Advanced Brazilian Malware Analysis

reversingminds-blog.logdown.com/posts/7807545-analysis-of-advanced-brazilian-banker-malware

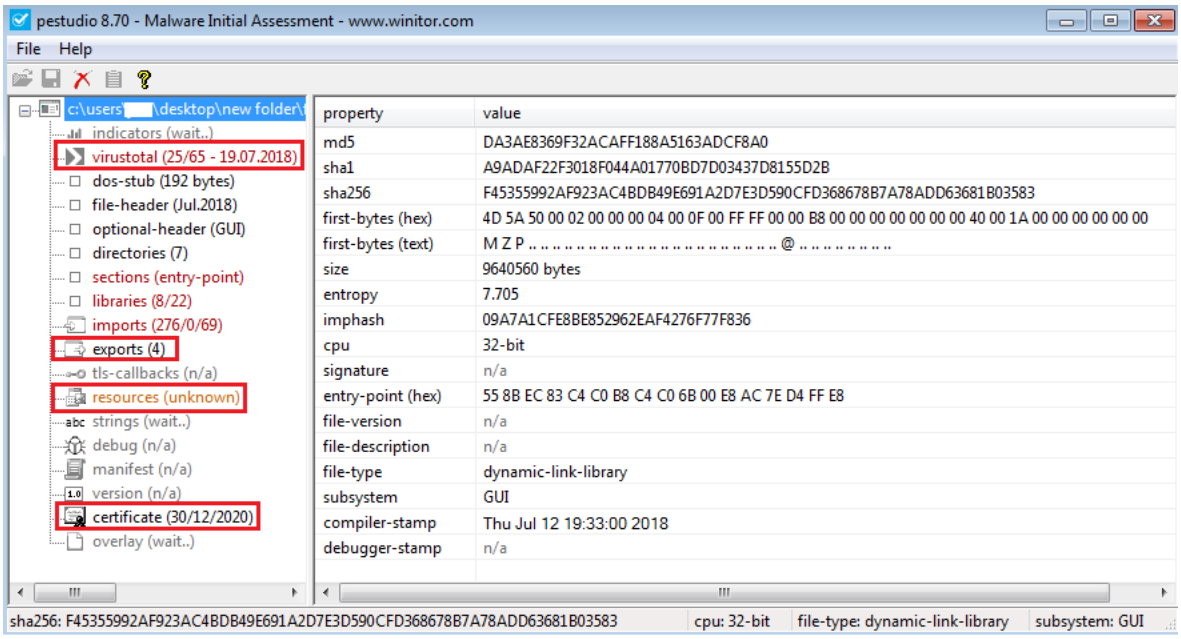
almost 4 years ago

Malware analysis sample with MD5 da3ae8369f32acaff188a5163adcf8a0

There is no info about how the sample infects the system, this sample could have been dropped/downloaded in an initial stage infection. For example, a common scenario, where an user receives an email with a malicious Word document attached. This word document could have **malicious Macros** or an exploit that takes advantage of **CVE-2017-11882** or **CVE-2018-0802** vulnerabilities in order to download, stablish the persistence and execute the second stage.

Moreover, the sample is a **DLL** and it is possible that it uses **DllHijacking** to be more stealthy during its execution, setting an autorun mecanism using the registry, scheduled task, service... pointing to the legitimate program that is going to load de malicious DLL.

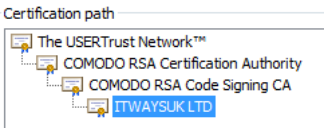
Static analysis - MD5 da3ae8369f32acaff188a5163adcf8a0



The sample is a **DLL compiled with Delphi**, the **Compiler timestamp:** is 12/07/2018 (it can be modified), with a **"valid certificate"** signed by **ITWAYSUK LTD**.

VT Detections

The sample is detected by **33/65 antivirus engines**, as "**Trojan.Banker**"



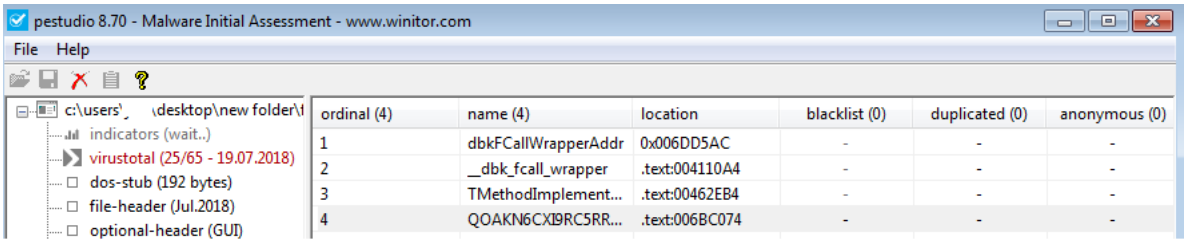
File information

Identification | Details | Content | Analyses | Submissions | ITW | Comments

2018-07-21 20:36:23 **33/65**
2018-07-19 08:11:51 **25/65**
2018-07-18 07:05:40 **21/65**
2018-07-16 20:02:45 **12/65**

