What do Win32/Redyms and TDL4 have in common?

By Aleksandr Matrosov

Archived: 2026-04-05 18:15:48 UTC

Malware

At the beginning of January 2013, we started tracking the interesting Win32/Redyms trojan family. Redyms is notable for changing search results from popular search engines on infected machines.

04 Feb 2013 • , 2 min. read

At the beginning of January 2013, we started tracking the interesting Win32/Redyms trojan family. Redyms is notable for changing search results from popular search engines on infected machines. This Trojan family is mostly spreading in the United States and Canada. It is in these regions that the cybercrime market pays the highest prices for redirection from popular search engines results to malicious ads/links. I've already published information about PPI (Pay-Per-Install) programs which distribute malware for BlackHat SEO (Cycbot: Ready to Ride).

Deeper analysis shows that Win32/Redyms has similar code functions and work activity to Win32/Agent.TJO (also known as part of the Olmarik/TDL4 family). Win32/Agent.TJO is a user-mode Trojan program, but based on clicker component functionality found in the TDL4 family (The Evolution of TDL: Conquering x64). TDL4, Win32/Agent.TJO and Win32/Redyms use similar techniques for network traffic manipulation in the active internet browser. So as to intercept and alter the data exchanged over the network the bot hooks several functions from Microsoft Windows Socket Provider (mswsock.dll):

```
WSPSend proc near
jmp     near ptr 1548D0h ; jmp to hooked WSPSend
WSPSend endp
```

```
WSPRecv proc near
jmp     near ptr 154E90h ; jmp to hooked WSPRecv
WSPRecv endp
```

```
WSPCloseSocket proc near
jmp     near ptr 1553D0h ; jmp to hooked WSPCloseSocket
WSPCloseSocket endp
```

All three malware families use the same hooking technique and a communication protocol encrypted by RC4 stream cipher. The communication routine call graph in Win32/Redyms looks like this:

The next interesting finding in Win32/Redyms is the DGA (Domain Generation Algorithm) used to generate C&C domain names. This DGA is based on simple alphabetical permutations and changes according to the init state constant. Reconstructed algorithms in python and decompiled assembly code are shown in the following figures:

```python
def dga_gen(domain_count, cc_const):
    tmp1 = cc_const * (domain_count + 5)
    tmp2 = (tmp1 % 7) + 9
    tmp3 = tmp1 + (tmp2 ^ 0x5f1)
    size = (tmp3 & 3) + 5


    data = [0 for i in xrange(tmp2 + size + 9)]
    for i in xrange(tmp2):
        if i == size:
            data[i] = 77
        else:
            data[i] = (tmp3 % 0x1A + 32) ^ 0xA1
        tmp3 = tmp3 + (i ^ 0x5B)

    data[tmp2 + 0] = 0x8f
    data[tmp2 + 1] = 0xcf
    data[tmp2 + 2] = 0xc4
    data[tmp2 + 3] = 0xd5
    data = [data[i]^0xA1 for i in xrange(tmp2 + 4)]
    for i in xrange(tmp2):
        if data[i] == 0x10:
            data[i] = 45
        else:
            data[i] = (data[i] + 65) & 0xFF

    domain = ""
    for i in xrange(len(data)):
        domain += "%c" % data[i]
    return domain
```

[python code]

```
tmp1 = cc_const * (domain_count + 5);
tmp2 = tmp1 % 7 + 9;
tmp3 = tmp1 + (tmp2 ^ 0x5F1);
size = (tmp3 & 3) + 5;
heap = GetProcessHeap();
data = (CHAR *)HeapAlloc(heap, 8u, size + tmp2 + 9);
if ( !data )
  goto END;
memset(data, 0, size + tmp2 + 9);
ctr_1 = 0;
if ( tmp2 )
{
  do
  {
    if ( ctr_1 == size )
      data[ctr_1] = 0x4D;
    else
      data[ctr_1] = (tmp3 % 0x1A + 32) ^ 0xA1;
    tmp3 += ctr_1++ ^ 0x5B;
  }
  while ( ctr_1 < tmp2 );
}
ctr_2 = 0;
*(_DWORD *)&data[tmp2] = 0xD5C4CF8F;
if ( tmp2 != 0xFFFFFFFC )
{
  do
  {
    data[ctr_2] ^= 0xA1u;
    ++ctr_2;
  }
  while ( ctr_2 < tmp2 + 4 );
}
ctr_3 = 0;
if ( tmp2 )
{
  do
  {
    byte_ch = data[ctr_3];
    if ( byte_ch == 0x10 )
      data[ctr_3] = 0x2D;
    else
      data[ctr_3] = byte_ch + 0x41;
    ++ctr_3;
  }
  while ( ctr_3 < tmp2 );
}
```

