# KRYPTOS LOGIC

4.6

AL 10

Friday, January 8, 2016

# ZeroAccess 3 Analysis

After the takedown attempt in December 2013, the current status of ZeroAccess (alias Sirefef) has been much disputed. Initially the botmaster had released a command to the remaining infrastructure which contained nothing but the message "White Flag", signaling they may had given up on the botnet; but between March 21st 2014 and July 2nd 2014, SecureWorks had reported the ZeroAccess infrastructure resumed distributing commands.

Ever since the takedown ZeroAccess has not attempted to infect new hosts, instead it simply uses the large number of remaining infections to perform tasks—until now, that is. On January 3rd R136a1 came across a previously unseen sample, which FireF0X confirmed to be a variant of ZeroAccess. It's uncertain how long this sample has been in the wild, but it certainly points towards the botnet being a little less dead than was previously claimed.

#### Dropper

.........

.......

00000000

0.0.0.0.0

\*\*\*\*\*

......

The first thing I noticed was the dropper has a resource file which contains a 61 KB PNG.



The image may look corrupted, which is because the developer decided to store the encrypted PE file in a valid PNG format. Rather than using stenography to hide the code in an existing image, the image *is* the code.

The dropper uses the native Ldr API to find and acquire the resource (LdrFindResource\_U, LdrAccessResource) followed by the GdiPlus API to convert the PNG to a bitmap (GdipCreateBitmapFromStream), get the size of the code (GdipGetImageWidth × GdipGetImageHeight × 4), then decrypts it by xor'ing it with the key 'wDT:'.

start p	start rec asar	
Battat-o Battat, en Processi Alliacat Battatat Nar 300 Nar 300 Nar 300 Nar 300 Nar 300 Nar 300 Nar 300 Nar 300 Nar 300	Hard ptr -486 upth- damid ptr winddrass. New LesSize- dwerd typ- dwerd ptr -50 dwerd ptr -10 dwerd ptr -1	- 645 1 gt - 212 1 gt - 225 1 gt - 255
Sub puth path path bea sur puth puth puth puth	esp, ABD ebs edi edi eax, [esp=540 ebs, ebx eak edi, ADDODAD	946atatry) Ione
nov para nov nov call test jy	Jespiniskingetar edi (espiniskingetar (espiniskingetar dettarfinderes dettarfinderes dettarfinderes test, tak Techanting	i, etx attentizej, enx angthj, etx arte_s
		The same rest in the same same same same same same same sam

Execution is then passed to some shellcode which is pointed to by the  $e_res$  field of the decrypted PE's header. The shellcode resides directly after the .reloc section but is not part of the PE and therefore will not be present if the executable is later dumped after it's loaded.

	T
🚺 🚄	
mov mov push mov add	<pre>eax, [esp+58h+decrypted_pe] [esp+58h+ProcedureAddress], ebx eax ecx, [eax+1Ch] ; IMAGE_DOS_HEADER-&gt;e_res ecx, eax</pre>
call mov test jz	ecx ; Shellcode entry point edi, eax edi, edi short loc_4011BA

A unique and interesting thing about this shellcode is the way in which it handles resolving strings. Due to the fact shellcode is designed to be executed at whatever address it's allocated, it can't have the address of the strings hardcoded in; instead, it has to find their absolute address somehow. In this shellcode the author places a relative call directly before the string which makes a relative call to a stub after it, the stub then pops the return address into the eax and returns to the shellcode. As a result when string\_ZwMapViewOfSection is called, the return address of call loc\_22E (the first byte of the string 'ZwMapViewOfSection') ends up in the eax register.



The shellcode itself seems pretty mundane: it simply sets a hardware breakpoint on ZwMapViewOfSection using ZwSetContextThread with flag set to CONTEXT\_DEBUG\_REGISTERS, then loads shellstyle.dll (a legitimate system DLL). What actually happens though is when the PE loader tries to map shellstyle into memory, the breakpoint is hit and the shellcode's exception handler is called. The exception handler then modifies the return address of ZwMapViewOfSection to point to a function which replaces the module with the one we saw decrypted earlier, as a result loading the malicious DLL in the place of shellstyle.dll.