McAfee	-	6.0.6.653	20180719
McAfee-GW-Edition	-	v2017.3010	20180719
Microsoft	Trojan:Win32/Tiggre!plock	1.1.15000.2	20180719
MicroWorld-eScan	-	14.0.297.0	20180719
NANO-Antivirus	Trojan.Win32.Banker1.ffkiky	1.0.116.23366	20180719
Paloalto	generic.ml	1.0	20180719
Panda	-	4.6.4.2	20180718
Qihoo-360	-	1.0.0.1120	20180719
Rising	Spyware.Mekotio!8.F5DF (TFE:dGZlOgXgSX/c2LZ/TA)	25.0.0.24	20180719
SentinelOne	-	1.0.17.227	20180701
Sophos	-	4.98.0	20180719

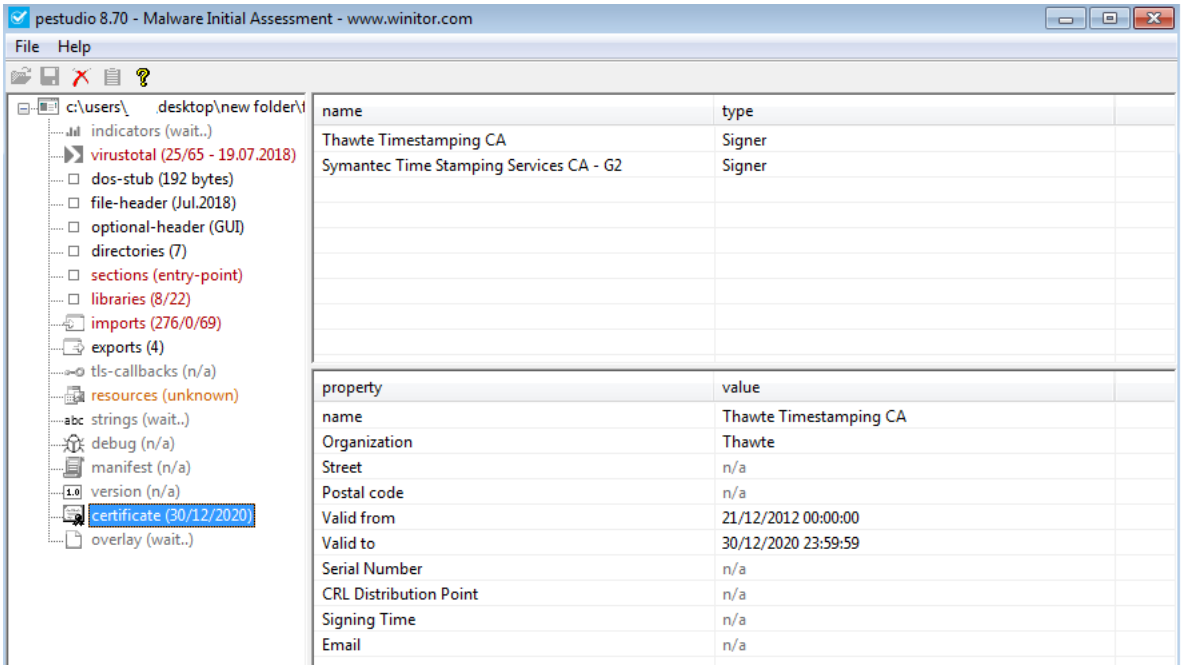
Exports



The sample has 4 exports, **dbkFCallWrapperAddr**, ***_dbk_fcall*wrapper** and **TMethodImplementationIntercept** exports are usual in **DLLs** compiled in **Delphi XE6**.

QOAKN6CXI9RC5RRTFXN13SHVYHD9KOR4SP is the export that performs the malicious operations.

Certificate



The signer is **ITWAYSUK LTD**, and it looks valid, malware developers uses stolen certificates in order to sign malware to bypass **Microsoft SmartScreen Application Reputation engine**.

```
#### PE SIGNATURE

Serial Number            : 23 4f 65 60 e6 7b 93 d4 45 86 21 7b e3 e7 49 52
Signers                  : ITWAYSUK LTD; COMODO RSA Code Signing CA; COMODO SECURE™
Counter signers          : Symantec Time Stamping Services Signer - G4; Symantec Time Stamping Services CA - G2; Thawte
Timestamping CA
```

Dynamic Analysis - Export QOAKN6CXI9RC5RRTFXN13SHVYHD9KOR4SP

The analysis starts executing the export **QOAKN6CXI9RC5RRTFXN13SHVYHD9KOR4SP**.

Mutex creation

```
339C55F821DC21D8012F20B87EA348F623
```

```
EIP .text:020BC004 _Check_Mutex proc near               ; CODE XREF: QOAKN6CXI9RC5RRTFXN13SHVYHD9KOR4SP+3↓p
    .text:020BC004 push    ebp
    .text:020BC005 mov     ebp, esp
    .text:020BC007 push    offset Name                  ; "339C55F821DC21D8012F20B87EA348F623"
    .text:020BC00C push    0                            ; int
    .text:020BC00E push    0                            ; lpMutexAttributes
    .text:020BC010 call    _CreateMutexW
    .text:020BC015 call    GetLastError_1
    .text:020BC01A cmp     eax, 0B7h
    .text:020BC01F jnz     short loc_20BC028
    .text:020BC021 xor     eax, eax
    .text:020BC023 call    ___Runtime_error_MessageBox_ops
    .text:020BC028 ; ---------------------------------------------------------------------------
    .text:020BC028
    .text:020BC028 loc_20BC028:                         ; CODE XREF: _Check_Mutex+1B↑j
    .text:020BC028 pop     ebp
    .text:020BC029 retn
    .text:020BC029  Check Mutex endp
```

