New ZeuS.Maple Variant Targets Canadian Online Banking Customers

securityintelligence.com/zeus-maple-variant-targets-canadian-online-banking-customers/

June 9, 2014

Home&nbsp/ Banking & Finance

ZeuS.Maple Variant Targets Canadian Online Banking Customers

Banking & Finance June 9, 2014

By Dana Tamir 5 min read

Ever since the ZeuS cyber crime toolkit source code leaked in 2011, malware authors have used its cogent malware development tools for generating new custom versions of the Trojan; examples include the ICE-IX and Citadel variants. Trusteer security research team identified a series of attacks carried out by a new ZeuS variant since January 2014. Seeing that this variant mainly targets customers of Canadian banks, IBM Trusteer security research team has named it “ZeuS.Maple.”

Trusteer researcher Avidan Avraham, who conducted a thorough analysis on the new variant, explains that ZeuS.Maple is a heavily modified version of ZeuS 2.0.8.9. It implements unique browser re-patching techniques (browser patching is a method of stealing information from browser sessions; re-patching ensures the patch stays in place), an alternative naming generation algorithm, different anti-debugging and new anti-VM capabilities. It uses an encrypted configuration stored in the Windows registry, and in order to remain stealthy, ZeuS.Maple distribution in the wild is limited and controlled.

Avraham adds that the enhancements introduced in ZeuS.Maple are improvements of known ZeuS capabilities, but they don’t really add new functionality. This is why it is interesting that the malware author designated this variant as ZeuS version 3.3.6.0 (as seen in the configuration).

Dissimulating the Executable in a New Installation Path

Most of the ZeuS-based Trojans generate a randomly named executable file and place it in a newly created folder under a randomly generated name; this makes it difficult to detect the file in the file system. ZeuS.Maple takes a different approach for naming the newly generated file: First it enumerates the %APPDATA% directory and chooses an existing folder for its dropped executable location. It then generates a file name from the combination of the directory name and a hard-coded string (a few string options exist). The new executable file is then dropped in the selected directory.

For example:

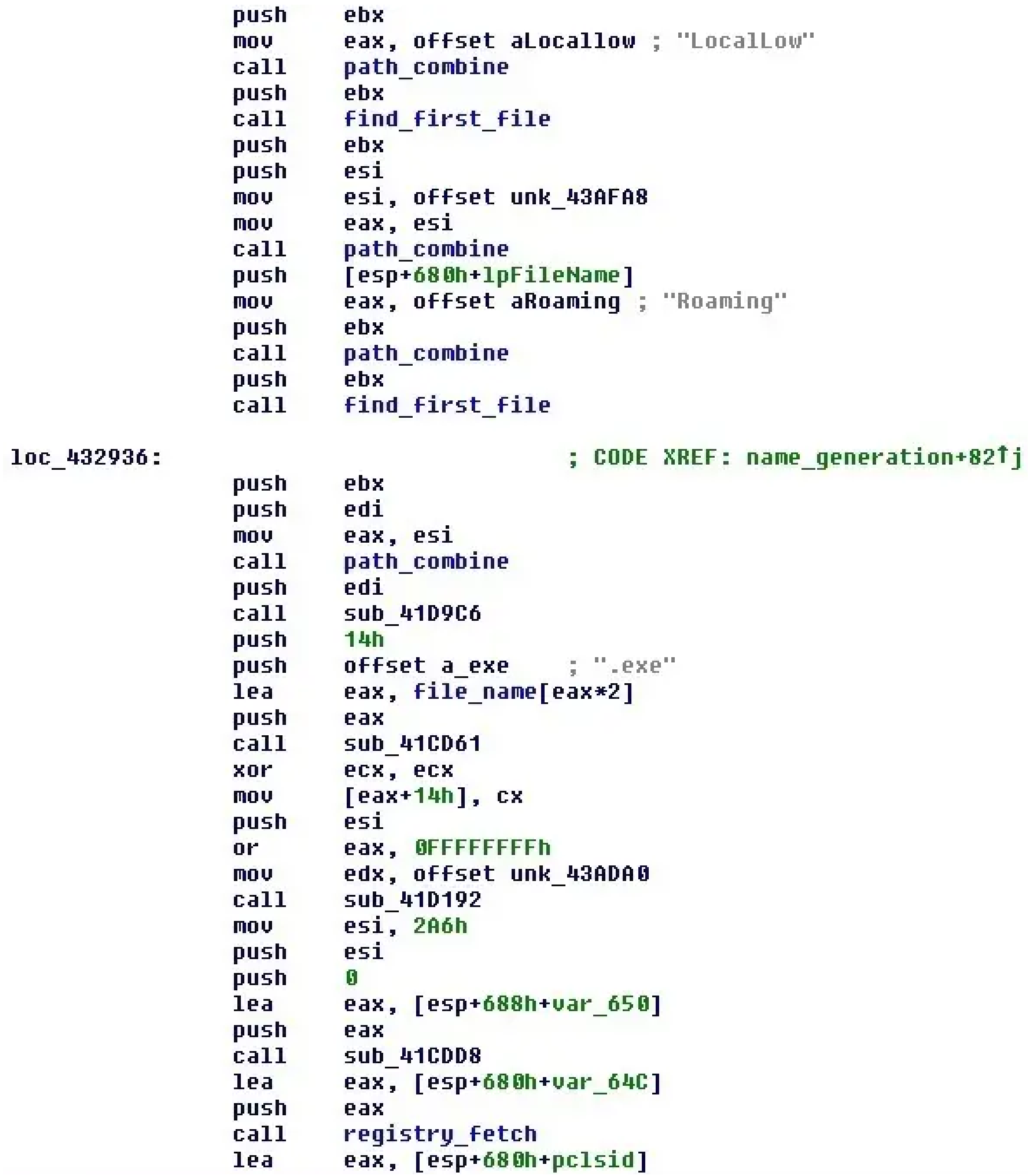
If the selected directory is *c:\users\user\appdata\roaming\microsoft\*

