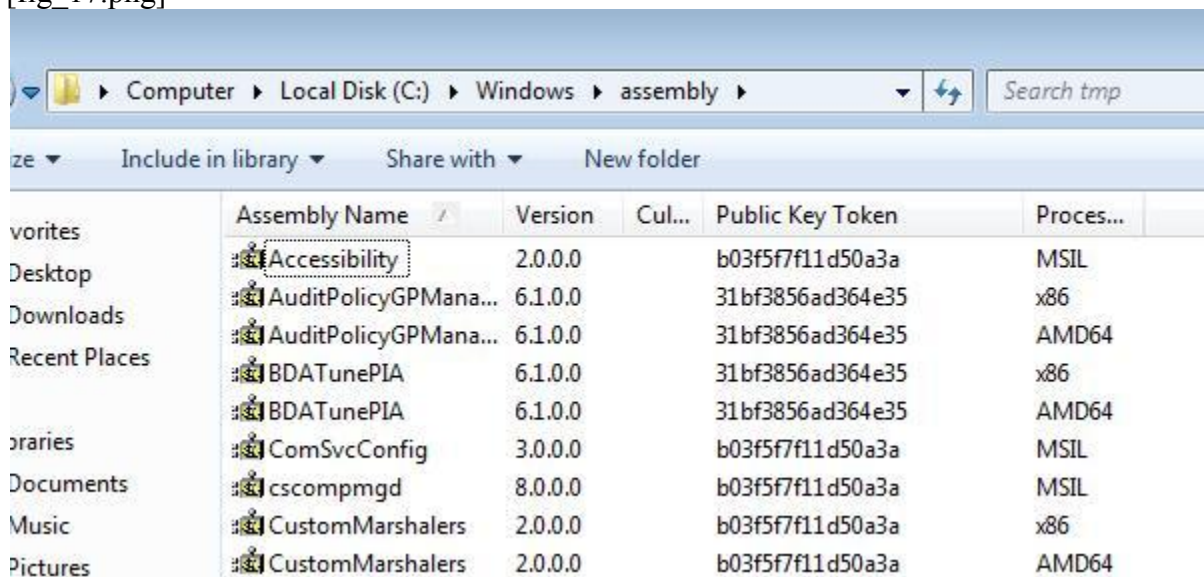
Under 64-bit Windows ZeroAccess does not use any kernel-mode code; all its execution takes place in user memory. The same technique of dropping the Flash Installer is used to elevate privileges. Reboot persistence is achieved through a file dropped into the user's AppData folder and a registry entry under *HKCU\Software\Microsoft\Windows NT\CurrentVersion\Winlogon*:

[fig_16.png]



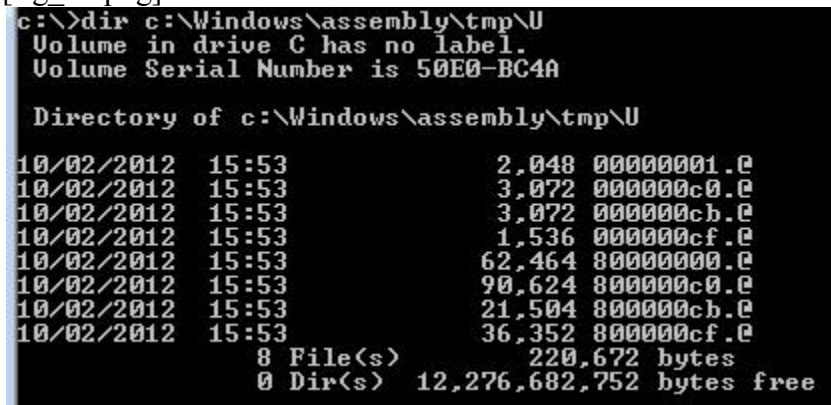
Again ZeroAccess attempts to make its files difficult to access, this time using no rootkit behaviour but by hiding inside the Global Assembly Cache (GAC). The GAC is a machine-wide cache of .NET assemblies used by Windows. It is located at %windir%\assembly. To view assemblies and add new ones, the Assembly Cache Viewer is used which is integrated into Explorer. When the GAC directory is browsed then the Cache Viewer is launched:

[fig_17.png]



The Cache viewer does not display the contents of the directory but displays information about the installed assemblies. ZeroAccess abuses this functionality by creating a directory inside the GAC folder and storing its files there. These files will not be observable by any casual user who browses the folder with Explorer but they can be seen if the Cache Viewer shell extension is disabled or from the command line:

[fig_18.png]



```
c:\>dir c:\Windows\assembly\tmp\U
Volume in drive C has no label.
Volume Serial Number is 50E0-BC4A

Directory of c:\Windows\assembly\tmp\U

10/02/2012  15:53                2,048 00000001.e
10/02/2012  15:53                3,072 000000c0.e
10/02/2012  15:53                3,072 000000ch.e
10/02/2012  15:53                1,536 000000cf.e
10/02/2012  15:53               62,464 80000000.e
10/02/2012  15:53               90,624 800000c0.e
10/02/2012  15:53               21,504 800000ch.e
10/02/2012  15:53               36,352 800000cf.e
               8 File(s)              220,672 bytes
               0 Dir(s) 12,276,682,752 bytes free
```

Memory Residence

Once ZeroAccess is in memory there are two main areas of activity: the rootkit and the payload.