Once the shellcode has replaced shellstyle.dll, execution is returned to the dropper which then uses LdrGetProcedureAddress to resolve and call the payload's entrypoint.

<pre>lea eax, [esp+58h+ProcedureAddress] pusb eax ; ProcedureAddress push ebp ; ProcedureOrdinal push edi ; ProcedureOrdinal push edi ; ModuleImageBase call ds:LdrGetProcedureAddress ; Resolve DLL entrypoin test eax, eax js short loc_4011B3 nov esi, ebp jnp short loc_4011BA loc_4011B3: ; D11In push edi call ds:LdrUnloadD11</pre>	it
<pre>lea eax, [esp+58h+ProcedureAddress] push eax ; ProcedureAddress push ebp ; ProcedureOrdinal push edi ; ProcedureOrdinal push edi ; ModuleImageBase call ds:LdrGetProcedureAddress ; Resolve DLL entrypoin test eax, eax js short loc_4011B3</pre>	nt.
push eax ; ProcedureAddress push ebp ; ProcedureUrdinal push ebx ; ProcedureName push edi ; ModuleImageBase call ds:LdrGetProcedureAddress ; Resolve DLL entrypoint test eax, eax js short loc_4011B3 mov esi, ebp jnp short loc_4011BA loc_4011B3: ; D11In push edi call ds:LdrUnloadD11	nt
push ebp ; ProcedureOrdinal push ebx ; ProcedureName push edi ; ModuleImageBase call ds:LdrGetProcedureAddress ; Resolve DLL entrypoint test eax, eax js short loc_4011B3 nov esi, ebp jnp short loc_4011BA loc_4011B3: ; D1LIn push edi call ds:LdrUnloadD11	nt:
push ebx ; ProcedureName push edi ; ModuleImageBase call ds:LdrGetProcedureAddress ; Resolve DLL entrypoint test eax, eax js short loc_4011B3 mov esi, ebp jnp short loc_4011BA loc_4011B3: ; D11In push edi call ds:LdrUnloadD11	nt
push edi ; ModuleImageBase call ds:LdrGetProcedureAddress ; Resolve DLL entrypoin test eax, eax js short loc_4011B3	nt
call ds:LdrGetProcedureAddress ; Resolve DLL entrypoin test eax, eax js short loc_4011B3	nt.
test eax, eax js short loc_4011B3	
js short loc_4011B3	
nov esi, ebp jnp short loc_4011BA uc_4011B3: ; D11La push edi call ds:LdrUnloadD11	
nov esi, ebp jnp short loc_4011BA push edi call ds:LdrUnloadD11	
nov esi, ebp jnp short loc_4011BA push edi call ds:LdrUnloadD11	
jnp short loc_4011BA loc_4011B3: ; D1110 push edi call ds:LdrUnloadD11	
push edi call ds:LdrUnloadD11	nageBase
call ds:LdrUnloadD11	1
Los h84484	
TUC_40TIBH: ; FreeType	
los osu focul@budlloostionSizel	
nuch any Espesimentificationsize	
push edx , rreesize	
nuch oax, [esp-oun-decrypted_pe]	
push BEEEEEEE ProceeeBandla	
rall dc:20FrapHirtualHanoru	
toct oci oci	
iz short loc h01109	
Jr	
	7
call [esp+S8h+ProcedureAddress] ; Call DLL entrupoint	

# **First Stage Payload**

The payload consists of a DLL file with an embedded Microsoft Cabinet file (.cab), which can be found by searching the binary for the string 'MSCF'. The cabinet contains 6 files: 3 for 32-bit and 3 for 64-bit.

File	Description		
s32	Boostrap list containing IPs and ports for 32-bit nodes		
s64	Boostrap list containing IPs and ports for 64-bit nodes		
k32	32-bit final stage payload		
k32	64-bit final stage payload		
132	Shellcode used to load k32		
132	Shellcode used to load k64		

Next the payload checks a couple of events to see if the system is already infected, if not it create the event \BaseNamedObjects\Restricted\{12E9D947-EDF5-4191-AADB-F51815F004D8}. A semi-unique identifier for the bot is created by MD5 hashing the volume creation time of \SystemRoot, then formatting it as a 32-bit GUID (just like with older versions of the bot).