And the hard-coded string is: ‘win’

The result will be: *c:\users\user\appdata\roaming\microsoft\winmicrosoft.exe*

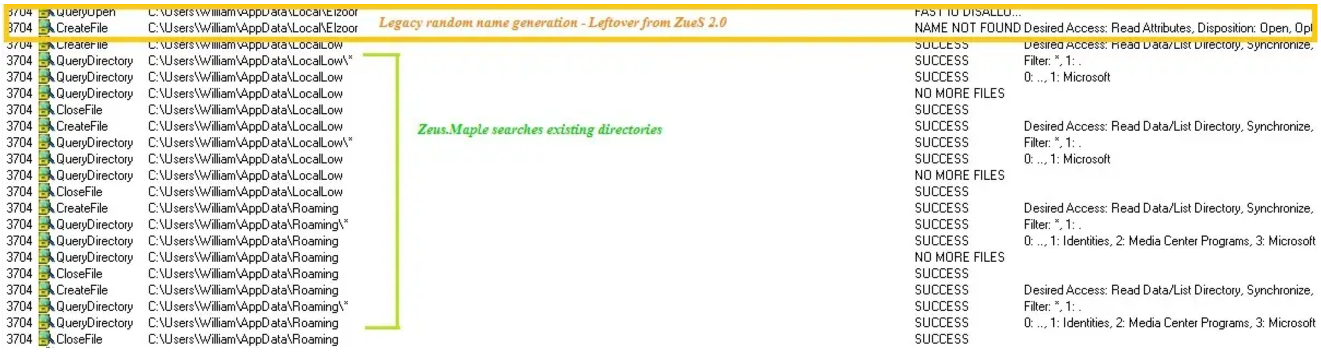
This technique of dissimulating the malicious executable within existing system paths makes the file look legitimate and enables it to stay stealthy.

The code used for the dissimulation is shown in Figure 1:



```
                push    ebx
                mov     eax, offset aLocallow ; "LocalLow"
                call    path_combine
                push    ebx
                call    find_first_file
                push    ebx
                push    esi
                mov     esi, offset unk_43AFA8
                mov     eax, esi
                call    path_combine
                push    [esp+680h+lpFileName]
                mov     eax, offset aRoaming ; "Roaming"
                push    ebx
                call    path_combine
                push    ebx
                call    find_first_file

loc_432936:                             ; CODE XREF: name_generation+82↑j
                push    ebx
                push    edi
                mov     eax, esi
                call    path_combine
                push    edi
                call    sub_41D9C6
                push    14h
                push    offset a_exe    ; ".exe"
                lea     eax, file_name[eax*2]
                push    eax
                call    sub_41CD61
                xor     ecx, ecx
                mov     [eax+14h], cx
                push    esi
                or      eax, 0FFFFFFFFh
                mov     edx, offset unk_43ADA0
                call    sub_41D192
                mov     esi, 2A6h
                push    esi
                push    0
                lea     eax, [esp+688h+var_650]
                push    eax
                call    sub_41CDD8
                lea     eax, [esp+680h+var_64C]
                push    eax
                call    registry_fetch
                lea     eax, [esp+680h+pclsid]
```

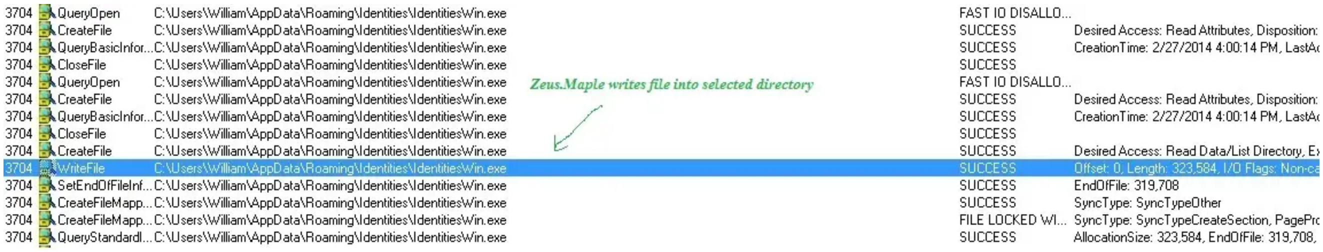
An additional piece of code found in ZeuS.Maple generates an ordinary ZeuS file name using Windows’ GetTickCount (a Windows function used by ZeuS to generate a random file name); however, it doesn’t write it to disk. It could be a leftover action from ZeuS source code.