Rootkit

If running under 32-bit Windows ZeroAccess will employ its kernel-mode rootkit. The rootkit's purpose is to:

- Hide the infected driver on the disk
- Enable read and write access to the encrypted files
- Deploy self defense (some versions)

The primary function of the rootkit component of ZeroAccess is to hide the changes made to the driver that was infected during installation. This is achieved by hooking the

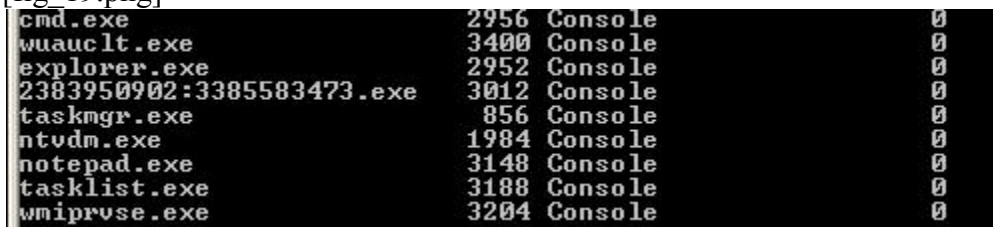
ZeroAccess

LowerDeviceObject of the *DR0* device of *\Driver\Disk*. A copy of the clean driver is stored in memory. Any process that attempts to read the infected driver from the disk will be presented with the clean driver.

If any of the components of ZeroAccess want to read or write to files stored inside the hidden folder then they need to do this without using the normal Win32 APIs as Windows will see the folder as a symbolic link and not realize it is also a genuine folder with files inside. The files also need to be decrypted to make any sense out of them. The rootkit driver facilitates seamless read and write to the hidden folder by creating a device named *ACPI#PNP0303#2&da1a3ff&0*. When files are accessed through this device they are decrypted on the fly.

Many versions of ZeroAccess employ aggressive self defense that is designed to protect the rootkit from security and AV software. A process is created that is monitored by the rootkit. If any application attempts to open this “bait” process then the rootkit will attack that application. The bait process has data stored in an Alternate Data Stream so the process name appears with a colon inside it:

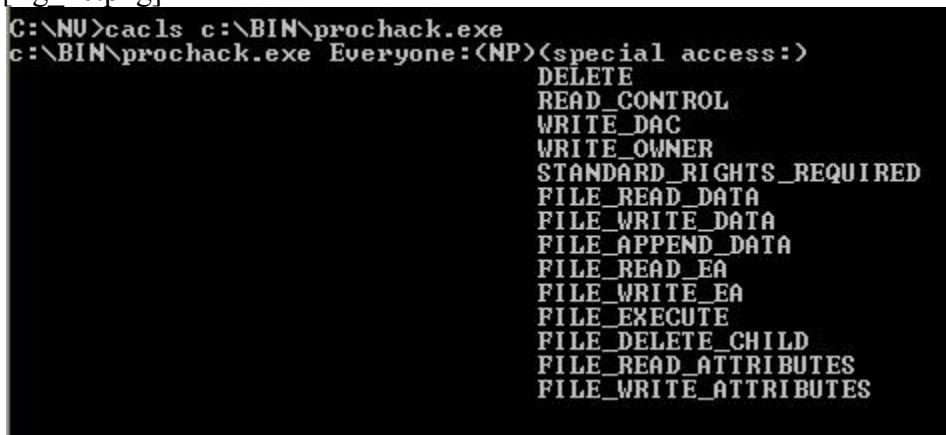
[fig 19.png]



Process Name	PID	Session Name	Session ID
cmd.exe	2956	Console	0
wuauc1t.exe	3400	Console	0
explorer.exe	2952	Console	0
2383950902:3385583473.exe	3012	Console	0
taskmgr.exe	856	Console	0
ntvdm.exe	1984	Console	0
notepad.exe	3148	Console	0
tasklist.exe	3188	Console	0
wmiprvse.exe	3204	Console	0