This version of ZeroAccess stores itself in a folder named #., which is contained within the directory <code>%localappdata%\Google\Desktop\Install<BotGUID></code>. The folder name #. is special because it makes use of a difference in operation between the NativeAPI (ntdll) and Windows API (kernel32). Using native API functions a folder with a dot at the end of the name is entirely valid and

functional; however, under the Windows API it is not; the result is a usable folder that the malware can freely use to store and execute its components, but that cannot be accessed or modified by explorer or any application not using the Native API. For further protection, the folder is set as an NTFS Reparse Point which redirects to nowhere, preventing access by any means that is susceptible to NTFS re-parsing.



The payload extracts the bootstrap list (s32 or s64 depending on architecture) from the cabinet file, then decrypts and stores it inside the install directory in a file named @. The dropper is also moved to the same directory with the name GoogleUpdate.exe.

10 mil		1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
and a	and addrest state - Training	
-	Par, Editor, John ; Ser	Inc. 18085 and the
ino	short loc 12005400	neu est, effect a532
		ney eax, 110m
	**	
	The description	
	LEC TERESTER	
	and CR, dt	
	and the cl	
	any al ob	
	neu ecc. est	
	new [cop+par at], as	
	lea eas, [ebp+var 10]	
	pach eas	
	call Calextract	
	test eas, eas	
	jz 100_100055F5	
	100 (M)	
	POV 051, 8080	
	cub [sph.ast.36]*	e11
	jnz short loc_100	055.08
-		
	9	
mev	ecx, [ebp+lpHen]	
C&11	500 1020470E	
Lab	the February Feb	in la
100	adv. futpreter_set	tetholes'
aush	Par Estuff	
push	esi : FileRa	ndle
puch	[obp+lphen] ; Buffer	and the second sec
lea	ecs. [ebp+mjectattribut	es] ; Objectattributes
6911	writefile	and the second
000	[copewar_a], eax	
1		
	the loss	-
	Les and Sciences and a	all the second se
	new arr. fature lotten	
	call free	
	cno [copywar A], 4	
	12 10c 102055F5	
		_
<b>1</b>	2	
mov	ecx, offset absogleapdat	e_e_: "douglespeate.exe"
nev	edx, edi	AN ALL TRANSPORTED LINES
sub	eds, ecs.	

GoogleUpdate.exe is added to autostart by creating a key under HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce with the the bot's GUID as its name; the key name is prefixed with a null byte and written as Unicode, preventing Regedit from displaying it.