Legacy random name generation - Leftover from ZueS 2.0

Zeus.Maple searches existing directories

```
3704 QueryOpen      C:\Users\William\AppData\Local\Elzoor     FAST IO DISALLO...
3704 CreateFile     C:\Users\William\AppData\Local\Elzoor     NAME NOT FOUND Desired Access: Read Attributes, Disposition: Open, Opl
3704 CreateFile     C:\Users\William\AppData\LocalLow         SUCCESS        Desired Access: Read Data/List Directory, Synchronize,
3704 QueryDirectory C:\Users\William\AppData\LocalLow\*       SUCCESS        Filter: *, 1: .
3704 QueryDirectory C:\Users\William\AppData\LocalLow         SUCCESS        0: .., 1: Microsoft
3704 QueryDirectory C:\Users\William\AppData\LocalLow         NO MORE FILES
3704 CloseFile      C:\Users\William\AppData\LocalLow         SUCCESS
3704 CreateFile     C:\Users\William\AppData\LocalLow         SUCCESS        Desired Access: Read Data/List Directory, Synchronize,
3704 QueryDirectory C:\Users\William\AppData\LocalLow\*       SUCCESS        Filter: *, 1: .
3704 QueryDirectory C:\Users\William\AppData\LocalLow         SUCCESS        0: .., 1: Microsoft
3704 QueryDirectory C:\Users\William\AppData\LocalLow         NO MORE FILES
3704 CloseFile      C:\Users\William\AppData\LocalLow         SUCCESS
3704 CreateFile     C:\Users\William\AppData\Roaming          SUCCESS        Desired Access: Read Data/List Directory, Synchronize,
3704 QueryDirectory C:\Users\William\AppData\Roaming\*        SUCCESS        Filter: *, 1: .
3704 QueryDirectory C:\Users\William\AppData\Roaming          SUCCESS        0: .., 1: Identities, 2: Media Center Programs, 3: Microsoft
3704 QueryDirectory C:\Users\William\AppData\Roaming          NO MORE FILES
3704 CloseFile      C:\Users\William\AppData\Roaming          SUCCESS
3704 CreateFile     C:\Users\William\AppData\Roaming          SUCCESS        Desired Access: Read Data/List Directory, Synchronize,
3704 QueryDirectory C:\Users\William\AppData\Roaming\*        SUCCESS        Filter: *, 1: .
3704 QueryDirectory C:\Users\William\AppData\Roaming          SUCCESS        0: .., 1: Identities, 2: Media Center Programs, 3: Microsoft
3704 CloseFile      C:\Users\William\AppData\Roaming          SUCCESS
```



Zeus.Maple writes file into selected directory

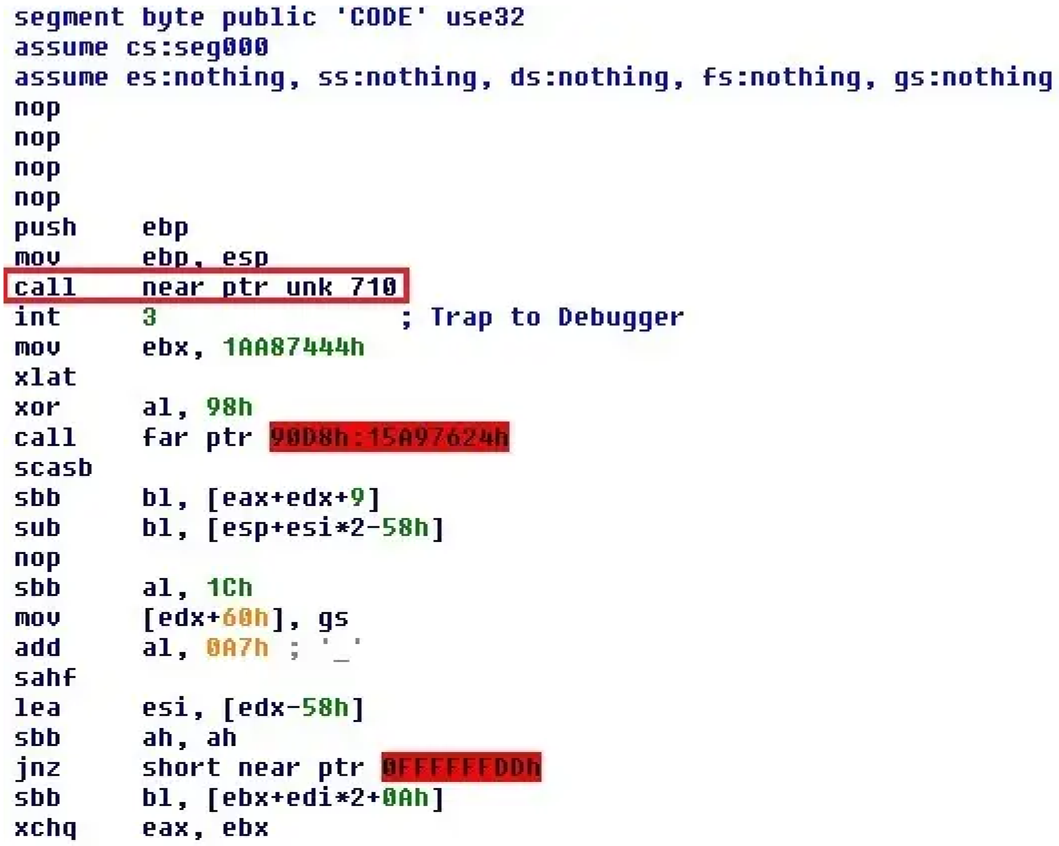
```
3704 QueryOpen        C:\Users\William\AppData\Roaming\Identities\IdentitiesWin.exe   FAST IO DISALLO...
3704 CreateFile       C:\Users\William\AppData\Roaming\Identities\IdentitiesWin.exe   SUCCESS          Desired Access: Read Attributes, Disposition:
3704 QueryBasicInfor... C:\Users\William\AppData\Roaming\Identities\IdentitiesWin.exe SUCCESS          CreationTime: 2/27/2014 4:00:14 PM, LastAc
3704 CloseFile        C:\Users\William\AppData\Roaming\Identities\IdentitiesWin.exe   SUCCESS
3704 QueryOpen        C:\Users\William\AppData\Roaming\Identities\IdentitiesWin.exe   FAST IO DISALLO...
3704 CreateFile       C:\Users\William\AppData\Roaming\Identities\IdentitiesWin.exe   SUCCESS          Desired Access: Read Attributes, Disposition:
3704 QueryBasicInfor... C:\Users\William\AppData\Roaming\Identities\IdentitiesWin.exe SUCCESS          CreationTime: 2/27/2014 4:00:14 PM, LastAc
3704 CloseFile        C:\Users\William\AppData\Roaming\Identities\IdentitiesWin.exe   SUCCESS
3704 CreateFile       C:\Users\William\AppData\Roaming\Identities\IdentitiesWin.exe   SUCCESS          Desired Access: Read Data/List Directory, Ex
3704 WriteFile        C:\Users\William\AppData\Roaming\Identities\IdentitiesWin.exe   SUCCESS          Offset: 0, Length: 323,584, I/O Flags: Non-ca
3704 SetEndOfFileInf... C:\Users\William\AppData\Roaming\Identities\IdentitiesWin.exe SUCCESS          EndOfFile: 319,708
3704 CreateFileMapp... C:\Users\William\AppData\Roaming\Identities\IdentitiesWin.exe  SUCCESS          SyncType: SyncTypeOther
3704 CreateFileMapp... C:\Users\William\AppData\Roaming\Identities\IdentitiesWin.exe  FILE LOCKED WI... SyncType: SyncTypeCreateSection, PagePrc
3704 QueryStandardI... C:\Users\William\AppData\Roaming\Identities\IdentitiesWin.exe  SUCCESS          AllocationSize: 323,584, EndOfFile: 319,708,
```