First, the ACL of the file for the process that has opened the bait process is changed so that the file can no longer be executed, using *ZwSetSecurityObject*:

[fig 20.png]



```
C:\>cac1s c:\BIN\prochack.exe
c:\BIN\prochack.exe Everyone:(NP)<special access:>
DELETE
READ_CONTROL
WRITE_DAC
WRITE_OWNER
STANDARD_RIGHTS_REQUIRED
FILE_READ_DATA
FILE_WRITE_DATA
FILE_APPEND_DATA
FILE_READ_EA
FILE_WRITE_EA
FILE_EXECUTE
FILE_DELETE_CHILD
FILE_READ_ATTRIBUTES
FILE_WRITE_ATTRIBUTES
```

The process itself is then attacked by injecting shell code into it that will terminate the process. This means that on ZeroAccess infected systems many security tools will be terminated and the ACL on their files will need to be changed before they can be executed again. This symptom is a good indicator of ZeroAccess infection and it would appear that the authors may have decided that this is **too** good an indicator of infection as most recent samples no longer include the self defense.

Payload

The payload of ZeroAccess is to connect to a peer-to-peer botnet and download further files. The network communication is initiated both from the kernel driver itself and from a component injected into user memory, usually inside either the address space of explorer.exe or svchost.exe, by the driver.

When initially installed, ZeroAccess includes with it a file that contains a list of 256 (0x100) IP addresses. Each IP address is followed by a dword time value that probably indicates the last contact time for each IP address as the list is sorted by the time value, highest first. This is the initial list of peers that the infected machine knows about in the botnet. The bot will attempt to contact each IP address in the list on a fixed port number that is stored inside the bot executable file. Once a successful connection is made commands will be issued. The bot also listens on the same high numbered TCP port that outgoing connections use, thus it attempts to become another node in the peer-to-peer botnet. However, it should be noted that the infected machine will need to be directly accessible from the internet with a public IP address for other peers to connect to it. Otherwise the infected machine will effectively become a passive node that can only connect to other nodes and obtain data; it cannot be connected to by other nodes.

All communication across the peer-to-peer network is encrypted with RC4 using a fixed key. This key has been observed to be the same for all variants of ZeroAccess encountered, even variants that use different port numbers and are instructed to download different types of malware. Upon successful connection to another node the bot will first issue a 'getL' command. This command is regularly repeated and is the main way of keeping up to date with other nodes. The other node then responds with a 'retL' command which includes the list of 256 (IP address, time) pairs that it currently holds and a list of files and timestamps for each file that it has downloaded. This keeps new nodes in the botnet updated with the currently accessible peers. A 'getF' command is then issued by the bot for each file contained in the list. This downloads the file and stores it under the hidden folder. Some variants will also store the downloaded files in a directory under the user's %AppData% path. Each downloaded file contains a resource named "33333" that contains a digital signature for the file. The bot verifies the signature is genuine using an RSA public key embedded inside it before the file is executed:

ZeroAccess

```
.
push    CALG_MD5                ; Algid
push    [ebp+hProv]             ; hProv
call    ds:CryptCreateHash
test    eax, eax
jz      short loc_AA3B56
push    esi
mov     esi, ds:CryptHashData
push    edi                     ; dwFlags
push    4                       ; dwDataLen
lea     eax, [ebp+FilenameDword]
push    eax                     ; pbData
push    [ebp+hHash]             ; hHash
call    esi ; CryptHashData     ; add hash of filename
test    eax, eax
jz      short loc_AA3B4C
push    edi                     ; dwFlags
push    [ebp+dwDataLen]         ; dwDataLen
push    [ebp+DataToVerify]      ; pbData
push    [ebp+hHash]             ; hHash
call    esi ; CryptHashData     ; add hash of file data
                                           ; with 123.. string

test    eax, eax
jz      short loc_AA3B4C
push    edi                     ; dwFlags
push    edi                     ; szDescription
push    [ebp+hPubKey]           ; hPubKey
push    40h                     ; dwSigLen
push    [ebp+pbSignature]       ; pbSignature
push    [ebp+hHash]             ; hHash
call    ds:CryptVerifySignatureW
mov     ebx, eax
```

[fig_21.png]

ZeroAccess has been seen to be downloading two main families of malware. The first is a type of click fraud [6] malware that appears to be very tightly bound to ZeroAccess, so much so that it may have been authored by the ZeroAccess owners. This malware can redirect browser search results to URL's of the author's choosing and will periodically query a server that will send back an xml file that contains a list of URL's and referrer URL's:

```

<xml>
  <doc>
    <url>http://184.171.169.130/click.php?c=
83ca737c464ae7dfdc808822e1bd2e029b1a15f4a2663831ded4e
    <ref>http://nofilter.com/?afdt=YWJhYT
    <n>5</n>
  </doc>
  <doc>
    <url>http://184.95.47.50/click.php?c=
4a6618b977021e57cc28624f25863abcb883ceea8e66daac69b07
    <ref>http://nofilter.com/?afdt=YWJhYT
    <n>5</n>
  </doc>