Throughout its installation process the bot sends status updates to a C&C via UDP on port 53 (DNS) with geolocation information obtained from the freegeoip.net api and an affiliate id stored in the bot (ZeroAccess uses an affiliate scheme in which people can sign up to be paid a small amount of money for each computer they infect with the malware). All status update communication is encrypted with the key  $0 \times 4C4F4E47$  ('LONG'), which is the same key as in previous versions.

```
Time
                                                                            Destination
                                                                                                                  Protocol Length Info
                                                                                                                                                         Dest Port
No.
                                      Source
      53 0.00000000 45.32.152.183
                                                                                                                  DNS
                                                                                                                                      73 60920 53
                                                                            8.8.8.8
 60920 0.020793000 8.8.8.8
                                                                       45.32.152.183
                                                                                                               DNS
                                                                                                                                    105 53
                                                                                                                                                         60920
      80 0.021276000 45.32.152.183 104.236.43.108
                                                                                                                 TCP
                                                                                                                               66 49205 80
 49205 0.117626000 104.236.43.108
                                                                           45.32.152.183
                                                                                                                   TCP
                                                                                                                                      66 80
                                                                                                                                                         49205
       80 0.117687000 45.32.152.183
                                                                        104.236.43.108
                                                                                                                  TCP
                                                                                                                                      54 49205 80
                                                                           104.236.43.108
                                                                                                                  HTTP 117 49205 80
        80 0.117819000 45.32.152.183
  49205 0.213822000 104.236.43.108
                                                                             45.32.152.183
                                                                                                                   TCP
                                                                                                                                       60 80
                                                                                                                                                         49205
                                                                                                                                    654 80
                                                                                                                   HTTP
                                                                                                                                     60 80
                                                                                                                                                        49205
 49205 0.341505000 104.236.43.108 45.32.152.183
                                                                                                               TCP
       80 0.341552000 45.32.152.183 104.236.43.108
80 0.341678000 45.32.152.183 104.236.43.108
                                                                                                                   TCP
                                                                                                                                       54 49205 80
                                                                                                                  TCP
                                                                                                                                      54 49205 80
 49205 0.437688000 104.236.43.108 45.32.152.183
                                                                                                                 TCP 60 80
                                                                                                                                                         49205
       53 0.438592000 45.32.152.183
53 0.445425000 45.32.152.183
                                                                            5.45.70.105
                                                                                                                   DN5
                                                                                                                                       62 60921 53
                                                                           5.45.70.105
                                                                                                                                     62 60922 53
                                                                                                              DNS
       53 0.448652000 45.32.152.183 5.45.70.105 DNS 62 60923 53
53 0.448789000 45.32.152.183 5.45.70.105 DNS 62 60924 53
       53 0.450571000 45.32.152.183 5.45.70.105 DNS 62 60925 53
      53 0.450706000 45.32.152.183 5.45.70.105
                                                                                                                  DNS
                                                                                                                                     62 60926 53
       53 0.454334000 45.32.152.183 5.45.70.105
53 0.456726000 45.32.152.183 5.45.70.105
                                                                                                                   DNS
                                                                                                                                       62 60927
                                                                                                                                                         53
                                                                                                                                      62 60928 53
                                                                                                                  DNS
       53 0.456927000 45.32.152.183
                                                                           5.45.70.105
                                                                                                                   DNS
                                                                                                                                       62 60929 53
 Hypertext Transfer Protocol
    HTTP/1.0 200 OK\r\n
        Access-Control-Allow-Credentials: true\r\n
        Access-Control-Allow-Methods: GET, HEAD, OPTIONS\r\n
        Access-Control-Allow-Origin: *\r\n
        Content-Type: application/json\r\n
        X-Database-Date: wed, 02 Dec 2015 07:06:47 GMT\r\n
        X-Ratelimit-Limit: 10000\r\n
        X-Ratelimit-Remaining: 9999\r\n
        X-Ratelimit-Reset: 3600\r\n
        Date: Mon, 04 Jan 2016 05:10:46 GMT\r\n

    Content-Length: 241\r\n

         \r\n
         [HTTP response 1/1]
         [Time since request: 0.223684000 seconds]
         [Request in frame: 6]
 JavaScript Object Notation: application/json
    ⊟ Object
        ⊛ Member кеу: "ip"

    Member Key: "country_code"

    Member Key: "country_name"
    Member Key: "country_name"
    Additional Control Contro Control Control Control Contro Control Control Control Contro

    Member Key: "region_code"

    Member Key: "region_name"

    Member Key: "zip_code"

    Member Key: "time_zone"

    Member Key: "latitude"

    Member Key: "longitude"

    Member Key: "metro_code"
```

Finally the main payloads 132 and k32 or 164 and k64 are extracted from the cabinet (my hypothesis is 'I' stands for 'Loader' and k stands for 'Kernel'). The ZeroAccess kernel is a DLL containing the main code for communicating with the peer-to-peer network and the loader is used to load the DLL in the same way the dropper loaded shellstyle.dll, except this time the DLL loaded and replaced is comres.dll.

If the process is UAC elevated it injects into svchost.exe, otherwise it will attempt to inject into explorer and bypass UAC using the same method as previous version (detailed in FireF0X's UACMe), before finally injecting into svchost.exe.