Barriers for Malware Researchers: Anti-VM, Anti-Debugging

Malware researchers will often try to run the malware in a synthetic environment and debug it to understand how it operates. ZeuS 2.0 variants are already designed with anti-debugging features that make the malware analysis more difficult. In most cases, the variants use well-known packers that can be easily identified with common tools. ZeuS.Maple uses a unique packer that is written in Visual Basic, which is notoriously complex to debug and makes the analysis more difficult.

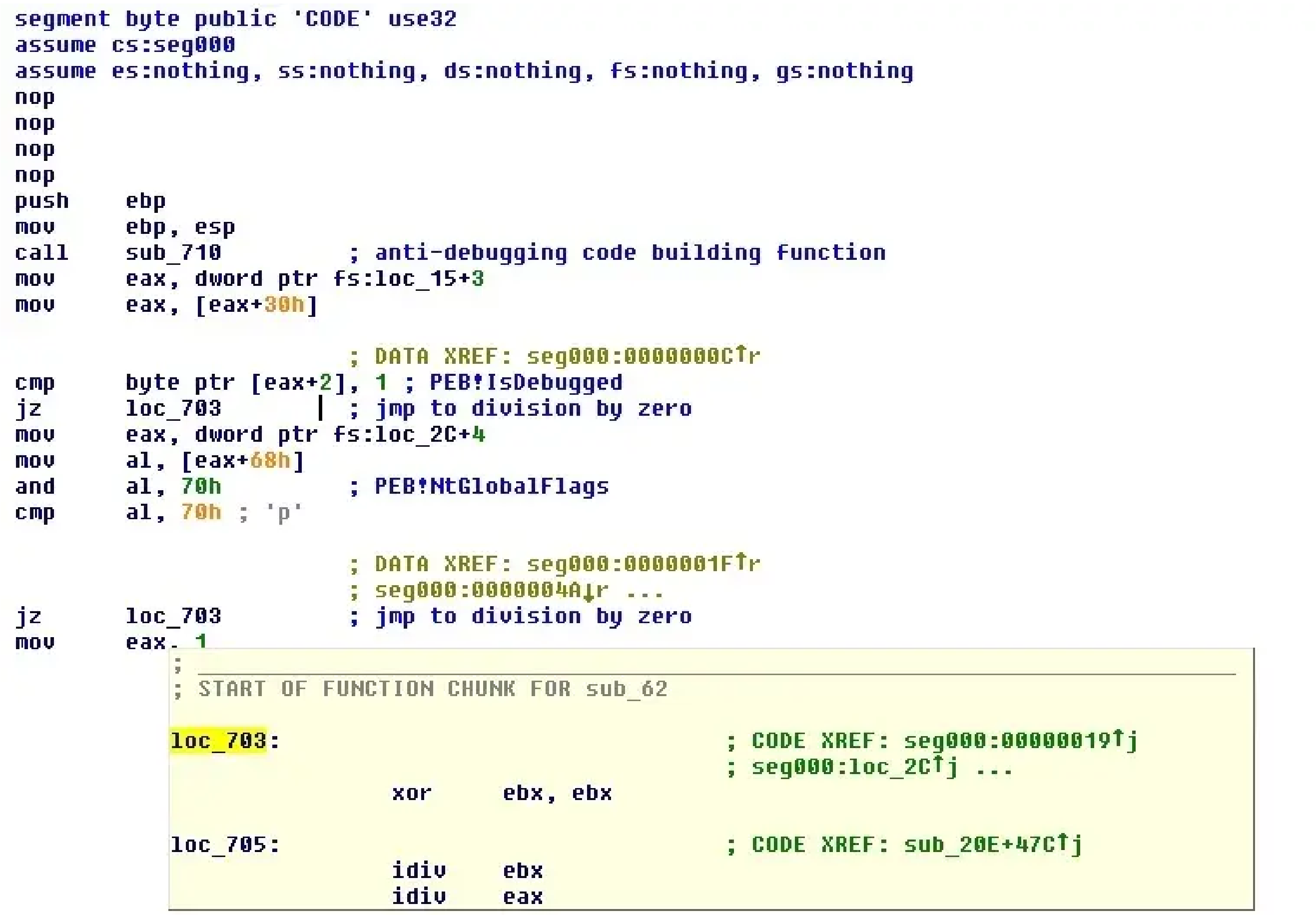
In addition, to prevent malware researchers from debugging the malware, ZeuS.Maple checks the value of two known Windows flags: PEB!IsDebuggedFlag and PEB!NtGlobalFlags. The code section that checks the flag value seems to be absent at first glance, but ZeuS.Maple unpacks this code section right before it uses it. In order to enable debug mode, we had to manipulate the flag value checks during runtime.

