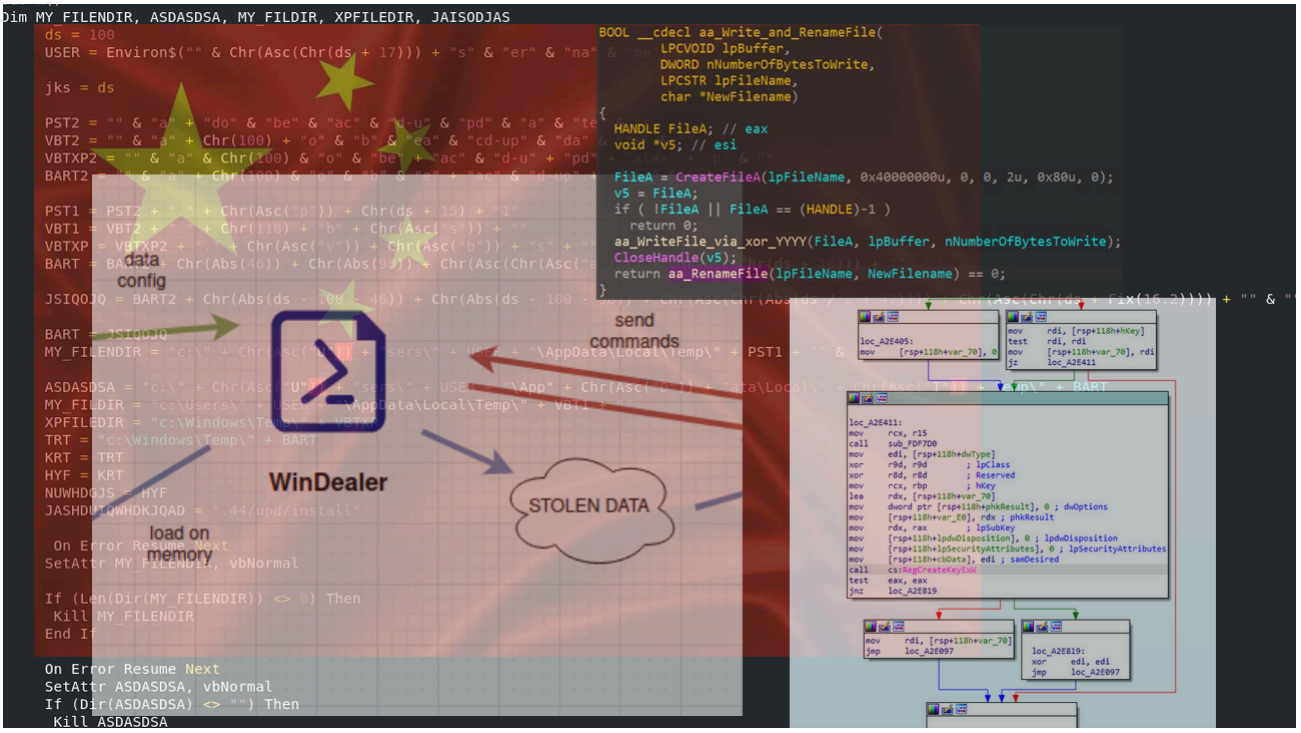
Malware analysis report: WinDealer (LuoYu Threat Group)

By MSSP Research Lab

Published: 2023-05-08 · Archived: 2026-04-05 19:13:17 UTC

7 minute read

WinDealer is a type of malware that is used for financial fraud and theft. It is a banking Trojan that is designed to steal sensitive financial information, such as login credentials, credit card numbers, and other personal information from victims' computers.



Following are the capabilities of the malware:

- Manipulation of files and file systems: reading, writing, and deleting files, listing directories, and collecting disk information
- Information collection: gathering device details, network settings, and/or keyboard layout, listing running processes, installed software, and configuration files of popular messaging services (Skype, QQ, WeChat, and Wangwang);
- Download and upload random file types; arbitrarily executed commands;
- System-wide text file and Microsoft Word document search;
- Screenshot taking;
- Discovery of networks through ping scan;
- Backdoor maintenance: enabling or disabling persistence (through the RUN key in the registry) and configuration changes

Threat actorPermalink

LuoYu is a threat group that is believed to be a Chinese state-sponsored hacking group. The group has been active since at least 2011 and is known to target a wide range of industries, including defense, government, telecommunications, and technology.

TargetPermalink

Geographies and sectors:

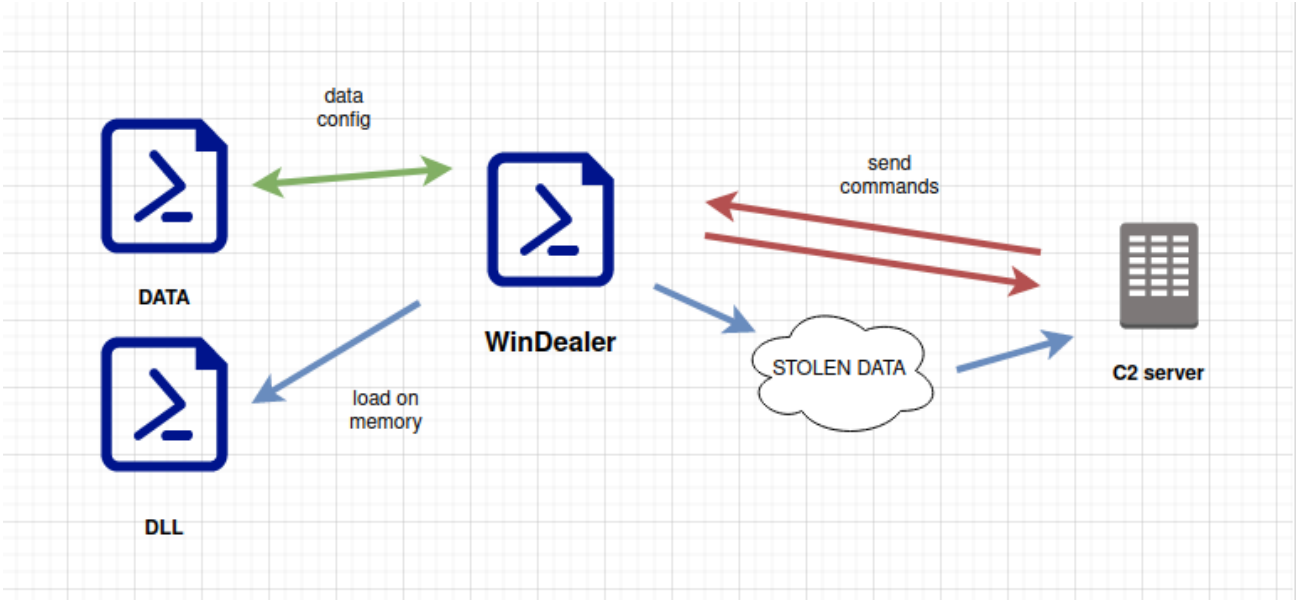
- Chinese subsidiaries of Japanese companies
- Users of a Chinese private bank

Industry:

- Technology
- Media
- Financial
- Military
- Telecom
- Ministries of Foreign Affairs

Cyber Kill ChainPermalink

WinDealer steals information of an infected `PC` and sends it to a `C2` server as described in here:



IdentificationPermalink

Two samples are being investigated:

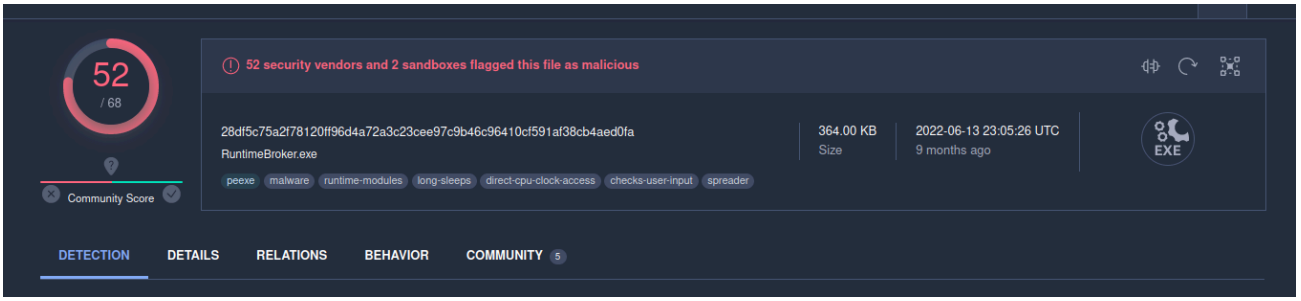
sample.exe:

File size: `372736 bytes`
MD5 sum: `cc7207f09a6fe41c71626ad4d3f127ce`

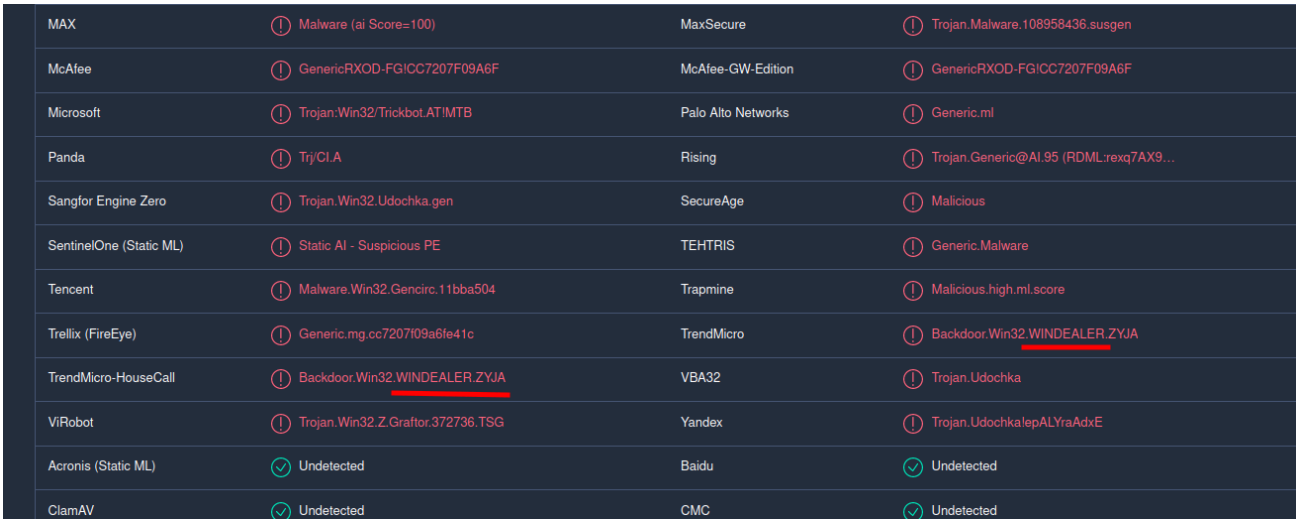
SHA-1 sum: `84e749c37978f9387e16fab29c7b1b291be93a63`
SHA-256 sum: `28df5c75a2f78120ff96d4a72a3c23cee97c9b46c96410cf591af38cb4aed0fa`

First of all, check our sample via VirusTotal:

https://www.virustotal.com/gui/file/28df5c75a2f78120ff96d4a72a3c23cee97c9b46c96410cf591af38cb4aed0fa/details



So, 52 of 68 AV engines detect our sample as malicious.