The injection methods works by writing the kernel, followed by the loader, to the remote process using <code>ZwAllocateVirtualMemory</code> and <code>ZwWriteVirtualMemory</code>, then executing the loader by calling <code>ZwQueueApcThread</code>. In the case of 64-bit, the payload will use Heaven's Gate to switch into 64-bit mode before calling the injection routine, allowing it to inject 64-bit processes from WOW64.



## **Peer to Peer Communications**

The new ZeroAccess variant largely preserves the communication protocol of its previous versions, with some notable changes. For completeness, the UDP packet format for C&C communication has a common header of the form

```
struct zeroaccess_packet {
    uint32_t crc32;
    uint32_t type;
    uint32_t id;
    uint32_t length : 16;
    uint32_t reply : 1;
    uint32_t packet_number : 15;
};
```

The packet\_number field appears to be new. It is used to ensure incoming packets match earlier requests, possibly to prevent 'retL' flooding by takedown attempts. The type field can be one of 'getL' and 'retL' —the 'newL' field of some earlier versions is now gone. id is a 32-bit bot session id, generated at startup and used to identify sessions. The reply field indicates whether the receiving node should reply with a getL of its own.

Every packet is encrypted—as before—with a very simple rolling-xor algorithm:

```
void zeroaccess_encrypt(unsigned char * message, size_t bytes) {
    uint32_t k = 0x31323334UL;
    assert(bytes % 4 == 0);
    for(size_t i = 0; i < bytes; i += 4) {
        store_le32(m + i, load_le32(m + i) ^ k);
        k = (k << 1) | (k >> 31);
    }
}
```

Note the new key, '1234', instead of the old one 'ftp2'. This same algorithm is used with the 'LONG' key mentioned above.

The getL packet is empty and consists only of the above header. The retL packet format is slightly changed due to one of the most noteworthy changes in the protocol—*UDP ports are now randomly-generated*, and thus the retL packets must include port information. The retL packet is of the form

```
struct peer_entry {
    uint32_t ip;
    uint32_t port : 14;
    uint32_t timestamp : 18;
};
struct file_entry {
    uint32_t id;
    uint32_t length;
    uint32_t length;
    uint8_t signature[128];
};
struct retL {
    peer_entry peers[16];
    file_entry files[/*variable*/];
};
```

On 32-bit Windows hosts, the port number is computed as peers[i].port + 16384, whereas on 64-bit Windows, it is computed as peers[i].port + 32768. It is therefore simple to figure out an individual node's platform by its UDP port number. The timestamp is computed as (GetTime() - 0x41000000) / 3600, where GetTime() is

```
ULONG GetTime() {
  FILETIME time;
  ULONG stamp;
  GetSystemTimeAsFileTime(&time);
  RtlTimeToSecondsSince1980((LARGE_INTEGER *)&time, &stamp);
```

```
return stamp;
}
```

We can see that the base timestamp  $0 \times 41000000$ , i.e., July 22 2014, lower-bounds the age of this variant. The file list format remains unchanged (modulo timestamp format), albeit with a distinct RSA-1024 verification key compared to the previous botnets.

The bot must also remember what its port number is. This information is stored in the Extended Attributes of the @ file. The list of known peers is stored in this file, as remarked above.

### **Statistics**

We have some early crawling numbers for this variant. But first, is may be useful to compare with the previous ZeroAccess 2 botnet. We see for ZeroAccess 2 daily distribution seemed rather consistent. The total confirmed connections were approximately 715,000 for the month.