Then uses **FindResourceW** in order to obtain the resource **Y8LNCZ6BLW**:

```
020BBD80 mov     [ebp+var_8], eax
020BBD83 push    offset aY8lncz6blw ; "Y8LNCZ6BLW"
020BBD88 push    0Ah
020BBD8A mov     ecx, ds:hmod
020BBD90 mov     dl, 1
020BBD92 mov     eax, ds:off_1E9059C
020BBD97 call    __FindResourceW_and_IAT_IF1_Switch
020BBD9C mov     [ebp+var_C], eax
```

This resource is read when the execution starts, could be part of the configuration of the malicious sample.

```
01EB3292 mov     eax, [ebp+lpType]
01EB3295 push    eax                 ; lpType
01EB3296 mov     eax, [ebp+lpName]
01EB3299 push    eax                 ; lpName
01EB329A push    esi                 ; hModule
01EB329B call    FindResourceW
01EB32A0 mov     edi, eax
01EB32A2 mov     [ebx+10h], edi
01EB32A5 test    edi, edi
01EB32A7 jnz     short loc_1EB32B0
urceW_and_IAT_Infernal_Switch+16 (Synchronized wit
8B E5 5D  XT+··+T¦¦+·dp[ïs]
00 00 00  +...¦...····....
00 42 00  Y.8.L.N.C.Z.6.B.
FF FF FF  L.W.....¦...····
```

Check infection

It creates a file in **C:\Users\[USER]\AppData\Local** with the name "**rundll.exe.txt**". If the file exists the system have been infected before.

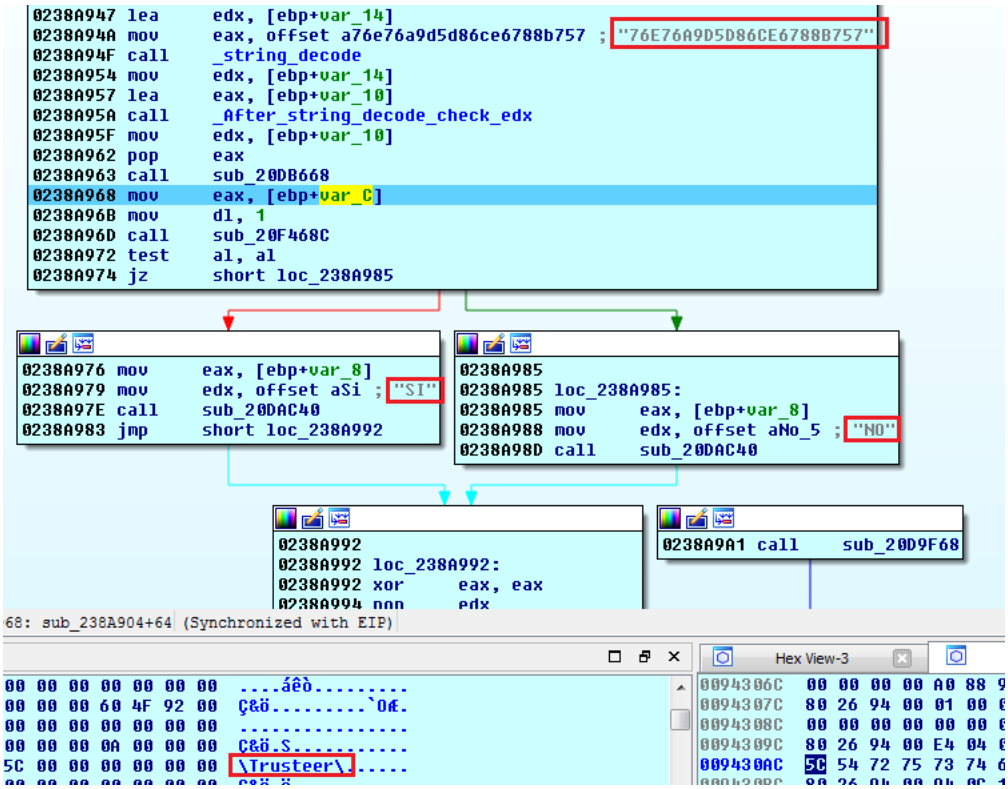
It uses de name of the process that launch the DLL to create the name of the file:

```
[NameOfTheProcess].exe.txt
```

In addition, checks if **testyy.txt** file exist in the path **%ALLUSERSPROFILE%\testyy.txt**

Decoding strings on demand

This sample decodes the strings on demand.



This technique makes the extraction of the strings more difficult.

Strings deciphered:

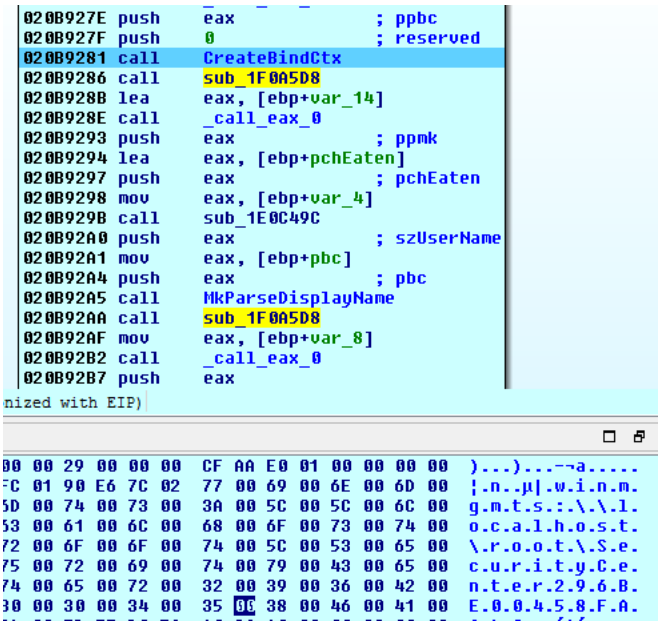
Ciphered
508DAE6287
7FC37BA858E96384B3A0BF2FCD789ABD78D81BBC7CEB13BB7EC4769B499F4580AF609DD96BE00458FA64
1C4D2818082A9658
0B52F322C66DD06197B14DA945ED21D618BF75DB758DB61222A246F92675EA5482AA5EE91BB440BC2EDE0C
6DF73DE34AC60227D0699D36CB74A65D
A8DB1DCE72933DF71BCD024533C6699A5CEB0848E166
48BB7DAE52F35D97BA6DA32B1D283E180B46E2c
0851F223C76AD56492BA5492AD5487B076DD17B95B81A72FC60020C156AA70D50E37E16F91CF13B711C97992E20C26D87DA43F84B6B6
76E76A9D5D86CE6788B757
E57BB96F919DF87B9BB0578BB95E9043E76FCE0022BCA633C11026CB0849FA5F82B46DE56D9B9B3D9C42F7207BBC7FA5
65F41E2C3CC62A38CE59F677B65F9046EB639A3DD41FC50D24A54AEF5895B92122D3024DE66CB139AE689A4387A2A4
DB1AD47BA9405AC87DA04E84B1568BB270E6
589E51F328DB74E60E3CE2658E3E9354F823D0033923C47FDF11C6
B5C959E662E2100F08
185BF60815007BEB
99C559E66A
0B6CFB13160B78EF7B80FE
9DD873899E99D243
1C68E867
1A7EE257EA1844E86FAC53BF4539C27B8FCE67FC3AA354F27FE66CE456AD8EC551FF73

Ciphered
09638E43CA46ACA043E9719ABB63E47D9A33CC1514B1A9E37BFF03372641C01DCA55CF1CD96DF058F01F291EBCA682B23EE978D05C
3B9BA85983BA28D97FA04F
2FA450FF2DCA1ADA0DCD7EC3719F4E
34A351FB22D170B0649F4286
D37DB95A86AF26D4073F
0959EC1CDE0240F628D67FC2678DBB679D29D80E31A557
78EF32D8073E95598AA650
2FA450FF2DCA1BD30425C90E39EE20C964E80B5D8F30DE77BE1E28D872D36F
CA18D77A91548D5382BC68FA38
A5D26291BA7CF509C46B9E30F5
61F60E3DEE085F9D48E10443F9
C21EDB0823D3013CDB063B975E8FB569
D3092EDA0ED40337E108
1542F228C77DD16890B8
67E20A36D5194AE411CE77D96DA549ED36AF
D40235E20821B346F61534AA41E80B
36AD5986A74192558DA34B89
6FE4103FEC0A5AEC1431D9758C
F450E91DDF17B74EFF3DEC6182A65B
D40235E20821A051FB3DDD7FB55E91
98CF6794B97FDC0227
78EC0628C361E4153AE306
F0669141EA0C4BE30335E47DA764
D30332E10B2CA143F314C5073EE40F38F46E89
499E5685A640975482BF61F136DC7F

WMI using monikers

The technique that the malicious sample uses to execute the **WMI** commands, is through the use of monikers (https://docs.microsoft.com/en-us/windows/desktop/wmisdk/constructing-a-moniker-string)

Using **CreateBindCtx** in order to create a bind context, then the function **MkParseDisplayName** will be called to create the moniker and finally, **BindToObject** will be called to execute, in this case **winmgmts:\localhost\root\SecurityCenter2** to obtain info about the antivirus, antispyware and firewall.



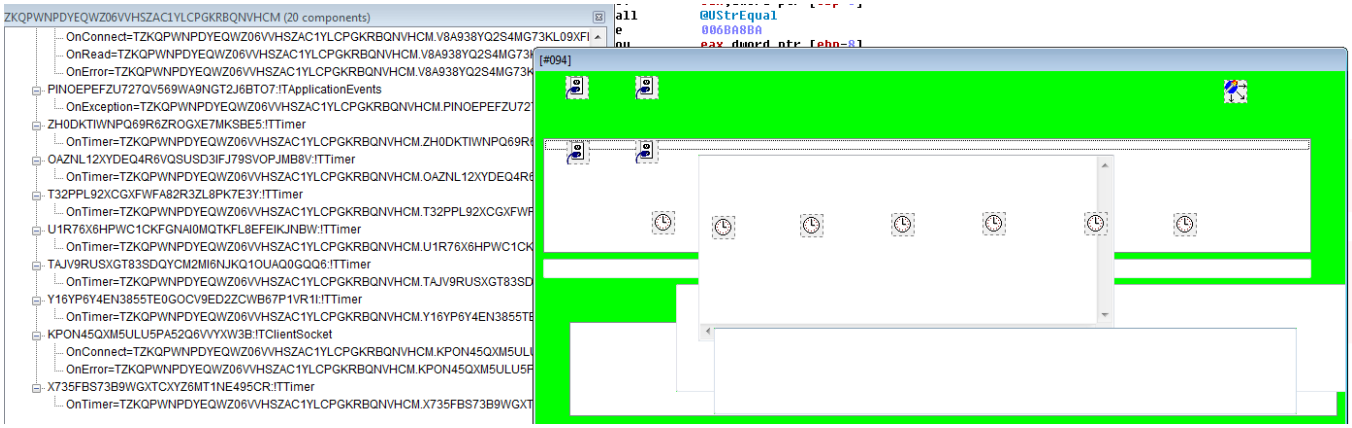
With this interface the malicious sample has the ability to use WMI.