More of them detect file as `Backdoor.Win32.WINDEALER.ZYJA`.

Static analysis[Permalink]

The specified sample is a PE file:

```
remnux@remnux:~/malware-analysis/2023-03-23-malware-analysis$ file samples/sample.exe
samples/sample.exe: PE32 executable (GUI) Intel 80386, for MS Windows
```

```
remnux@remnux:~/malware-analysis/2023-03-23-malware-analysis$ hexdump -C samples/sample.exe | head
00000000  4d 5a 90 00 03 00 00 00  04 00 00 00 ff ff 00 00  |MZ..............|
00000010  b8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  |........@.......|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 f0 00 00 00  |................|
00000040  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  |is program canno|
00000060  74 20 62 65 20 72 75 6e  20 69 6e 20 44 4f 53 20  |t be run in DOS |
00000070  6d 6f 64 65 2e 0d 0d 0a  24 00 00 00 00 00 00 00  |mode....$.......|
00000080  ba 28 ec 17 fe 49 82 44  fe 49 82 44 fe 49 82 44  |.(...I.D.I.D.I.D|
00000090  85 55 8e 44 f8 49 82 44  7d 41 df 44 fa 49 82 44  |.U.D.I.D}A.D.I.D|
remnux@remnux:~/malware-analysis/2023-03-23-malware-analysis$
```

Use `exiftool` for looking metadata:

```
└─$ exiftool samples/sample.exe
ExifTool Version Number         : 12.49
File Name                       : sample.exe
Directory                       : samples
File Size                       : 373 kB
File Modification Date/Time     : 2022:02:01 19:18:04+03:00
File Access Date/Time           : 2023:03:30 10:55:29+03:00
File Inode Change Date/Time     : 2023:03:30 10:54:40+03:00
File Permissions                : -rw-r--r--
File Type                       : Win32 EXE
File Type Extension             : exe
MIME Type                       : application/octet-stream
Machine Type                    : Intel 386 or later, and compatibles
Time Stamp                      : 2021:01:25 13:32:26+03:00
Image File Characteristics      : No relocs, Executable, No line numbers,
PE Type                         : PE32
Linker Version                  : 6.0
Code Size                       : 86016
Initialized Data Size           : 303104
Uninitialized Data Size         : 0
Entry Point                     : 0x15020
OS Version                      : 4.0
Image Version                   : 0.0
Subsystem Version               : 4.0
Subsystem                       : Windows GUI
File Version Number             : 11.0.18362.267
Product Version Number          : 11.0.18362.267
File Flags Mask                 : 0x003f
File Flags                      : (none)
File OS                         : Win32
Object File Type                : Executable application
File Subtype                    : 0
Language Code                   : English (U.S.)
Character Set                   : Unicode
Comments                        :
Company Name                    :
File Description                : Runtime Broker
File Version                    : 11, 0, 18362, 267
Internal Name                   : Microsoft Windows Operating System
```

And we see that file timestamp is `2021-01-25 13:32:26+03.00`

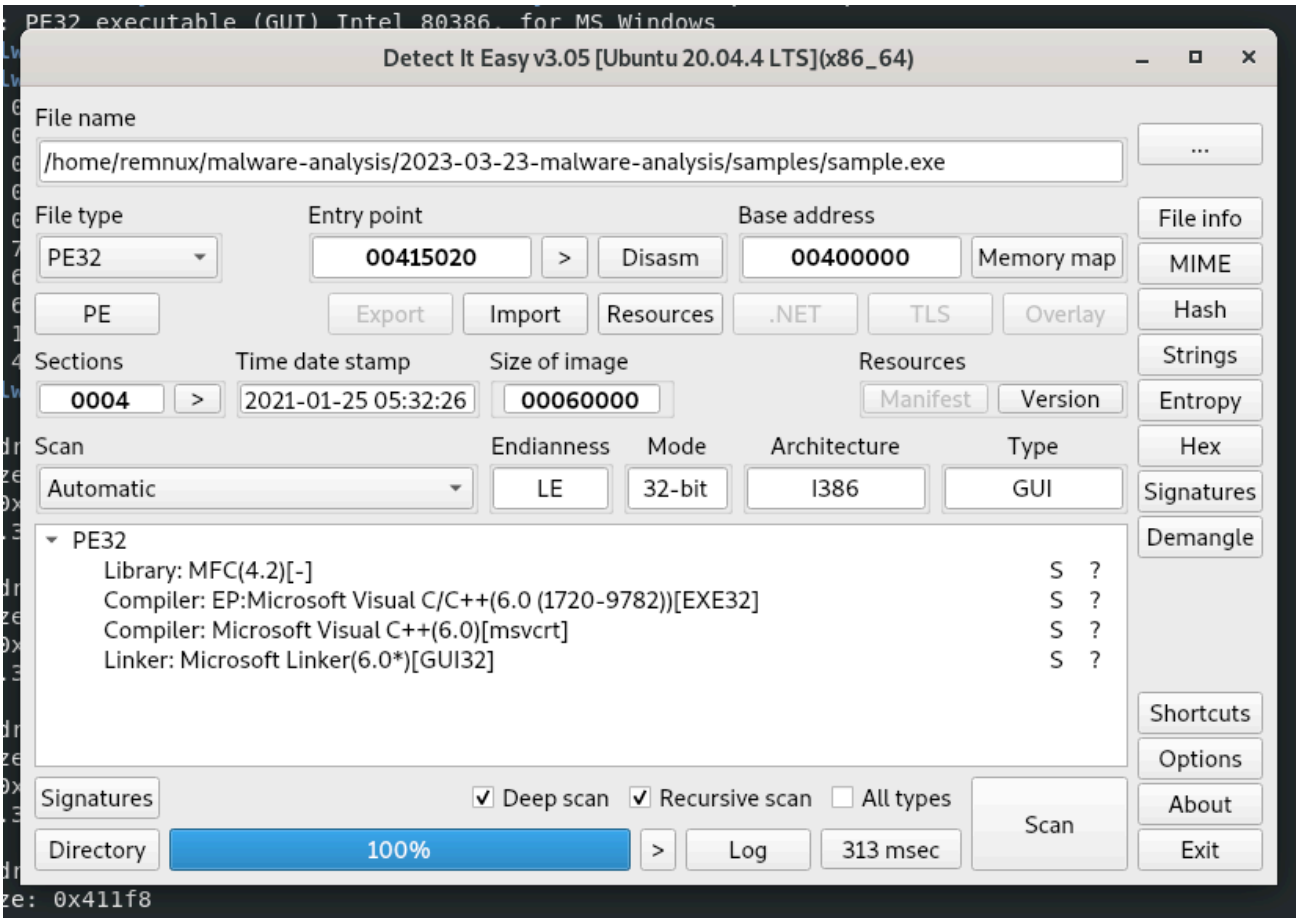
Executable sample is not packed by `upx`:

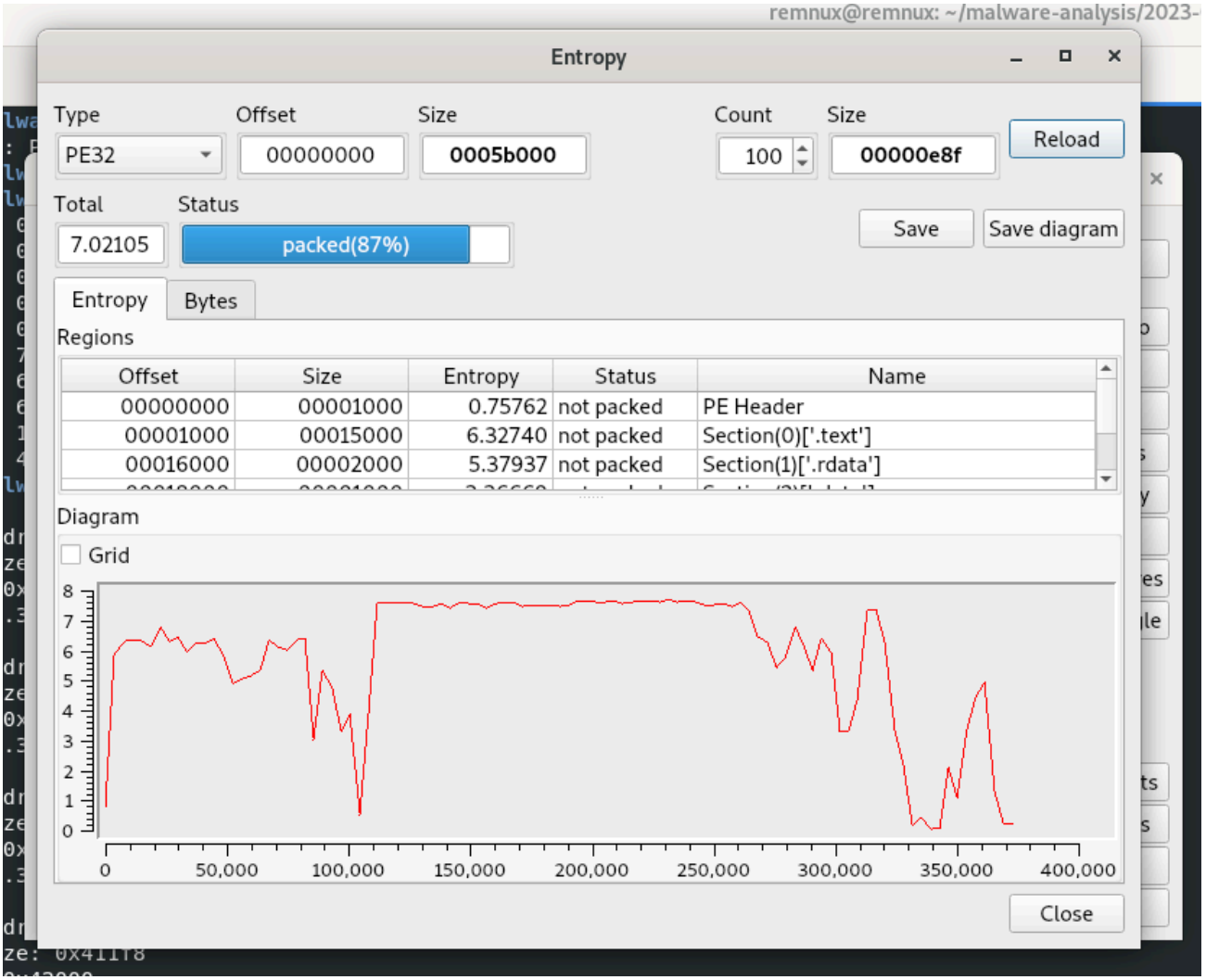
```
                       Ultimate Packer for eXecutables
                          Copyright (C) 1996 - 2020
UPX 3.96        Markus Oberhumer, Laszlo Molnar & John Reiser   Jan 23rd 2020

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
upx: samples/sample.exe: NotPackedException: not packed by UPX
```

What about Shannon entropy of the sample:

```
.text
        virtual address: 0x1000
        virtual size: 0x144c6
        raw size: 0x15000
        entropy: 6.327385942283466
.rdata
        virtual address: 0x16000
        virtual size: 0x1d82
        raw size: 0x2000
        entropy: 5.379194415510608
.data
        virtual address: 0x18000
        virtual size: 0x5ca8
        raw size: 0x1000
        entropy: 3.366340507761682
.rsrc
        virtual address: 0x1e000
        virtual size: 0x411f8
        raw size: 0x42000
        entropy: 7.04482469997232
```

Analysze with DIE says that the compiler is `Microsoft Visual Studio C++ (6.0)` :

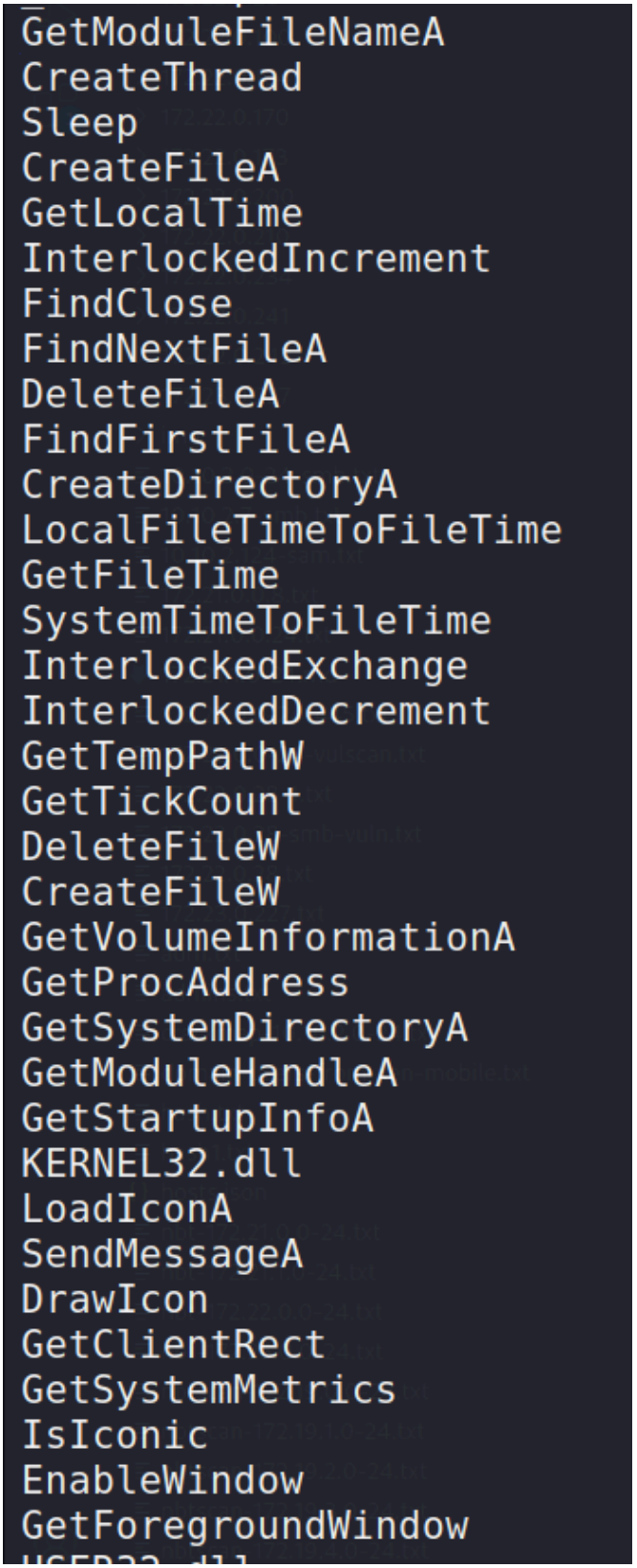




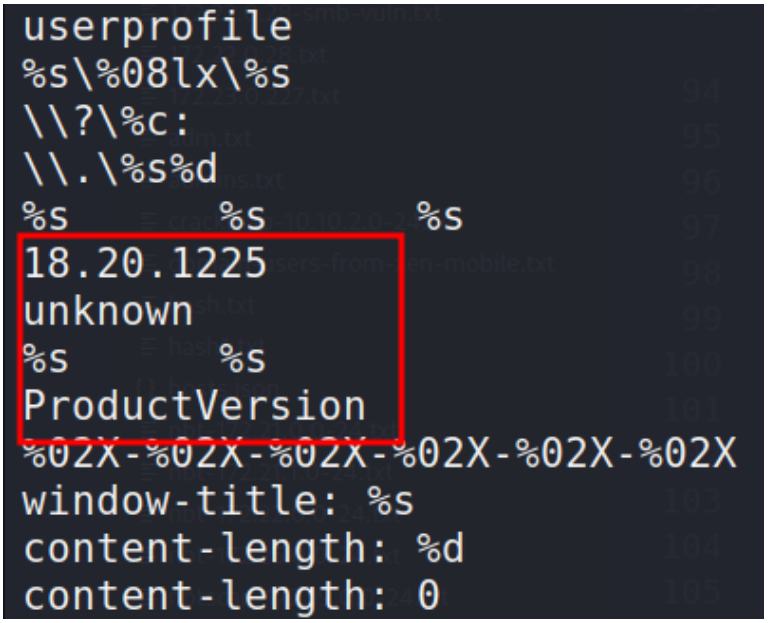
Malware contains encrypted DLL:

```
000171f0       000173dc 00000000 00000000 00017d40 000161c4

       DLL Name: MSVCP60.dll
       vma:  Hint/Ord Member-Name Bound-To
       17cb0     813  ?_C@?1??_Nullstr@?$basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@std@@CAPBDXZ@4DB
       17c0a    1052  ?assign@?$basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@std@@QAEAAV12@ABV12@II@Z
       17bba    1016  ?_Tidy@?$basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@std@@AAEX_N@Z
       17b62    1056  ?assign@?$basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@std@@QAEAAV12@PBDI@Z
       17b18     233  ??1?$basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@std@@QAE@XZ
       17b00     269  ??1_Winit@std@@QAE@XZ
       17ae8     165  ??0_Winit@std@@QAE@XZ
       17ac8     265  ??1Init@ios_base@std@@QAE@XZ
       17aa8     158  ??0Init@ios_base@std@@QAE@XZ
       17d0c     267  ??1_Lockit@std@@QAE@XZ
       17d26     162  ??0_Lockit@std@@QAE@XZ
       17c66    1633  ?npos@?$basic_string@DU?$char_traits@D@std@@V?$allocator@D@2@@std@@2IB

00017204       00000000 00000000 00000000 00000000 00000000
```

Interesting strings:



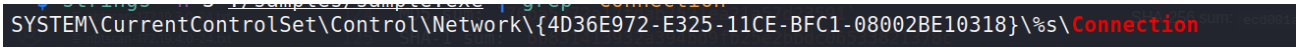
The hardcoded version of WinDealer:



`18.20.1225` - version: `18`, year: `2020`, month and day: `12.25`

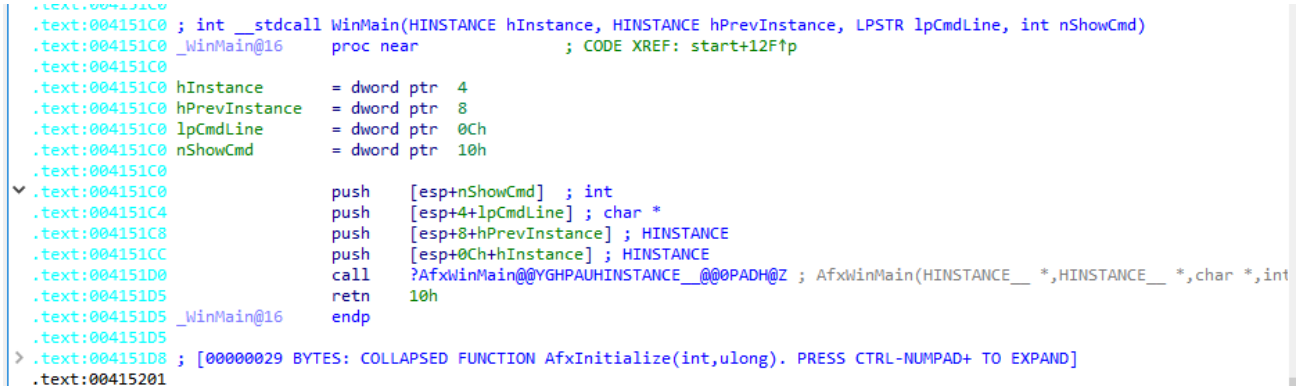
another intersting strings is:

`SYSTEM\CurrentControlSet\Control\Network\{4D36E972-E325-11CE-BFC1-08002BE10318}\%s\Connection`:

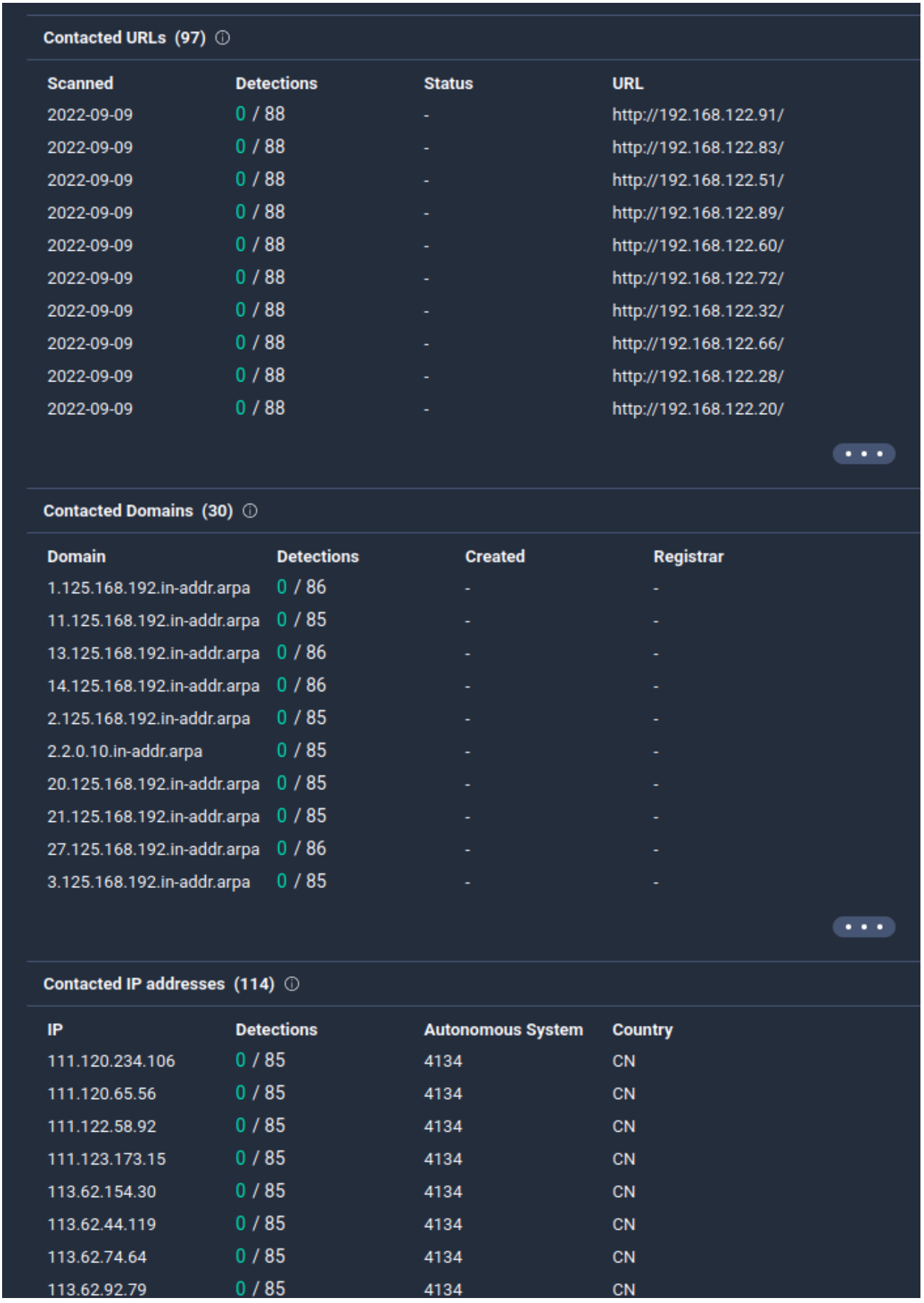


Dynamic analysisPermalink

The sample is GUI application:



Contacted IP addresses is:



Contacted URLs (97)

Scanned	Detections	Status	URL
2022-09-09	0 / 88	-	http://192.168.122.91/
2022-09-09	0 / 88	-	http://192.168.122.83/
2022-09-09	0 / 88	-	http://192.168.122.51/
2022-09-09	0 / 88	-	http://192.168.122.89/
2022-09-09	0 / 88	-	http://192.168.122.60/
2022-09-09	0 / 88	-	http://192.168.122.72/
2022-09-09	0 / 88	-	http://192.168.122.32/
2022-09-09	0 / 88	-	http://192.168.122.66/
2022-09-09	0 / 88	-	http://192.168.122.28/
2022-09-09	0 / 88	-	http://192.168.122.20/

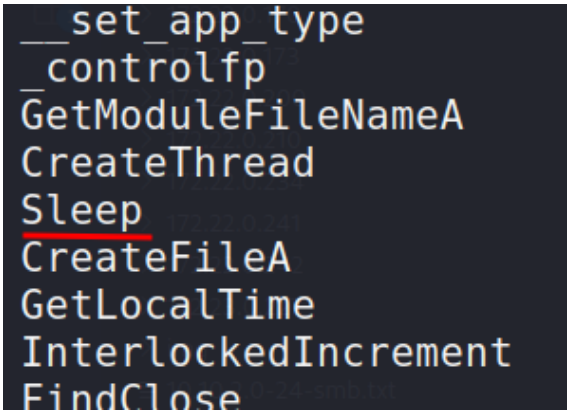
Contacted Domains (30)

Domain	Detections	Created	Registrar
1.125.168.192.in-addr.arpa	0 / 86	-	-
11.125.168.192.in-addr.arpa	0 / 85	-	-
13.125.168.192.in-addr.arpa	0 / 86	-	-
14.125.168.192.in-addr.arpa	0 / 86	-	-
2.125.168.192.in-addr.arpa	0 / 85	-	-
2.2.0.10.in-addr.arpa	0 / 85	-	-
20.125.168.192.in-addr.arpa	0 / 85	-	-
21.125.168.192.in-addr.arpa	0 / 85	-	-
27.125.168.192.in-addr.arpa	0 / 86	-	-
3.125.168.192.in-addr.arpa	0 / 85	-	-

Contacted IP addresses (114)

IP	Detections	Autonomous System	Country
111.120.234.106	0 / 85	4134	CN
111.120.65.56	0 / 85	4134	CN
111.122.58.92	0 / 85	4134	CN
111.123.173.15	0 / 85	4134	CN
113.62.154.30	0 / 85	4134	CN
113.62.44.119	0 / 85	4134	CN
113.62.74.64	0 / 85	4134	CN
113.62.92.79	0 / 85	4134	CN

May sleep (evasive loops) to hinder dynamic analysis:




```
.idata:00416000                                         ; DATA XREF: ...
.idata:00416004 ; void (__stdcall *Sleep)(DWORD dwMilliseconds)
.idata:00416004                 extrn Sleep:dword       ; CODE XREF: sub_4022BE+11↑p
.idata:00416004                                         ; sub_402C9C+D4↑p ...
.idata:00416008 ; HANDLE (__stdcall *CreateFileA)(LPCSTR lpFileName, DWORD dwDesiredAccess, DWORD dwShareMode, LPSECURITY_ATTRIB
.idata:00416008                 extrn CreateFileA:dword ; CODE XREF: sub_403CBA+AE↑p
.idata:00416008                                         ; sub_403FE4+AE↑p ...
```