The screenshot below shows the obfuscated code prior to the unpacking function at unk_710:



```
segment byte public 'CODE' use32
assume cs:seg000
assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
nop
nop
nop
nop
push    ebp
mov     ebp, esp
call    near ptr unk_710
int     3               ; Trap to Debugger
mov     ebx, 1AA87444h
xlat
xor     al, 98h
call    far ptr 90D8h:15A97624h
scasb
sbb     bl, [eax+edx+9]
sub     bl, [esp+esi*2-58h]
nop
sbb     al, 1Ch
mov     [edx+60h], gs
add     al, 0A7h ; '_'
sahf
lea     esi, [edx-58h]
sbb     ah, ah
jnz     short near ptr 0FFFFFFDDh
sbb     bl, [ebx+edi*2+0Ah]
xchg    eax, ebx
```

After the call at unk_710 is completed, the code is readable and executable — see below. It is clear that this code section looks for flags inside the PEB and raises an exception if the process is being debugged.



```
segment byte public 'CODE' use32
assume cs:seg000
assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
nop
nop
nop
nop
push    ebp
mov     ebp, esp
call    sub_710         ; anti-debugging code building function
mov     eax, dword ptr fs:loc_15+3
mov     eax, [eax+30h]

                        ; DATA XREF: seg000:0000000C↑r
cmp     byte ptr [eax+2], 1 ; PEB!IsDebugged
jz      loc_703         ; jmp to division by zero
mov     eax, dword ptr fs:loc_2C+4
mov     al, [eax+68h]
and     al, 70h         ; PEB!NtGlobalFlags
cmp     al, 70h ; 'p'

                        ; DATA XREF: seg000:0000001F↑r
                        ; seg000:0000004A↓r ...
jz      loc_703         ; jmp to division by zero
mov     eax, 1
```

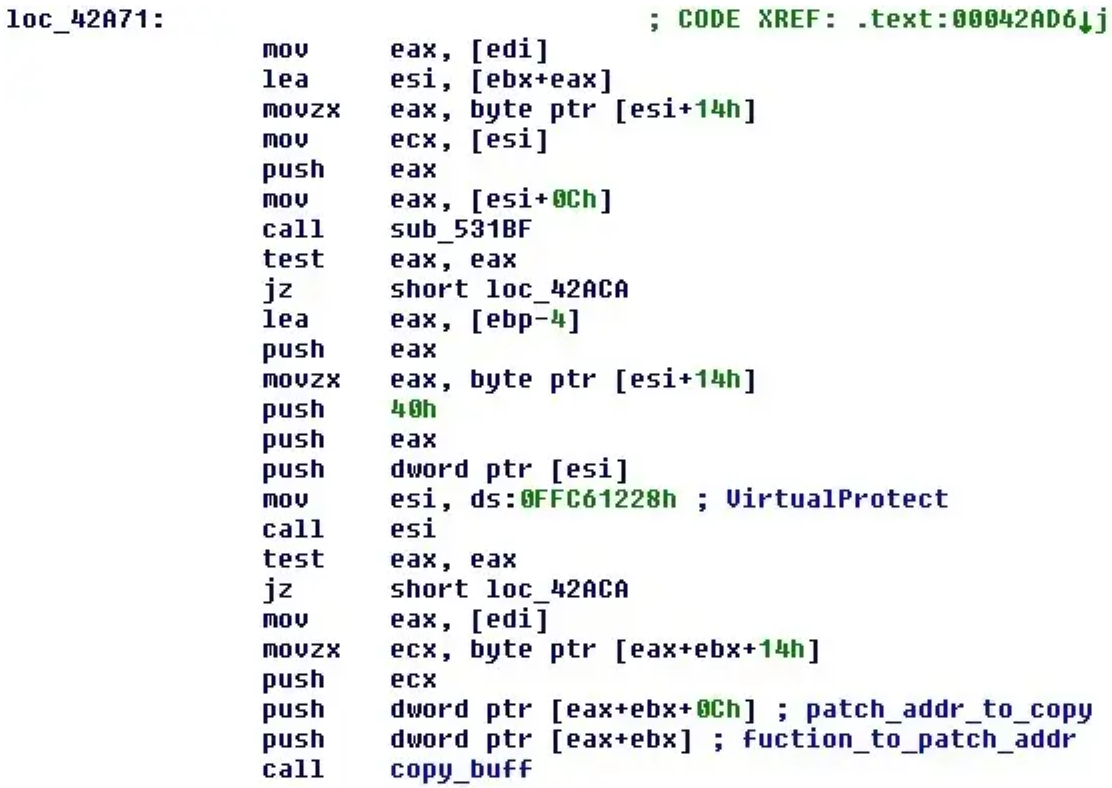
```
;
; START OF FUNCTION CHUNK FOR sub_62

loc_703:                                ; CODE XREF: seg000:00000019↑j
                                        ; seg000:loc_2C↑j ...
                xor     ebx, ebx

loc_705:                                ; CODE XREF: sub_20E+47C↑j
                idiv    ebx
                idiv    eax
```

The new anti-VM capabilities that were added to this variant of ZeuS are not so impressive: The malware simply checks if VMware Tools is installed on the machine (VMware Tools is a free, optional suite of utilities that enhance the performance of the virtual machine's guest operating system and improves management of the virtual machine). To bypass this check, malware researchers can simply uninstall VMware Tools.