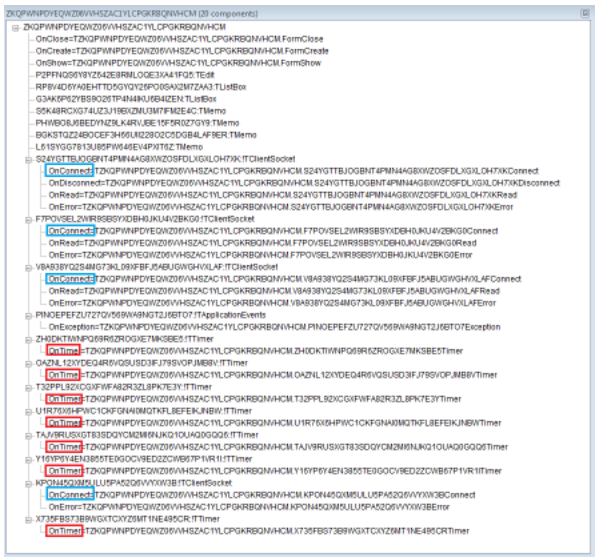
Some of the commands executed with this method:

Command	Description
winmgmts:\localhost\root\SecurityCenter2	WMI Service to obtain info about installed security products.
SELECT * FROM AntivirusProduct	Info about Antivirus (Enabled or Disabled)
hnetcfg.fwmgr	WMI Firewall Service to obtain info and control the firewall policy

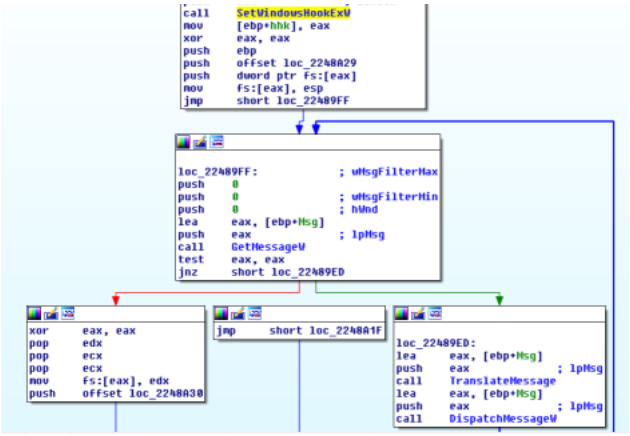
Core

The core funcionality of this malicious sample is performed using **callback functions:**

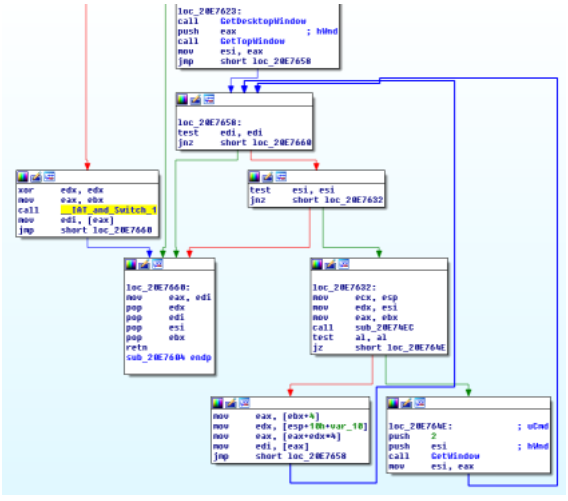




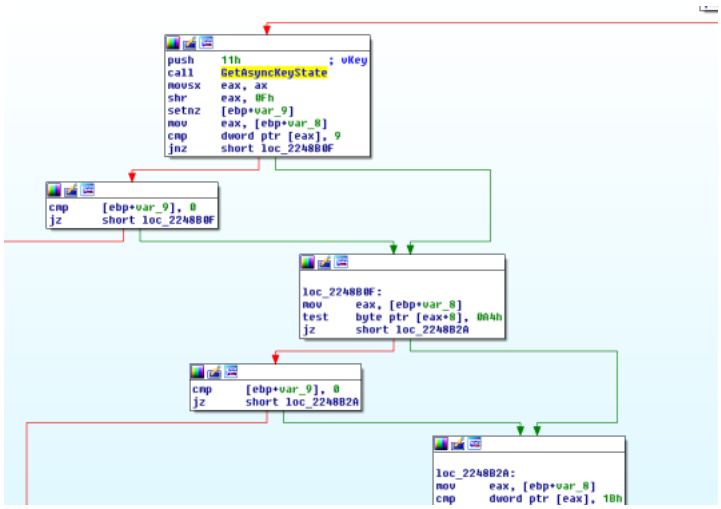
These callbacks are implemented using **timers** and **Windows Hooks** (using **SetWindowsHookEx**)



The different functions of the callbacks obtain info about the state and position of the windows, using functions like **GetDesktopWindow**, **GetTopWindow**, **GetWindow**