The operator has the power to rename, move, and delete files on the target machine:

```
.idata:00416004 ; void (__stdcall *Sleep)(DWORD dwMilliseconds)
.idata:00416004                 extrn Sleep:dword       ; CODE XREF: sub_4022BE+11↑p
.idata:00416004                                         ; sub_402C9C+D4↑p ...
.idata:00416008 ; HANDLE (__stdcall *CreateFileA)(LPCSTR lpFileName, DWORD dwDesiredAccess, DWORD dwShareMode, LPSECURITY_ATTRIBUT
.idata:00416008                 extrn CreateFileA:dword ; CODE XREF: sub_403CBA+AE↑p
.idata:00416008                                         ; sub_403FE4+AE↑p ...
.idata:0041600C ; void (__stdcall *GetLocalTime)(LPSYSTEMTIME lpSystemTime)
.idata:0041600C                 extrn GetLocalTime:dword
```

```
.idata:00416044                                         ; DATA XREF: sub_40A1A7+11D↑r
.idata:00416048 ; HANDLE (__stdcall *CreateFileW)(LPCWSTR lpFileName, DWORD dwDesiredAccess, DWORD dwShareMode, LPSECURITY_ATTRIBU
.idata:00416048                 extrn CreateFileW:dword ; CODE XREF: sub_40A360+17↑p
.idata:00416048                                         ; DATA XREF: sub_40A360+17↑r
```

```
.idata:00416040                 extrn GetTickCount:dword
.idata:00416040                                         ; CODE XREF: sub_409905↑p
.idata:00416040                                         ; sub_414433+1EF↑p ...
.idata:00416044 ; BOOL (__stdcall *DeleteFileW)(LPCWSTR lpFileName)
.idata:00416044                 extrn DeleteFileW:dword ; CODE XREF: sub_40A1A7+11D↑p
.idata:00416044                                         ; DATA XREF: sub_40A1A7+11D↑r
.idata:00416048 ; HANDLE (__stdcall *CreateFileW)(LPCWSTR lpFileName, DWORD dwDesiredAccess
.idata:00416048                 extrn CreateFileW:dword ; CODE XREF: sub_40A360+17↑p
.idata:00416048                                         ; DATA XREF: sub_40A360+17↑r
```

Also malware search through directories and enum filesystem:

```
.idata:00416010                                         ; sub_407935+3B↑p ...
.idata:00416014 ; BOOL (__stdcall *FindClose)(HANDLE hFindFile)
.idata:00416014                 extrn FindClose:dword   ; CODE XREF: sub_404333+24C↑p
.idata:00416014                                         ; sub_40458D+483↑p ...
.idata:00416018 ; BOOL (__stdcall *FindNextFileA)(HANDLE hFindFile, LPWIN32_FIND_DATAA lpFindFileData)
.idata:00416018                 extrn FindNextFileA:dword
.idata:00416018                                         ; CODE XREF: sub_404333+23B↑p
.idata:00416018                                         ; sub_40458D+4A2↑p ...
.idata:0041601C ; BOOL (__stdcall *DeleteFileA)(LPCSTR lpFileName)
.idata:0041601C                 extrn DeleteFileA:dword ; CODE XREF: sub_404333+132↑p
.idata:0041601C                                         ; sub_40458D+492↑p ...
.idata:00416020 ; HANDLE (__stdcall *FindFirstFileA)(LPCSTR lpFileName, LPWIN32_FIND_DATAA lpFindFileData)
.idata:00416020                 extrn FindFirstFileA:dword
.idata:00416020                                         ; CODE XREF: sub_404333+A7↑p
.idata:00416020                                         ; sub_40458D+187↑p ...
.idata:00416024 ; BOOL (__stdcall *CreateDirectoryA)(LPCSTR lpPathName, LPSECURITY_ATTRIBUTES lpSecurityAttributes)
.idata:00416024                 extrn CreateDirectoryA:dword
.idata:00416024                                         ; CODE XREF: sub_4065A2+1A4↑p
.idata:00416024                                         ; DATA XREF: sub_4065A2+1A4↑r
```

and collecting volume information:

```
.idata:00416048                                         ; DATA XREF: sub_40A360+17↑r
.idata:0041604C ; BOOL (__stdcall *GetVolumeInformationA)(LPCSTR lpRootPathName, LPSTR lpVolumeNameBuffer, DWORD nVolumeNameSize,
.idata:0041604C                 extrn GetVolumeInformationA:dword
.idata:0041604C                                         ; CODE XREF: sub_40C0B3+165↑p
.idata:0041604C                                         ; DATA XREF: sub_40C0B3+165↑r
```

API hookingPermalink

Using `InterlockedExchange`, probably the malware sample is hooking the winapi functions:

```
.idata:00416030                                         ; DATA XREF: ...
.idata:00416034 ; LONG (__stdcall *InterlockedExchange)(volatile LONG *Target, LONG Value)
.idata:00416034                 extrn InterlockedExchange:dword
.idata:00416034                                         ; CODE XREF: sub_407935+E9↑p
.idata:00416034                                         ; sub_408014+15↑p ...
.idata:00416038 ; LONG (__stdcall *InterlockedDecrement)(volatile LONG *lpAddend)
.idata:00416038                 extrn InterlockedDecrement:dword
.idata:00416038                                         ; CODE XREF: sub_407935+C5↑p
.idata:00416038                                         ; sub_407935+D9↑p
.idata:00416038                                         ; DATA XREF: ...
.idata:0041603C ; DWORD (__stdcall *GetTempPathW)(DWORD nBufferLength, LPWSTR lpBuffer)
```

AV/Sandbox evasionPermalink

In the malware sample above, the delay timeout is set using the `GetTickCount()` timer function. The `Sleep()` function is called in a loop until the timer timeout. In the sandbox, delays that are performed by the `Sleep()` function are skipped (replaced with a very short timeout) and the virtually elapsed time will be much higher than the requested timeout. The concept behind these methods is to measure elapsed time while running several forms of delays in parallel:

```
.idata:0041603C                                         ; DATA XREF: sub_408238+100↑r
.idata:00416040 ; DWORD (__stdcall *GetTickCount)()
.idata:00416040                 extrn GetTickCount:dword
.idata:00416040                                         ; CODE XREF: sub_409905↑p
.idata:00416040                                         ; sub_414433+1EF↑p ...
.idata:00416044 ; BOOL (__stdcall *DeleteFileW)(LPCWSTR lpFileName)
```

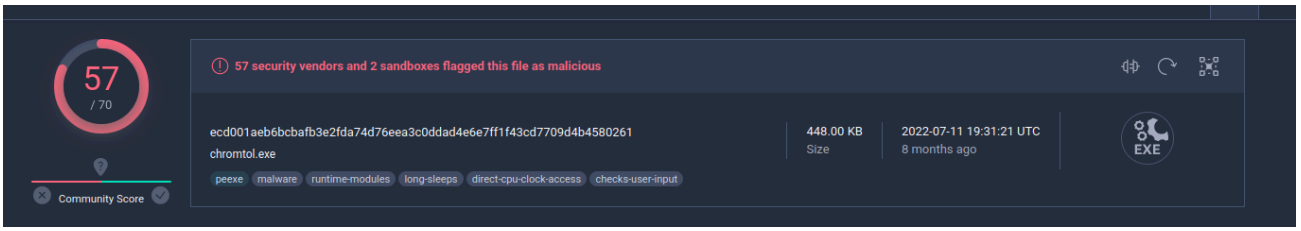
---

sample2.exe:

File size: `458752 bytes`
MD5 sum: `76ba5272a17fdab7521ea21a57d23591`
SHA-1 sum: `6b831413932a394bd9fb25e2bbdc06533821378c`
SHA-256 sum: `ecd001aeb6bcbafb3e2fda74d76eea3c0ddad4e6e7ff1f43cd7709d4b4580261`

VirusTotal scan result:

https://www.virustotal.com/gui/file/ecd001aeb6bcbafb3e2fda74d76eea3c0ddad4e6e7ff1f43cd7709d4b4580261/detection



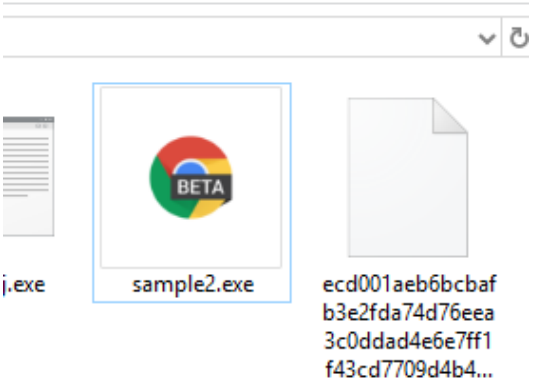
Rising	Trojan.Generic@AI.98 (RDML:LHNE4akpK...	Sangfor Engine Zero	Trojan.Win32.Udochka.gen
SecureAge	Malicious	SentinelOne (Static ML)	Static AI - Suspicious PE
Sophos	Mal/Generic-S	Symantec	Trojan Horse
TACHYON	Trojan/W32.Udochka.458752	Tencent	Malware.Win32.Gencirc.11f9795d
Trapmine	Malicious.moderate.ml.score	Trellix (FireEye)	Generic.mg.76ba5272a17fdab7
TrendMicro	Backdoor.Win32.WINDEALER.ZYJA	TrendMicro-HouseCall	Backdoor.Win32.WINDEALER.ZYJA
VBA32	Trojan.Udochka	VIPRE	Gen:Trojan.Brresmon.Gen.1
ViRobot	Trojan.Win32.S.Agent.458752.OV	Yandex	Trojan.Udochkal/zDrv9OdlJY
Zillya	Trojan.Agent.Win32.1984122	Acronis (Static ML)	Undetected
Baidu	Undetected	ClamAV	Undetected

Static analysis[Permalink]

The specified sample is a PE file:

```
samples/sample2.exe: PE32 executable (GUI) Intel 80386, for MS Windows
```

```
00000000  4d 5a 90 00 03 00 00 00  04 00 00 00 ff ff 00 00  |MZ..............|
00000010  b8 00 00 00 00 00 00 00  40 00 00 00 00 00 00 00  |........@.......|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 01 00 00  |................|
00000040  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  |is program canno|
00000060  74 20 62 65 20 72 75 6e  20 69 6e 20 44 4f 53 20  |t be run in DOS |
00000070  6d 6f 64 65 2e 0d 0d 0a  24 00 00 00 00 00 00 00  |mode....$.......|
00000080  7a 2b ec 17 3e 4a 82 44  3e 4a 82 44 3e 4a 82 44  |z+..>J.D>J.D>J.D|
00000090  45 56 8e 44 38 4a 82 44  bd 42 df 44 3a 4a 82 44  |EV.D8J.D.B.D:J.D|
```