Browser Patching and Web-Injection

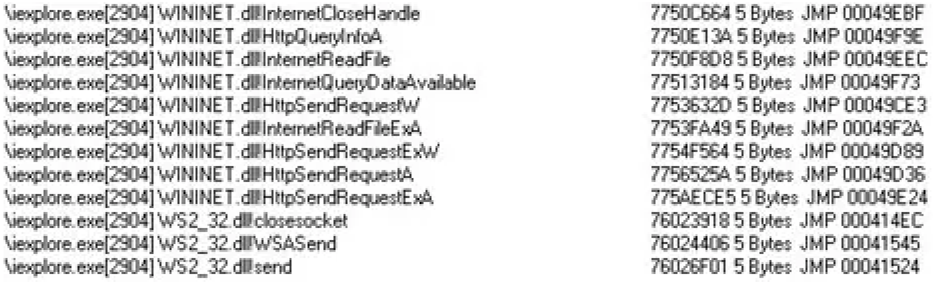
ZeuS.Maple uses browser patching to implement Web-injection functionality, which facilitates information stealing and financial fraud. Browser patching on its own isn't new to ZeuS; however, ZeuS.Maple is the only variant that also re-patches the browser in order to protect its patches and ensure that they stay in place.

In the figure below, the code repeatedly goes over some function addresses and writes the patched function over the function address.



```
loc_42A71:                              ; CODE XREF: .text:00042AD6↓j
                mov     eax, [edi]
                lea     esi, [ebx+eax]
                movzx   eax, byte ptr [esi+14h]
                mov     ecx, [esi]
                push    eax
                mov     eax, [esi+0Ch]
                call    sub_531BF
                test    eax, eax
                jz      short loc_42ACA
                lea     eax, [ebp-4]
                push    eax
                movzx   eax, byte ptr [esi+14h]
                push    40h
                push    eax
                push    dword ptr [esi]
                mov     esi, ds:0FFC61228h ; VirtualProtect
                call    esi
                test    eax, eax
                jz      short loc_42ACA
                mov     eax, [edi]
                movzx   ecx, byte ptr [eax+ebx+14h]
                push    ecx
                push    dword ptr [eax+ebx+0Ch] ; patch_addr_to_copy
                push    dword ptr [eax+ebx] ; fuction_to_patch_addr
                call    copy_buff
```

Important patch list on Internet Explorer:



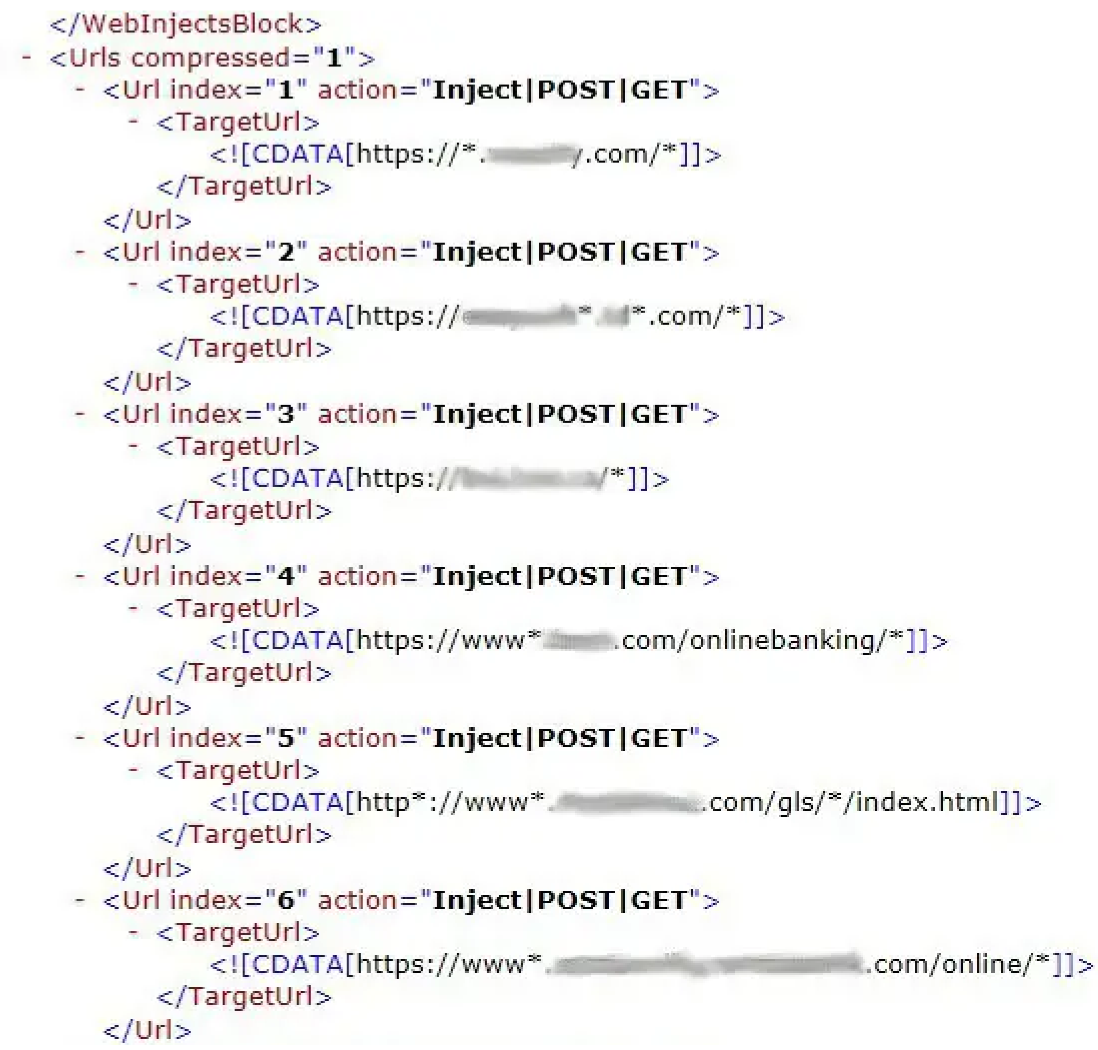
```
\iexplore.exe[2904] WININET.dll!InternetCloseHandle          7750C664 5 Bytes  JMP 00049EBF
\iexplore.exe[2904] WININET.dll!HttpQueryInfoA               7750E13A 5 Bytes  JMP 00049F9E
\iexplore.exe[2904] WININET.dll!InternetReadFile             7750F8D8 5 Bytes  JMP 00049EEC
\iexplore.exe[2904] WININET.dll!InternetQueryDataAvailable   77513184 5 Bytes  JMP 00049F73
\iexplore.exe[2904] WININET.dll!HttpSendRequestW             7753632D 5 Bytes  JMP 00049CE3
\iexplore.exe[2904] WININET.dll!InternetReadFileExA          7753FA49 5 Bytes  JMP 00049F2A
\iexplore.exe[2904] WININET.dll!HttpSendRequestExW           7754F564 5 Bytes  JMP 00049D89
\iexplore.exe[2904] WININET.dll!HttpSendRequestA             7756525A 5 Bytes  JMP 00049D36
\iexplore.exe[2904] WININET.dll!HttpSendRequestExA           775AECE5 5 Bytes  JMP 00049E24
\iexplore.exe[2904] WS2_32.dll!closesocket                   76023918 5 Bytes  JMP 000414EC
\iexplore.exe[2904] WS2_32.dll!WSASend                       76024406 5 Bytes  JMP 00041545
\iexplore.exe[2904] WS2_32.dll!send                          76026F01 5 Bytes  JMP 00041524
```