In addition, the functions check the keys introduced by the user using **GetAsyncKeyState** function:

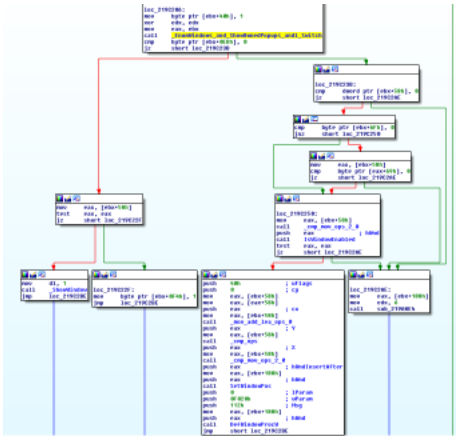


If it detects that the user is trying to transfer money using the web browser from one of the Chilean banks that monitors, it will steal the user credentials in order to perform money transfers to another account.

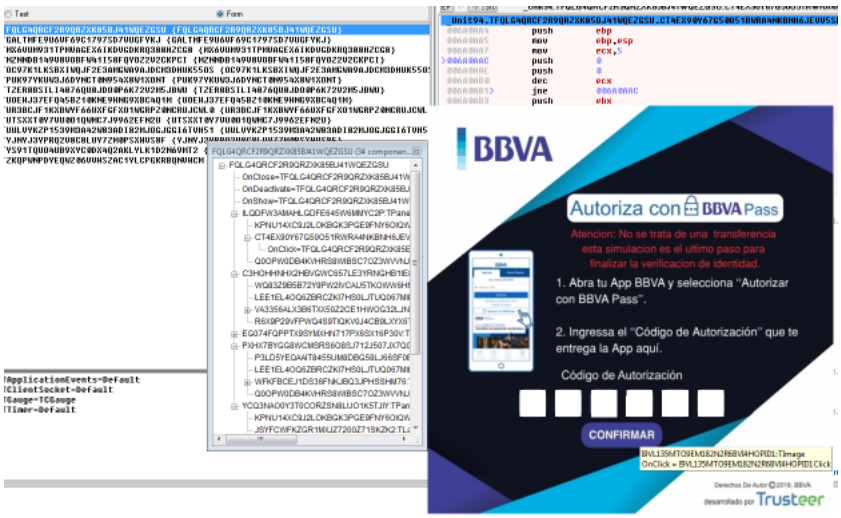
In addition and due to the use of the second factor is nowadays something common, a way to obtain the second factor code using social engineering has been implemented.

The technique consist in show a popup asking the user for the second factor code.

Bank steal info (second factor)



There are several forms to accomplish this task.



As it can be seen, there are specific forms in order to make the scam more effective.

BBVA bank is not the only one affected by this malicious sample, in the next section it is possible to find the affected financial institutions.

Affected Financial Institutions:

Looking the images:

- BBVA Chile
- Santander Chile
- Banco de Chile
- BCI
- CORPBANCA
- BancoEstado
- Scotiabank
- Itaú
- BancoBice
- BancoSecurity
- Banco Internacional Chile

Looking strings (deciphered):

- Scotiabank
- Banco Consorcio
- Banco Bci
- Transbank S.A
- HSBCnet
- BTG Pactual
- Banco Bice
- Banco Edwards
- Banco Condell
- Banco Ripley
- BBVA Chile
- Banco Falabella
- Banco Estado

- Banco de Chile
- Banco Santander Chile
- Banco Itaú (bancoita)
- Banco Security
- Banco Internacional (Chile)
- Banco CorpBanca

Clipboard cryptohijacking

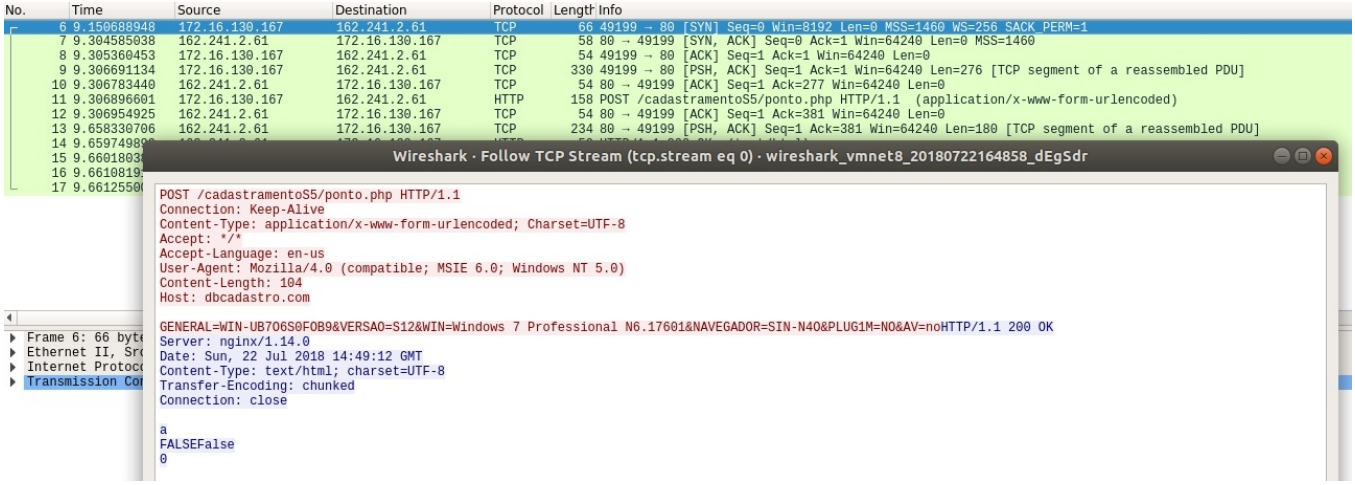
Another characteristic of this malicious sample, is the ability to detect and change the user BTC address from the clipboard by the hardcoded in the malicious sample, with the objective to trick the user to send the funds to the cybercriminal BTC address.