Run exiftool for extracting metadata:

```
└─$ exiftool samples/sample2.exe
ExifTool Version Number         : 12.49
File Name                       : sample2.exe
Directory                       : samples
File Size                       : 459 kB
File Modification Date/Time     : 2022:02:01 19:18:04+03:00
File Access Date/Time           : 2023:03:30 10:56:46+03:00
File Inode Change Date/Time     : 2023:03:30 10:54:17+03:00
File Permissions                : -rw-r--r--
File Type                       : Win32 EXE
File Type Extension             : exe
MIME Type                       : application/octet-stream
Machine Type                    : Intel 386 or later, and compatibles
Time Stamp                      : 2021:03:06 04:13:51+03:00
Image File Characteristics      : No relocs, Executable, No line numbers, No symbols, 32-bit
PE Type                         : PE32
Linker Version                  : 6.0
Code Size                       : 81920
Initialized Data Size           : 393216
Uninitialized Data Size         : 0
Entry Point                     : 0x145f0
OS Version                      : 4.0
Image Version                   : 0.0
Subsystem Version               : 4.0
Subsystem                       : Windows GUI
File Version Number             : 8.8.1314.190
Product Version Number          : 8.8.1314.190
File Flags Mask                 : 0x003f
File Flags                      : (none)
File OS                         : Win32
Object File Type                : Executable application
File Subtype                    : 0
Language Code                   : English (U.S.)
Character Set                   : Unicode
Comments                        :
```

The sample is a Windows GUI file with timestamp: `2021:03:06 04:13:51+03:00`

Dynamic analysis[Permalink]

Generating victim `ID` set in a registry key:

```
mov     eax, [esp+size] ; 0x43
mov     ecx, [esp+buf_str] ; "mac: 00:0C:29:43:FA:A5\x09VMware Virtual NVMe Disk\x09MVaWerN MV_E0000usr"
sub     esp, 10h
lea     edx, [esp+10h+hexdigest]
```

The format of the `victim ID` is `md5("<MAC address>+<Physical_Drive_info>+<username>")`. The malware generates a unique registry entry to store the victim ID for subsequent execution. The `victim ID` is not saved as raw data; instead, the malware changes the `4` bytes `victim ID` to an IP address format.

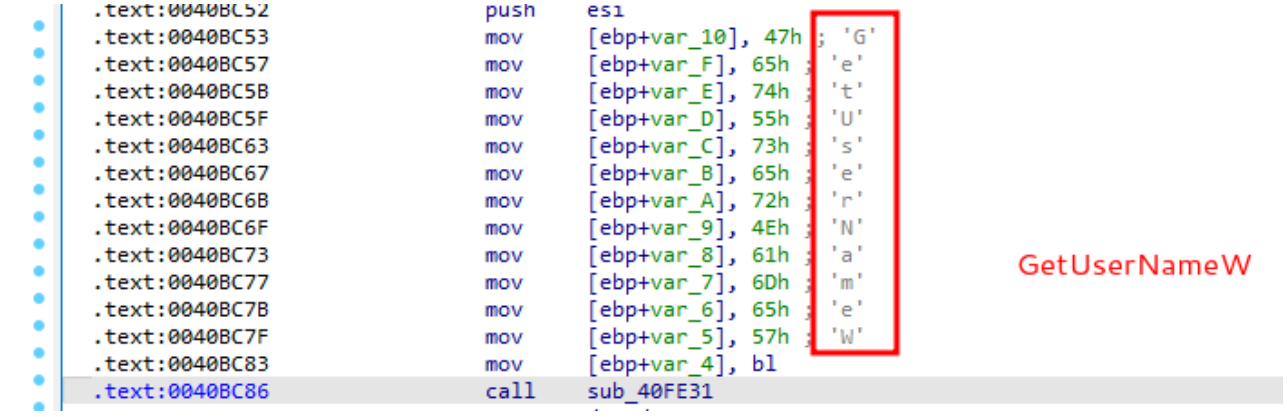
This sample collecting host information:

```
char __thiscall get_victim_info_entry(_DWORD *this)
{
  getComputerName();
  getUsername(this);
  getCPUinfo(this);
  getOSversion(this);
  getNetworkCard(this);
  TimeZone(this);
  getPulbicIP(this);
  if ( this[9] )
    EnumUserInfo(this);
  return 1;
```

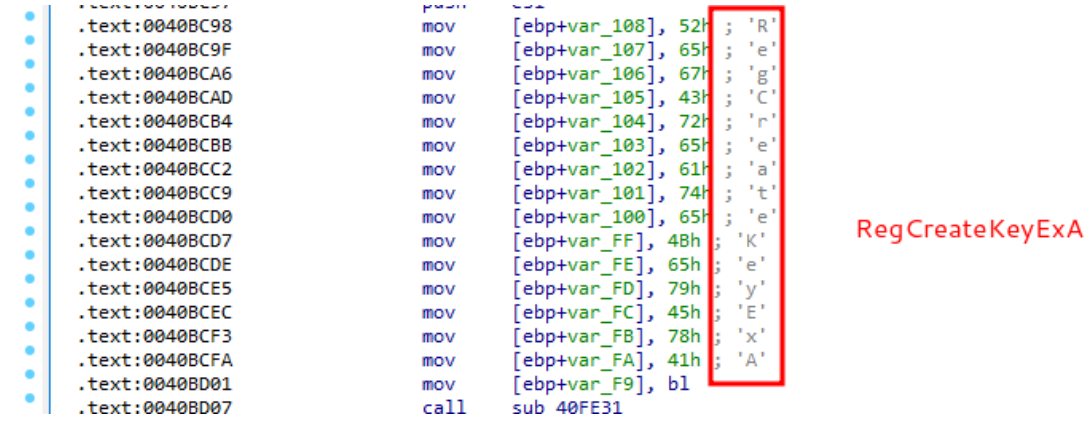
Encoding[Permalink]

Malware sample use function call obfuscation:

`GetUserNameW`:



`RegCreateKeyExA`:



`RegDeleteKeyA` and `RegCloseKey`:

```
.text:0040BAD0    mov     [ebp+var_E8], 52h ; 'R'
.text:0040BAD7    mov     [ebp+var_E7], 65h ; 'e'
.text:0040BADE    mov     [ebp+var_E6], 67h ; 'g'
.text:0040BAE5    mov     [ebp+var_E5], 44h ; 'D'
.text:0040BAEC    mov     [ebp+var_E4], 65h ; 'e'
.text:0040BAF3    mov     [ebp+var_E3], 6Ch ; 'l'
.text:0040BAFA    mov     [ebp+var_E2], 65h ; 'e'
.text:0040BB01    mov     [ebp+var_E1], 74h ; 't'
.text:0040BB08    mov     [ebp+var_E0], 65h ; 'e'
.text:0040BB0F    mov     [ebp+var_DF], 4Bh ; 'K'
.text:0040BB16    mov     [ebp+var_DE], 65h ; 'e'
.text:0040BB1D    mov     [ebp+var_DD], 79h ; 'y'
.text:0040BB24    mov     [ebp+var_DC], 41h ; 'A'
.text:0040BB2B    mov     [ebp+var_DB], bl
.text:0040BB31    call    sub_40FE31
.text:0040BB36    mov     dword_41C8CC, eax
.text:0040BB3B    lea     eax, [ebp+var_9C]
.text:0040BB41    push    eax
.text:0040BB42    push    esi
.text:0040BB43    mov     [ebp+var_9C], 52h ; 'R'
.text:0040BB4A    mov     [ebp+var_9B], 65h ; 'e'
.text:0040BB51    mov     [ebp+var_9A], 67h ; 'g'
.text:0040BB58    mov     [ebp+var_99], 43h ; 'C'
.text:0040BB5F    mov     [ebp+var_98], 6Ch ; 'l'
.text:0040BB66    mov     [ebp+var_97], 6Fh ; 'o'
.text:0040BB6D    mov     [ebp+var_96], 73h ; 's'
.text:0040BB74    mov     [ebp+var_95], 65h ; 'e'
.text:0040BB7B    mov     [ebp+var_94], 4Bh ; 'K'
.text:0040BB82    mov     [ebp+var_93], 65h ; 'e'
.text:0040BB89    mov     [ebp+var_92], 79h ; 'y'
.text:0040BB90    mov     [ebp+var_91], bl
```

RegDeleteKeyA

RegCloseKey

`RegQueryValueExA`:

```
.text:0040B985    push    eax
.text:0040B986    push    esi
.text:0040B987    mov     [ebp+var_11C], 52h ; 'R'
.text:0040B98E    mov     [ebp+var_11B], 65h ; 'e'
.text:0040B995    mov     [ebp+var_11A], 67h ; 'g'
.text:0040B99C    mov     [ebp+var_119], 51h ; 'Q'
.text:0040B9A3    mov     [ebp+var_118], 75h ; 'u'
.text:0040B9AA    mov     [ebp+var_117], 65h ; 'e'
.text:0040B9B1    mov     [ebp+var_116], 72h ; 'r'
.text:0040B9B8    mov     [ebp+var_115], 79h ; 'y'
.text:0040B9BF    mov     [ebp+var_114], 56h ; 'V'
.text:0040B9C6    mov     [ebp+var_113], 61h ; 'a'
.text:0040B9CD    mov     [ebp+var_112], 6Ch ; 'l'
.text:0040B9D4    mov     [ebp+var_111], 75h ; 'u'
.text:0040B9DB    mov     [ebp+var_110], 65h ; 'e'
.text:0040B9E2    mov     [ebp+var_10F], 45h ; 'E'
.text:0040B9E9    mov     [ebp+var_10E], 78h ; 'x'
.text:0040B9F0    mov     [ebp+var_10D], 41h ; 'A'
.text:0040B9F7    mov     [ebp+var_10C], bl
```