The Encrypted Configuration

Like other ZeuS variants, ZeuS.Maple's configuration is stored in the Windows registry. However, unlike other variants, it uses the executable name, or a GUID format string, as the name for the registry key (instead of the regular generated name). The data is encrypted with AES-128 instead of RC4 which is commonly used with other ZeuS variants. However this isn't unique since AES-128 has been previously used with other variants. After decrypting the malware configuration, we've noticed that the ZeuS version ID is 3.3.6.0, which indicates that this is a brand new variant of ZeuS, as previously mentioned.

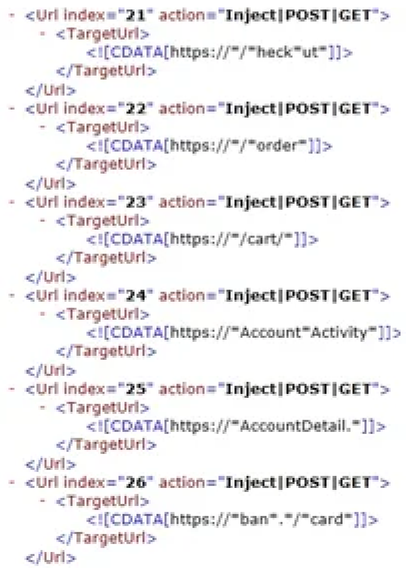
As for the targets, the main targets include 14 leading financial institutions located in Canada. In addition, it contains some "universal" attacks on URLs that consist of generic strings for e-commerce targets.

A sample of the financial institutions targeted as seen in the configuration (shown in IBM Trusteer's format):



```
    </WebInjectsBlock>
  - <Urls compressed="1">
    - <Url index="1" action="Inject|POST|GET">
      - <TargetUrl>
          <![CDATA[https://*.[redacted].com/*]]>
        </TargetUrl>
      </Url>
    - <Url index="2" action="Inject|POST|GET">
      - <TargetUrl>
          <![CDATA[https://[redacted]*[redacted]*.com/*]]>
        </TargetUrl>
      </Url>
    - <Url index="3" action="Inject|POST|GET">
      - <TargetUrl>
          <![CDATA[https://[redacted]/*]]>
        </TargetUrl>
      </Url>
    - <Url index="4" action="Inject|POST|GET">
      - <TargetUrl>
          <![CDATA[https://www*[redacted].com/onlinebanking/*]]>
        </TargetUrl>
      </Url>
    - <Url index="5" action="Inject|POST|GET">
      - <TargetUrl>
          <![CDATA[http*://www*.[redacted].com/gls/*/index.html]]>
        </TargetUrl>
      </Url>
    - <Url index="6" action="Inject|POST|GET">
      - <TargetUrl>
          <![CDATA[https://www*.[redacted].com/online/*]]>
        </TargetUrl>
      </Url>
```

In addition to the listed financial institutions, ZeuS.Maple targets general e-commerce transactions but looks for URLs that contain strings like: ‘order,’ ‘cart,’ ‘account activity’ and more:

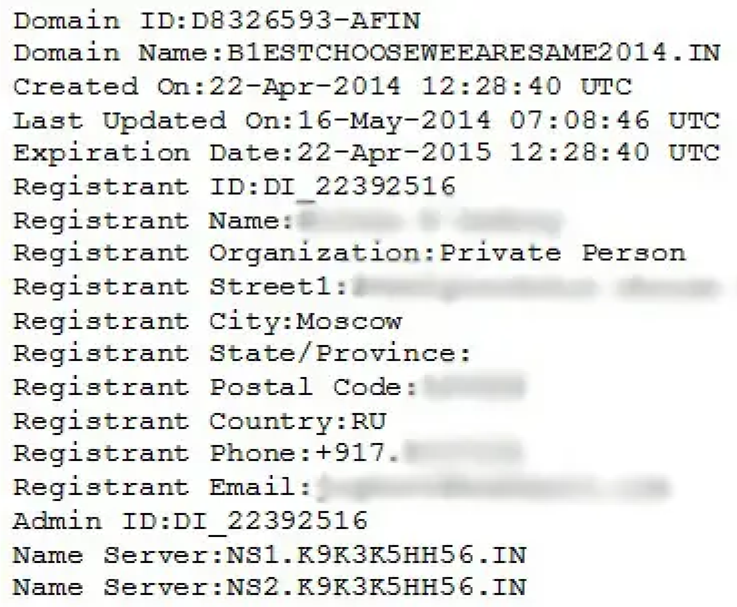


```
- <Url index="21" action="Inject|POST|GET">
  - <TargetUrl>
      <![CDATA[https://*/*heck*ut*]]>
    </TargetUrl>
  </Url>
- <Url index="22" action="Inject|POST|GET">
  - <TargetUrl>
      <![CDATA[https://*/*order*]]>
    </TargetUrl>
  </Url>
- <Url index="23" action="Inject|POST|GET">
  - <TargetUrl>
      <![CDATA[https://*/cart/*]]>
    </TargetUrl>
  </Url>
- <Url index="24" action="Inject|POST|GET">
  - <TargetUrl>
      <![CDATA[https://*Account*Activity*]]>
    </TargetUrl>
  </Url>
- <Url index="25" action="Inject|POST|GET">
  - <TargetUrl>
      <![CDATA[https://*AccountDetail.*]]>
    </TargetUrl>
  </Url>
- <Url index="26" action="Inject|POST|GET">
  - <TargetUrl>
      <![CDATA[https://*ban*.*/*card*]]>
    </TargetUrl>
  </Url>
```

Command and Control Communication

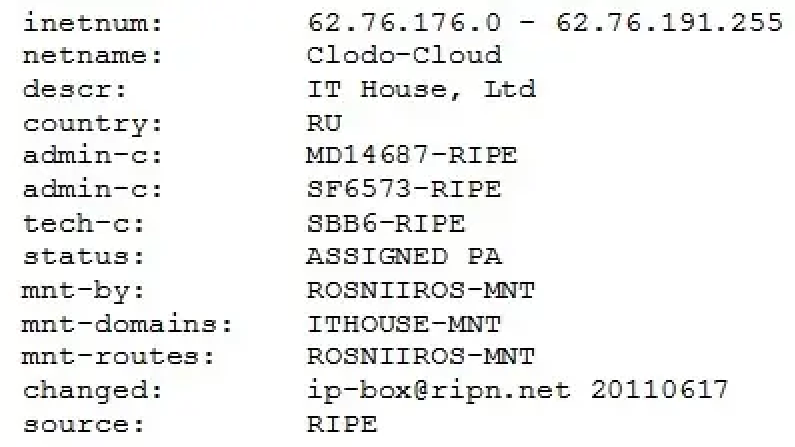
ZeuS.Maple uses nginx-based C&C. Each server has the .in DNS suffix, and the communication is directed to the /www/ folder. The ‘.in’ suffix should be an indicator of the location of the server (India); however, when looking up the server details, we see it is located in Russia. The domain is registered under a fake name and address.

The latest active sample we analyzed communicated with C&C b1estchooseweearesame2014.in/www/ – this resolved to the IP address 62.76.190.115 –



```
Domain ID:D8326593-AFIN
Domain Name:B1ESTCHOOSEWEEARESAME2014.IN
Created On:22-Apr-2014 12:28:40 UTC
Last Updated On:16-May-2014 07:08:46 UTC
Expiration Date:22-Apr-2015 12:28:40 UTC
Registrant ID:DI_22392516
Registrant Name:[redacted]
Registrant Organization:Private Person
Registrant Street1:[redacted]
Registrant City:Moscow
Registrant State/Province:
Registrant Postal Code:[redacted]
Registrant Country:RU
Registrant Phone:+917.[redacted]
Registrant Email:[redacted]
Admin ID:DI_22392516
Name Server:NS1.K9K3K5HH56.IN
Name Server:NS2.K9K3K5HH56.IN
```

The server IP address seems to be registered to a Russian Internet service provider.



```
inetnum:        62.76.176.0 - 62.76.191.255
netname:        Clodo-Cloud
descr:          IT House, Ltd
country:        RU
admin-c:        MD14687-RIPE
admin-c:        SF6573-RIPE
tech-c:         SBB6-RIPE
status:         ASSIGNED PA
mnt-by:         ROSNIIROS-MNT
mnt-domains:    ITHOUSE-MNT
mnt-routes:     ROSNIIROS-MNT
changed:        ip-box@ripn.net 20110617
source:         RIPE
```

Conclusion

The base code of ZeuS 2.0 remains a central source for malware authors as it continues to enable the evolution of the ZeuS malware family. The ZeuS.Maple variant provides an interesting example of new and improved methods used by malware developers to bypass automated security controls as well as human malware researchers.

We expect this trend to continue as we find more sophisticated, stealthy variants of ZeuS targeting specific geographical regions.

Read the white paper: Accelerating growth and digital adoption with seamless identity trust

Dana Tamir
Director of Enterprise Security at Trusteer, an IBM Company

Dana Tamir is Director of Enterprise Security at Trusteer, an IBM Company. In her role she leads activities related to enterprise advanced threat protection ...