					2 · · · ·
the second se					
					74810
				-	
			1.4.1		
(mm)			-		. *
a production management		control of our local states	1000	reported process into the	49.1 × (4.24)
A MARY REAL PROPERTY AND ADDRESS OF	here -	summer and the state of the state			
<ol> <li>Address of the Application Company</li> </ol>	Tab.	second 12, 100, 1770 Month	-		
A DESCRIPTION OF A DESC		register of parts in the second		Access Page 11	10.100 (m. 10)
A rest of the other ways and the	1000	control of the local strength and the			11 F 12 M
a second addition, which is not whether it	The second se	conservation and a second second second	100 C	1- feet	all strain and
A reason of the local of the second second	Contract in the second s	connect by some diverse with	and the second s	TOP PARTY IN	10 111 10 St
A Desire of the second se		control of the later, or control of		way water later	-10.0 × 2.0
a contract of the provide states of the		-many parties and beautions	iner i	Autor and A	
A COMPACT OF STATE AND ADDRESS OF TAXABLE PARTY.	state.	second of the distances			10-11
A DESCRIPTION OF A DESC	whet patronages	"respect to the state of the	and a second sec	single and second se	1.41-10.41
A DESCRIPTION OF A DESC	alterna manage	CONTRACTOR OF A DESCRIPTION OF A DESCRIP		where our second him	P1.500.00.011
A long to per second second	(0.0. April )	manufacture and provide the state	6471.		1.000.000 M
A second se	Contra a	second in the August State		The West	the data was not as
increased and state down in the second second		Gauget reported in the state of the	1007T -	100	1.00.00
A state of the second sec	-min*1	CONTRACTOR AND ADDRESS OF ADDRESS OF ADDRESS ADDRE		Without B common 11.	10.000 P
a conversion of the state in the income of		control for land, prost, from the		of star	all for a low-
1. And in case of the local division of the	in contain	tioned his state management	and a second	with the Million	10 March 10
A subscript pro- built with the subscript of	inter-	simult haven, area word		Task time	1.00-00.00
·		opposite, and press, in concentrated			10.0111.00
A DESCRIPTION OF A DESC	_	control of them, property with			
1 manual and stated as the state of the stat	Caracter 1	manufacture and an include	and a	The second se	17. m. 3m 37
I WARD TRADE IN THE OWNER IN		transmitten and an and an and			0.0710.000
A summer of states, prior in fact, summaries in	Augusture -	comment and lines, prime showing		which an angles was suit.	and 100 per cont
·	-	ments for loss and the set		terminal and terminal	
A Design of the same is not become a		second by part with \$1.000	100	and strength	
1 mart is in a market when the	which prove-	many, he sees should be been		train and A	
a many located provide in the second	and the second	common des leves, entitation leves			1.0.000
A REAL PROPERTY AND INCOME.	Sec.	second on the local bands		and the second termination for the	and the second second
I served address to be in the server of	Tanan.	second in the second second second		application and a Characterizati	177 0.00
A DESCRIPTION OF A DESCRIPTION OF		cannot be and been set			W100.010
A state of the sta	and and	many in our sum the			100 CT

Looking deeper into a sub aggregation of country and unique IPs counted we can see a high level in US on Comcast, although this could be explained by IP churning due to DHCP for the cable provider. Nonetheless, this is a high number of uniques compared to any other ISP.



Finally, we see a rather healthy distribution of nodes and countries with a heavy weight in Japan and US. Additionally, we observe an unusual high amount of unique IPs the day after Christmas.



In comparison, the latest Zero Access 3 variant, we see the IP distribution is primarily focused in Russia. Whether this team is preparing the infrastructure for a larger planned botnet or it is focusing its efforts in the area is still unknown.



We can potentially hypothesize that this botnet has been retooled to allow botnet subgroups using the generation of new encryption keys and RSA binary signing keys. The result of this would mean multiple actors could use the same malware, similarly to trends that have been observed with other seed-based malware, like Tiny Banker (aka Tinba). This would make takedowns more difficult in that there is no one central peer to peer network, but rather a cluster of peer to peer networks. Another plausible hypothesis is that the ZeroAccess code base has been sold (or stolen) to a third party, one who is more interested in targeting Eastern Europe.

These types of networks are becoming more and more resilient to takedowns, as observed with Dridex, and even ZeroAccess itself is still maintaining its presence. With multiple authors and actors involved, the difficulty of takedown attempts by technical means is also compounded by the difficulty of prosecuting multiple authors (and actors) in multiple geopolitically diverse jurisdictions.

We would like to thank MalwareTech for his invaluable contributions to this research.