RegQueryValueExA

`GetTokenInformation`:

```
.text:0040BE44                push    esi
.text:0040BE45                mov     [ebp+var_130], 47h ; 'G'
.text:0040BE4C                mov     [ebp+var_12F], 65h ; 'e'
.text:0040BE53                mov     [ebp+var_12E], 74h ; 't'
.text:0040BE5A                mov     [ebp+var_12D], 54h ; 'T'
.text:0040BE61                mov     [ebp+var_12C], 6Fh ; 'o'
.text:0040BE68                mov     [ebp+var_12B], 6Bh ; 'k'
.text:0040BE6F                mov     [ebp+var_12A], 65h ; 'e'
.text:0040BE76                mov     [ebp+var_129], 6Eh ; 'n'
.text:0040BE7D                mov     [ebp+var_128], 49h ; 'I'
.text:0040BE84                mov     [ebp+var_127], 6Eh ; 'n'
.text:0040BE8B                mov     [ebp+var_126], 66h ; 'f'
.text:0040BE92                mov     [ebp+var_125], 6Fh ; 'o'
.text:0040BE99                mov     [ebp+var_124], 72h ; 'r'
.text:0040BEA0                mov     [ebp+var_123], 6Dh ; 'm'
.text:0040BEA7                mov     [ebp+var_122], 61h ; 'a'
.text:0040BEAE                mov     [ebp+var_121], 74h ; 't'
.text:0040BEB5                mov     [ebp+var_120], 69h ; 'i'
.text:0040BEBC                mov     [ebp+var_11F], 6Fh ; 'o'
.text:0040BEC3                mov     [ebp+var_11E], 6Eh ; 'n'
.text:0040BECA                mov     [ebp+var_11D], bl
```

GetTokenInformation

`OpenProcessToken`:

```
.text:0040BF47                push    esi
.text:0040BF48                mov     [ebp+var_25C], 4Fh ; 'O'
.text:0040BF4F                mov     [ebp+var_25B], 70h ; 'p'
.text:0040BF56                mov     [ebp+var_25A], 65h ; 'e'
.text:0040BF5D                mov     [ebp+var_259], 6Eh ; 'n'
.text:0040BF64                mov     [ebp+var_258], 50h ; 'P'
.text:0040BF6B                mov     [ebp+var_257], 72h ; 'r'
.text:0040BF72                mov     [ebp+var_256], 6Fh ; 'o'
.text:0040BF79                mov     [ebp+var_255], 63h ; 'c'
.text:0040BF80                mov     [ebp+var_254], 65h ; 'e'
.text:0040BF87                mov     [ebp+var_253], 73h ; 's'
.text:0040BF8E                mov     [ebp+var_252], 73h ; 's'
.text:0040BF95                mov     [ebp+var_251], 54h ; 'T'
.text:0040BF9C                mov     [ebp+var_250], 6Fh ; 'o'
.text:0040BFA3                mov     [ebp+var_24F], 6Bh ; 'k'
.text:0040BFAA                mov     [ebp+var_24E], 65h ; 'e'
.text:0040BFB1                mov     [ebp+var_24D], 6Eh ; 'n'
.text:0040BFB8                mov     [ebp+var_24C], bl
.text:0040BFBE                call    sub_40FE31
```

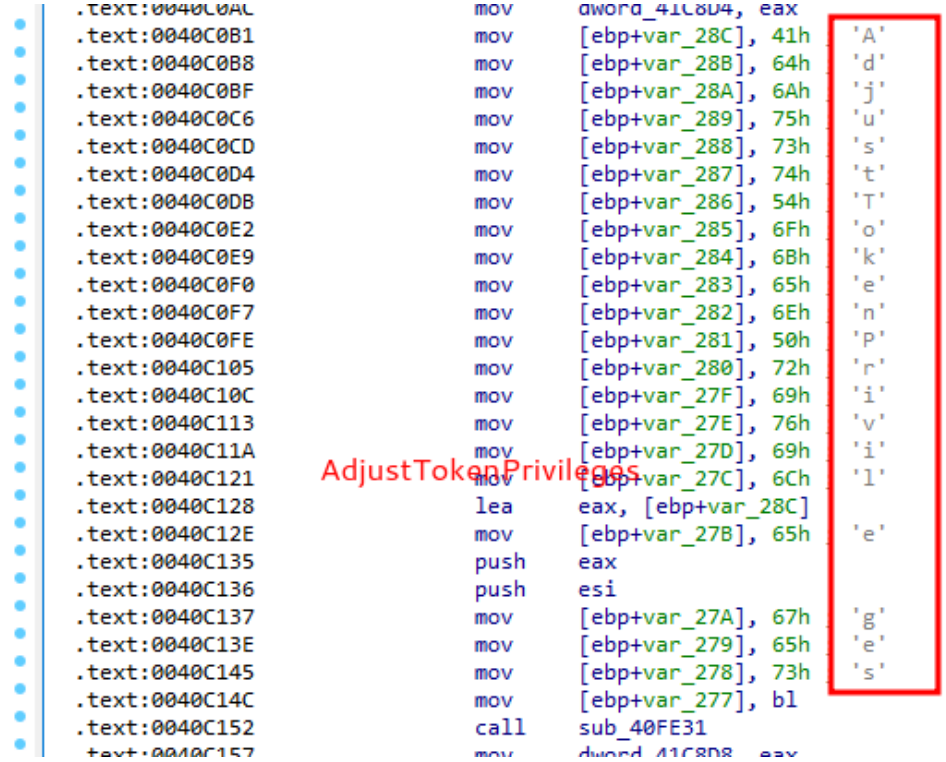
OpenProcessToken

`OpenThreadToken`:

```
.text:0040C163                push    esi
.text:0040C164                mov     [ebp+var_234], 4Fh ; 'O'
.text:0040C16B                mov     [ebp+var_233], 70h ; 'p'
.text:0040C172                mov     [ebp+var_232], 65h ; 'e'
.text:0040C179                mov     [ebp+var_231], 6Eh ; 'n'
.text:0040C180                mov     [ebp+var_230], 54h ; 'T'
.text:0040C187                mov     [ebp+var_22F], 68h ; 'h'
.text:0040C18E                mov     [ebp+var_22E], 72h ; 'r'
.text:0040C195                mov     [ebp+var_22D], 65h ; 'e'
.text:0040C19C                mov     [ebp+var_22C], 61h ; 'a'
.text:0040C1A3                mov     [ebp+var_22B], 64h ; 'd'
.text:0040C1AA                mov     [ebp+var_22A], 54h ; 'T'
.text:0040C1B1                mov     [ebp+var_229], 6Fh ; 'o'
.text:0040C1B8                mov     [ebp+var_228], 6Bh ; 'k'
.text:0040C1BF                mov     [ebp+var_227], 65h ; 'e'
.text:0040C1C6                mov     [ebp+var_226], 6Eh ; 'n'
.text:0040C1CD                mov     [ebp+var_225], bl
.text:0040C1D3                call    sub_40FE31
```

OpenThreadToken

`AdjustTokenPrivileges`:



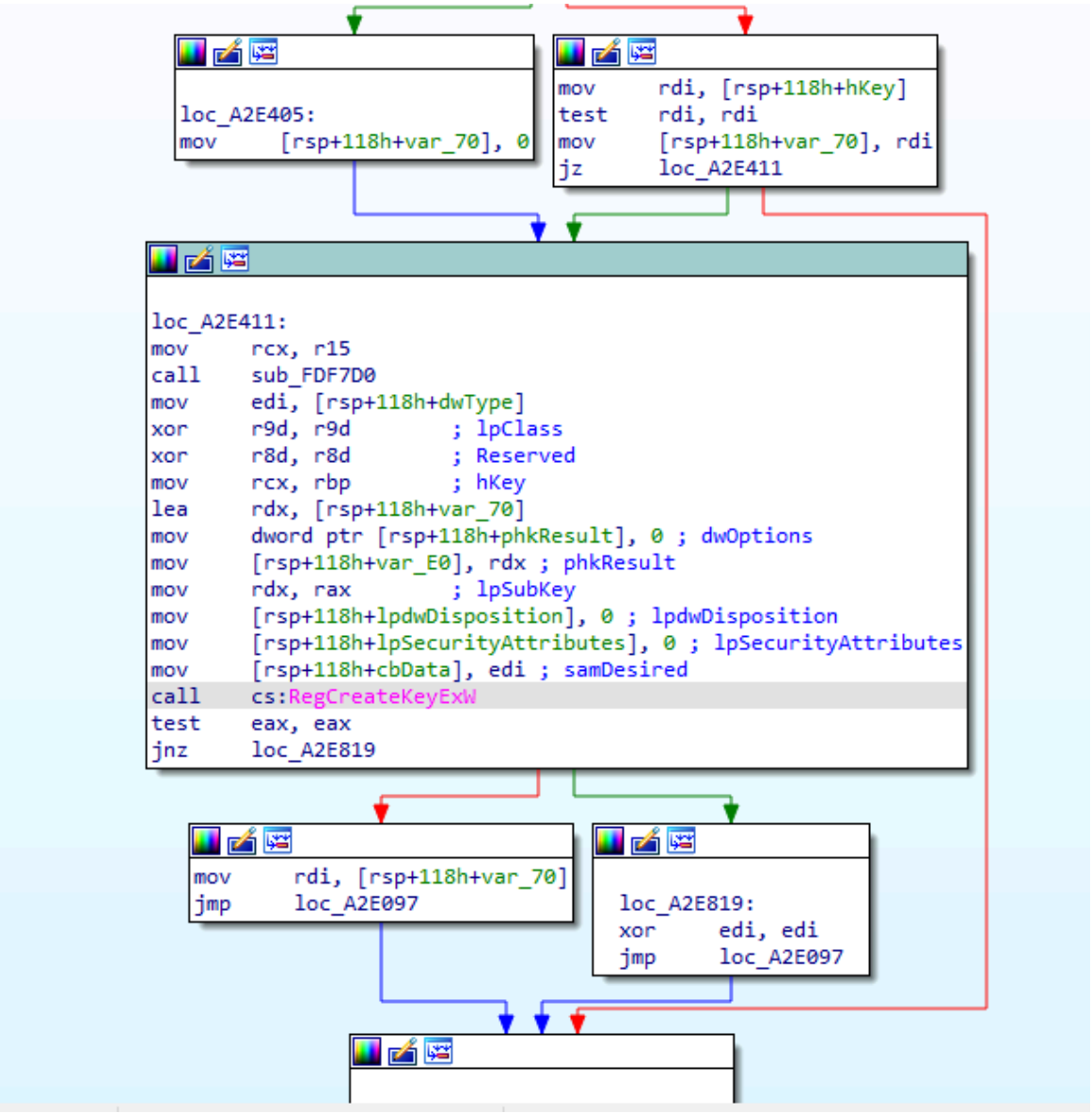
.etc.