BTC address used by this malicious sample:

```
163McXwBrc9S7JzbgegzVuw7QTJ9H1dQj7
```

Communication

The sample establishes communication with the domain **hxxp://dbcadastro.com/cadastramentoS5/ponto.php** (IP 162.241.2.61), the server returns **"a FALSEfalse 0"**.



The message sent by the malicious sample contains the following info:

Command	Description
GENERAL=[SystemName]	System name
VERSAO=[Maybe the malware version]	Maybe the malware version
WIN=[Windows OS Version]	Windows OS version
NAVEGADOR=[WebBrowserInfo]	Web browser info
PLUG1M=[SI\	NO]
AV=[SI\	NO]

Conclusion

Brazilian banker malware by their tactics, techniques and procedures (TTPs) targeting Chileans financial instituions, in order to steal, bank credentials using fake bank pop-ups. It seems more advanced than the common Brazilian malware, using certificates, on demand string deciphering and WMI monikers. Moreover implements clipboard cryptohijacking techniques.

Malicious sample characteristics and capabilities:

- TTPs related to Brazilian malware (Delphi)
- Chileans bank affectation.
- Use of WMI monikers to obtain info and manipulate the antivirus, firewall etc...
- Uses valid certificate to bypass Microsoft SmartScreen.
- Ciphered strings, only deciphered on demand.
- Steals bank credentials and double factor tricking the user, using fake popups requesting for second factor code.
- Clipboard BTC address cryptohijacking.
- Keylogger features

IOCs

Process with this mutex:

```
339C55F821DC21D8012F20B87EA348F623
```

C&C - Domains and IPs

```
http://dbcadastro.com/cadastramentoS5/ponto.php
162.241.2.61
190.114.253.206 (Dynamic DNS)
```

Domains from VT passive DNS related to 162.241.2.61

```
2018-07-21 impressoscapao.com.br
2018-07-21 www.impressoscapao.com.br
2018-07-21 www.wonderdesigngrafico.com
2018-07-21 wonderdesigngrafico.com
2018-07-20 targetnegocios.com.br
2018-07-20 www.lilianecassinelli.com
2018-07-20 lilianecassinelli.com
2018-07-20 treinalinux.com
2018-07-20 www.treinalinux.com
2018-07-20 www.flordelizbrechoinfantil.com.br
2018-07-20 flordelizbrechoinfantil.com.br
2018-07-20 www.cfnow.com.br
2018-07-20 cfnow.com.br
2018-07-20 www.rcstylebrasilia.com.br
2018-07-20 rcstylebrasilia.com.br
2018-07-20 www.bepersonalstyle.com.br
2018-07-20 bepersonalstyle.com.br
```

Domains from VT passive DNS related to 190.114.253.206 (Dynamic DNS)

```
2018-04-05 ssl.ddns.me
2018-01-19 ssl2018.brasilia.me
```

BTC Address used for clipboard cryptohijacking

```
163McXwBrc9S7JzbgegzVuw7QTJ9H1dQj7
```

Bitcoin Address Addresses are identifiers which you use to send bitcoins to another person.

Summary	
Address	163McXwBrc9S7JzbgegzVuw7QTJ9H1dQj7
Hash 160	374aaadb74912c38bb38fcc50581893b568f7672

Transactions	
No. Transactions	9
Total Received	0.13497611 BTC
Final Balance	0 BTC

Request Payment | Donation Button

Transactions (Oldest First)

Filter

8542357c93f7d314f78767b7798ff46e3406d791a5dac14763f0b3245eeb5d7b			2018-07-07 18:23:42
163McXwBrc9S7JzbgegzVuw7QTJ9H1dQj7	→	1DVQES79cV2wvrPr6gncCXrzoY86D2PJbB	0.08563777 BTC
			-0.0836792 BTC
b1fba705107cdcc35768488330e9db74a82f8dd5ca3bd7ca267389c98ff4a929			2018-07-07 00:48:05
1NuZcmnf9kFZnYHjj9vtcd35Rx7x24vH6b	→	163McXwBrc9S7JzbgegzVuw7QTJ9H1dQj7	0.0836792 BTC
			0.0836792 BTC
53a07d6850119b9ef6462c0239d402c374fb16fefc61b101f89ff0709019e525			2018-05-13 00:26:58
163McXwBrc9S7JzbgegzVuw7QTJ9H1dQj7	→	17Gc8dQARfSmoTuFX3dw1Jda4K98ZjvpiZ	0.1673 BTC
			-0.00697943 BTC

PE Signature

```
#### PE SIGNATURE

Serial Number             : 23 4f 65 60 e6 7b 93 d4 45 86 21 7b e3 e7 49 52
Signers                   : ITWAYSUK LTD; COMODO RSA Code Signing CA; COMODO SECURE™
Counter signers           : Symantec Time Stamping Services Signer - G4; Symantec Time Stamping Services CA - G2; Thawte
Timestamping CA
```