[decompiled assembly code]

The list of C&C domain names generated looks like this:

```
http://erdoypf-inr.net/erdoypf.php
http://reqblcsh-a.net/reqblcsh.php
http://anzku-bqe.net/anzku.php
http://sfrcmdt-wbfiktb.net/sfrcmdt.php
http://fsepzqgv-osvxg.net/fsepzqgv.php
http://obnyi-pesxbeg.net/obnyi.php
http://boalvm-rfkor.net/boalvm.php
http://kxjuevl-otx.net/kxjuevl.php
http://xkwhriyn-g.net/xkwhriyn.php
http://gtfqa-hwk.net/gtfqa.php
http://ylxisjz-chloqzh.net/ylxisjz.php
http://lykvfwmb-uybdm.net/lykvfwmb.php
http://uhteo-vkydhkm.net/uhteo.php
http://hugrbs-xlqux.net/hugrbs.php
http://qdpakbr-uzd.net/qdpakbr.php
http://dqcnxoet-m.net/dqcnxoet.php
http://mzlwg-ncq.net/mzlwg.php
http://erdoypf-inruwfn.net/erdoypf.php
http://reqblcsh-aehjs.net/reqblcsh.php
http://anzku-bqejnqs.net/anzku.php
http://namxhy-drwad.net/namxhy.php
http://wjvgqhx-afj.net/wjvgqhx.php
http://jwitdukz-s.net/jwitdukz.php
http://sfrcm-tiw.net/sfrcm.php
http://kxjuevl-otxaclt.net/kxjuevl.php
http://xkwhriyn-gknpy.net/xkwhriyn.php
http://gtfqa-hwkptwy.net/gtfqa.php
http://tgsdne-jxcgj.net/tgsdne.php
http://cpbmwnd-glp.net/cpbmwnd.php
```

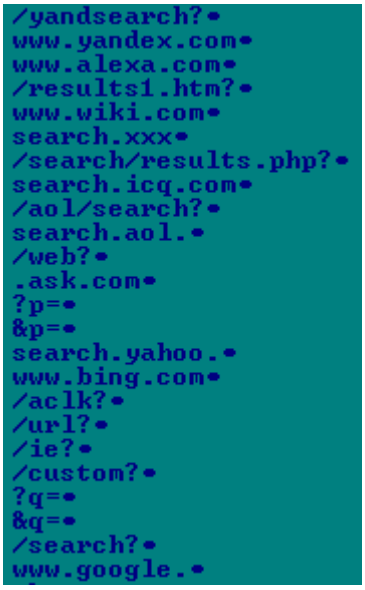
The first domain names from the list were registered in the middle of December 2012 or at the beginning of January 2013. This indicates indirectly that Win32/Redyms was being distributed at the end of December.

Win32/Redyms injects malicious code into all active processes on the infected machine. If activity from the most popular browsers was detected, a malicious thread was injected into the browser process and functions from mswsock.dll were hooked.

```
char __stdcall inject_thread_to_browser(int a1)
{
  HANDLE v1; // edi@2
  int v2; // eax@4

  GetCurrentProcess();
  if ( is_wow64_process() )
  {
    v1 = OpenProcess(PROCESS_QUERY_INFORMATION, 0, *(a1 + 8));
    if ( v1 )
    {
      is_wow64_process();
      CloseHandle(v1);
    }
  }
  LOBYTE(v2) = is_browser_process((a1 + 36));
  if ( v2 )
  {
    v2 = open_process_mutex(*(a1 + 8));
    if ( !v2 )
      LOBYTE(v2) = inject_thread(*(a1 + 8), 0);
  }
  return v2;
}
```

The injected code intercepts network activity from the browser process and looks up search engines from the following list:



If search engine activity is detected, all search requests are redirected to the C&C and the URLs shown in the search results will be changed in accordance with the list received from the C&C. The URL checking engine is based on hooking the WSPSend() routine, which uses Adelson-Velsky/Landis (AVL) trees as data structure in order to manage the data. For operating with the AVL trees data structure the malware utilizes the structure RTL_GENERIC_TABLE from kernel32.dll. TDL4 uses the same ideas in the user-mode component cmd.dll.

Aleksandr Matrosov, Security Intelligence Team Lead

SHA1 hashes for analyzed samples:

```
 Win32/Redyms.AB: 07e73ac58bee7bdc26d289bb2697d2588a6b7e64
Win32/Agent.TJO: 3777c3e98e5be549a7c73f6841c759a9f8a098c3
```

Let us keep you up to date

Sign up for our newsletters

Source: https://www.welivesecurity.com/2013/02/04/what-do-win32redyms-and-tdl4-have-in-common/