So, malware sample use one of the interesting classic APT techniques: Token theft via turn on `SeDebugPrivilege`:

```
 //....
HANDLE token;
TOKEN_PRIVILEGES tp;
LUID luid;
BOOL res = TRUE;

tp.PrivilegeCount = 1;
tp.Privileges[0].Luid = luid;
tp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;

if (!LookupPrivilegeValue(NULL, priv, &luid)) res = FALSE;
if (!OpenProcessToken(GetCurrentProcess(), TOKEN_ADJUST_PRIVILEGES, &token)) res = FALSE;
if (!AdjustTokenPrivileges(token, FALSE, &tp, sizeof(TOKEN_PRIVILEGES), (PTOKEN_PRIVILEGES)NULL, (PDWORD)NULL)) res =
//...
```

Registry Modifications and Persistence[Permalink]

With a high degree of probability, it can be argued that WinDealer has the functionality of interacting with the registry, probably for persistence mechanism:



Encryption[Permalink]

Generate 16 bytes `AES` key to encrypt C2 communication:

```
.idata:004152CC ; int (*sprintf)(char *const Buffer, const char *const Format, ...)
.idata:004152CC                 extrn sprintf:dword     ; CODE XREF: sub_402162+38↑p
.idata:004152CC                                         ; sub_402162+7D↑p ...
```

```
push    14h             ; unsigned int
call    operator new(uint)
push    0               ; Time
mov     esi, eax
call    ds:time
add     esp, 8
mov     edi, eax
push    esi             ; 0x286AD70
call    GetCurrentThreadId
push    eax             ; 0x1CC8
call    GetCurrentProcessId
push    eax             ; 0x1E38
push    edi             ; 0x613F0C54
lea     eax, [esp+4Ch+buf_str]
push    offset aUUUX    ; "%u%u%u%x"
push    eax             ; Buffer
call    ds:sprintf      ; ret. 02EFFA0C = "163152187677367 3682b6ad70"
lea     ecx, [esp+54h+buf_str] ; "163152187677367 3682b6ad70"
push    eax             ; size
push    ecx             ; buf_str = "163152187677367 3682b6ad70"
push    offset aes_key
call    md5_40B5C0      ; ret. aes_key =
                        ; 0041EB40  62 EE BC 0D 65 E4 D7 9D C4 CF 06 6
```

C2 anti-tracking mechanism[Permalink]

This malware sample employs an IP Generation Algorithm to generate a random C2 IP address when the backdoor lacks C2 configuration. The IP produced at random will exist inside particular IP address ranges:

`113.62.0.0 - 113.63.255.255` or
`111.120.0.0 - 111.123.255.255`

```
111.120.191.65:6999 (UDP)
111.120.234.106:6999 (UDP)
111.120.65.56:6999 (UDP)
111.121.61.108:6999 (UDP)
111.122.152.112:6999 (UDP)
111.122.58.92:6999 (UDP)
111.123.173.15:6999 (UDP)
111.123.9.24:6999 (UDP)
113.62.154.30:6999 (UDP)
113.62.175.80:6999 (UDP)
```

This mechanism will prevent researchers from tracking down the real C2 IP.

`Backdoor.Win32.WINDEALER.ZYJA` is a variant of the `WinDealer` malware family. It is a type of backdoor malware that is designed to allow remote attackers to gain unauthorized access to an infected computer system. Once installed, the

malware creates a backdoor on the infected system, which allows the attacker to control the system and steal sensitive data.

The `Backdoor.Win32.WINDEALER.ZYJA` variant is known to be spread through spear-phishing emails that contain malicious attachments. Once the attachment is opened, the malware is installed and begins to communicate with a remote command-and-control server, allowing the attacker to send commands to the infected system and exfiltrate data.

The malware is capable of performing a range of malicious activities, including stealing credentials and sensitive data, taking screenshots, recording keystrokes, and executing arbitrary commands on the infected system. The malware is also capable of bypassing antivirus and other security software, making it difficult to detect and remove.

IOCsPermalink

versionsPermalink

Malware	version	md5	sha1
WinDealer	18.20.1225	76ba5272a17fdab7521ea21a57d23591	6b831413932a394bd9fb25e2bbdc06533821378c
WinDealer	18.20.1225	cc7207f09a6fe41c71626ad4d3f127ce	84e749c37978f9387e16fab29c7b1b291be93a63

domain IPsPermalink

- `113.62.0.0/15 111.120.0.0/14`
- port `55556/TCP` , `6999/UDP`
- `221.195.68.71/32`
- `122.112.245.55/32`

Yara rules (from Malpedia)Permalink

```
rule win_windealer_auto {

   meta:
       author = "Felix Bilstein - yara-signator at cocacoding dot com"
       date = "2023-01-25"
       version = "1"
       description = "Detects win.windealer."
       info = "autogenerated rule brought to you by yara-signator"
       tool = "yara-signator v0.6.0"
       signator_config = "callsandjumps;datarefs;binvalue"
       malpedia_reference = "https://malpedia.caad.fkie.fraunhofer.de/details/win.windealer"
       malpedia_rule_date = "20230124"
       malpedia_hash = "2ee0eebba83dce3d019a90519f2f972c0fcf9686"
       malpedia_version = "20230125"
       malpedia_license = "CC BY-SA 4.0"
       malpedia_sharing = "TLP:WHITE"

   /* DISCLAIMER
```

```
 * The strings used in this rule have been automatically selected from the
 * disassembly of memory dumps and unpacked files, using YARA-Signator.
 * The code and documentation is published here:
 * https://github.com/fxb-cocacoding/yara-signator
 * As Malpedia is used as data source, please note that for a given
 * number of families, only single samples are documented.
 * This likely impacts the degree of generalization these rules will offer.
 * Take the described generation method also into consideration when you
 * apply the rules in your use cases and assign them confidence levels.
 */


strings:
    $sequence_0 = { 668b91d2070000 8a89d0070000 52 51 }
        // n = 4, score = 800
        //   668b91d2070000       | mov                 dx, word ptr [ecx + 0x7d2]
        //   8a89d0070000         | mov                 cl, byte ptr [ecx + 0x7d0]
        //   52                   | push                edx
        //   51                   | push                ecx

    $sequence_1 = { ff15???????? 85c0 7407 50 ff15???????? 6a01 }
        // n = 6, score = 800
        //   ff15????????         |
        //   85c0                 | test                eax, eax
        //   7407                 | je                  9
        //   50                   | push                eax
        //   ff15????????         |
        //   6a01                 | push                1

    $sequence_2 = { 6a01 50 56 e8???????? 83c410 8bc7 }
        // n = 6, score = 800
        //   6a01                 | push                1
        //   50                   | push                eax
        //   56                   | push                esi
        //   e8????????           |
        //   83c410               | add                 esp, 0x10
        //   8bc7                 | mov                 eax, edi

    $sequence_3 = { 6a00 ff15???????? 85c0 7407 50 ff15???????? 6a01 }
        // n = 7, score = 800
        //   6a00                 | push                0
        //   ff15????????         |
        //   85c0                 | test                eax, eax
        //   7407                 | je                  9
        //   50                   | push                eax
        //   ff15????????         |
        //   6a01                 | push                1

    $sequence_4 = { 6a04 50 6a04 68???????? 68???????? }
        // n = 5, score = 800
```

```
            //   6a04                | push                4
            //   50                  | push                eax
            //   6a04                | push                4
            //   68????????          |
            //   68????????          |

        $sequence_5 = { 56 57 68da070000 e8???????? }
            // n = 4, score = 800
            //   56                  | push                esi
            //   57                  | push                edi
            //   68da070000          | push                0x7da
            //   e8????????          |

        $sequence_6 = { 50 56 e8???????? 83c410 8b4618 }
            // n = 5, score = 800
            //   50                  | push                eax
            //   56                  | push                esi
            //   e8????????          |
            //   83c410              | add                 esp, 0x10
            //   8b4618              | mov                 eax, dword ptr [esi + 0x18]

        $sequence_7 = { 8b4d08 668b91d2070000 8a89d0070000 52 51 }
            // n = 5, score = 800
            //   8b4d08              | mov                 ecx, dword ptr [ebp + 8]
            //   668b91d2070000      | mov                 dx, word ptr [ecx + 0x7d2]
            //   8a89d0070000        | mov                 cl, byte ptr [ecx + 0x7d0]
            //   52                  | push                edx
            //   51                  | push                ecx

        $sequence_8 = { 53 56 57 68da070000 }
            // n = 4, score = 800
            //   53                  | push                ebx
            //   56                  | push                esi
            //   57                  | push                edi
            //   68da070000          | push                0x7da

        $sequence_9 = { 8b4d08 668b91d2070000 8a89d0070000 52 }
            // n = 4, score = 800
            //   8b4d08              | mov                 ecx, dword ptr [ebp + 8]
            //   668b91d2070000      | mov                 dx, word ptr [ecx + 0x7d2]
            //   8a89d0070000        | mov                 cl, byte ptr [ecx + 0x7d0]
            //   52                  | push                edx

    condition:
        7 of them and filesize < 770048
}
```

By Cyber Threat Hunters from MSSPLab:

- @cocomelonc
- @wqkasper

Thanks for your time happy hacking and good bye!
All drawings and screenshots are MSSPLab's