```

[fig_22.png]

The infected machine will send HTTP requests to each URL specified in the “<url>” tag with the *Referer* field of the HTTP request set to the URL from the “<ref>” field. This generates income for the affiliate whose ID is embedded in the referrer URL. The click fraud payload can be said to be very tightly bound to ZeroAccess itself because the same DGA (Domain Generation Algorithm) is used to generate the *Host* field of the HTTP request when retrieving URL data:

[fig_23.png]

```

+ [SEQ/ACK analysis]
- Hypertext Transfer Protocol
- GET /new/links2.php?w=192 HTTP/1.0\r\n
+ [Expert Info (Chat/Sequence): GET /new/links2.php?w=192
  Request Method: GET
  Request URI: /new/links2.php?w=192
  Request Version: HTTP/1.0
  Host: ajlnjugl.cn\r\n
  User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:9.0.1) Gecko/
  Connection: close\r\n
  \r\n

```

The other main payload is a spambot. When this payload is downloaded it installs itself, downloads spam templates and target email addresses and sends spam. It is likely that the authors of the spambot are renting a portion of the ZeroAccess botnet to deliver their malware.

The two differing version are most easily identified by the port numbers that they use. The click fraud downloading variant tends to use ports 21810 and 22292 whereas the spambot downloading variety uses port 34354.

Conclusion

We have explored where ZeroAccess infections come from, how the rootkit establishes control over a system and what activities it carries out once installed.

We can say that ZeroAccess is an advanced malware delivery platform that is controlled through a difficult to crack peer-to-peer infrastructure. Once it gains a foothold on a system it can be very difficult to remove. It has adapted as its target environment has evolved, adding compatibility for 64-bit architectures and multi-user, multi-privilege systems.

ZeroAccess remains hidden on an infected machine while downloading more visible components that generate revenue for the botnet owners. Currently the downloaded malware is mostly aimed at sending spam and carrying out click fraud, but previously the botnet has been instructed to download other malware and it is likely that this will be the case again in the future. ZeroAccess should be considered an advanced and dangerous threat that requires a fully featured, multi-layered protection strategy.

References

- [1] Sophos Security Threat Report 2012, Anatomy of an attack: Drive-by downloads and Blackhole, <http://www.sophos.com/en-us/security-news-trends/reports/security-threat-report/html-09.aspx>
- [2] Skyrim, [http:// www.elderscrolls.com/skyrim/](http://www.elderscrolls.com/skyrim/)
- [3] <http://msdn.microsoft.com/en-us/library/windows/desktop/ms687420%28v=vs.85%29.aspx>
- [4] <http://msdn.microsoft.com/en-us/library/windows/desktop/bb530716%28v=vs.85%29.aspx>
- [5] <http://msdn.microsoft.com/en-us/library/windows/desktop/ms682586%28v=vs.85%29.aspx>
- [6] http://en.wikipedia.org/wiki/Click_fraud

Appendix

P2P RC4 Key

The RC4 key used in all P2P communications is the MD5 of the fixed dword value: 0xCD6734FE.

RSA Public key

The RSA public key used to verify the signature on the downloaded files uses a 512 bit modulus, shown here. (By current cryptographic standards, this is considered weak.)

-----BEGIN PUBLIC KEY-----

MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBAM6rnSxDOEEP8safnkTPWes+fNxaJtBc
Mc8rAjpE3hgC0ZQBxCAb48WQ8UmH4UDnSTMK0rCaqgG7vzfktgQUVYsCAwEAAQ==

-----END PUBLIC KEY-----

Detection names used by Sophos Anti-Virus

1. Infected files will be detected and blocked as Mal/ZAccess-x, Troj/ZAccess-x, Mal/Sirefef-x or Troj/Sirefef-x , where x denotes an alphabetic suffix (-A, -B, etc.) On a properly-protected system, this should prevent infection in the first place.
2. Active processes will be reported and blocked by the Sophos run-time HIPS (Host Intrusion Detection System) as HPmal/ZAccess-A. This gives an extra layer of safety by providing proactive detection and prevention even of samples which evade detection in (1) above.
3. The Zero Access rootkit itself will be detected in kernel memory, and can be cleaned up, as Troj/ZAKmem-A. This means that the malware can be remediated even on systems where the rootkit is already active and stealthing.