Snort Rule

```
alert tcp any any -> any any (msg:”BrazilianBanker”;
content:"POST";
http_method;
pcre:"GENERAL=.*&VERSAO=.*&WIN=.*&NAVEGADOR=.*&PLUG1M=.*&AV=.*";)
```

Generic Yara Rule

```
rule GenericBrazilianBankerRule {
   meta:
      description = "GenericBrazilianBankerRule"
      date = "2018-07-22"
      hash1 = "7acb19b31a431ba3ca05acff9c1b378eb1658585761ff84ca762e2b5f16098d0"
      hash2 = "d0e232ac6602d7c09e5ee233fb5865c1b9286e90973058665e0c76917a50c95d"
      hash3 = "b04e2037771923747ff98afb6cfd1d6769b6d266f781e66ebe7d92fd43e7b92b"
      hash4 = "e133c2134604d5ae038583f848df13cc8ab42be77e25554d78a938f2ff078437"
      hash5 = "6aaf83ff0b2deda98eb39150ff90c47b4a5f5f78d1eb0f185017ede95bfdedf5"
      hash6 = "97e10f669c1094838f3814e8f850d8cf9479db3a8b5fc7fe2bcde1edf1977dfa"
      hash7 = "985c6d89543563901c131c5cc7143f680fd957b8aa74c0f5dd25f7e462e354e9"
      hash8 = "1eef0649b231f9ce0838f1a04658ad4f6c8563b9ad7358397db09d21ac744a52"
      hash9 = "e65948d6caefa012741edda1f9f99b56abfed5e66d101178cd846679717c6b29"
      hash10 = "1386c27973e299b5fb07bb6ad065e02c6bcac5d2b29da15a068be9f6d29dfa30"
      hash11 = "4a18f1e284fd06ef9b96bdc4b0b1666810f4868fc8d1edfbaf46727dfce416eb"
      hash12 = "425a4bfcb429761781550117cc95f4cb3778279bb9535caeb3824e086593faa1"
      hash13 = "6e82426990e76b0847fd0446c54c92a0b5833f65aee2d135fea183c67badd944"
      hash14 = "ba1ccda1cb3e73b95f75b014abb5d078510b3ad71fde21164494714d9bb776e9"
      hash15 = "bc5662a336871a35ead6522dd17a83861338cf354b4baa62a672ffdc111c9f96"
      hash16 = "f233313327906be03487c3f20732f5cf8f8e1150d19a9dc30e68e9db2d85a0ca"
      hash17 = "19ea016096b35c7af9a4b7b4f586070e3203f4b91be329d26783c6b1f3ec8346"
      hash18 = "007cb339f4314da51f34f46d51adb9537229750e6112f4cf192db872042793b6"
      hash19 = "a8dcda65baf611c2a7c35a129eb6903c779e45a90f91d4515db5d4af72bbebf5"
      hash20 = "067fb3fbfaee68e825af3b184bf61b7997ed3b0a1cf833aecba40b00d00fcb04"
      hash21 = "0dca0f585bb175dc5b248cbf2d326516477361999 99d3af533f917e009ea9f11"
      hash22 = "77426ec69ec283fb561022e32c37768da6ae08e5ea24bb8aae134af971ded426"
      hash23 = "b96a43bfbf8a03b019fb3cd82834576b23299d12ebe985ea19684829b6be22ce"
      hash24 = "94c69d61e769cbd0229af67461749121f02d067ec4c1b1d14f95f35af2576243"
      hash25 = "f65a4dfeef0a7b5d539ab889d8badf0100017ebe11a9cb784882813ddbf3a00c"
      hash26 = "8418c5026cc9a1656859bac2c5f504561c76559d5ebcbdf3c0a7a74cf3b4458c"
      hash27 = "033489ac01edb282a139a19058fb746db01f62c5c70bf49cc34e5cc35130cd4b"
      hash28 = "4ac058056fa965b6a9ae5efa8a4af44827952ae3c64af9352250f48eeefda0a2"
      hash29 = "60a4801983780a0b2b971bfe906e8ab2204323538ce964ddc093ed493136177b"
      hash30 = "30ddce8086742c014e3a796c4406262a0696bdf8703cf0996a32f8fa27449c2a"
      hash31 = "a617e7d9c5066ad2e125297257f92fcf1ba2106adde60571799c07c0ce55f96f"
      hash32 = "f45355992af923ac4bdb49e691a2d7e3d590cfd368678b7a78add63681b03583"
      hash33 = "a4c94417cb5f33054feee449de342d6afb7c1836194259af3b5048d3e06cf4c3"
      hash34 = "f2f645c0864cf536c9461339633a3ede4bd9f58dead05d10fafbd18429bba206"
      hash35 = "2b6b1c4de97695c278348f3e34e274f3ba6328a210f9e2dbe3035c9eff595cfd"
      hash36 = "78ded77048f73d94a6bee27ef9a229c2e07de616732b8ce0d990f38a158e5ada"
      hash37 = "a02e430146b555b68db882aeb26a02fcc044bcb27c23f2dc80d579e41775b7d5"
      hash38 = "3ae06cbc3ae679d9eb19a03277c9c87258c4607fd3c561523afae89742b6895e"
      hash39 = "f00724324df876d30b5b708301c4179b67f39927276e9c9a17d24e769d5b8152"
      hash40 = "b101e15184908561673ebc20ca4464789d363bdc8f5a3d54f4ca127fac57e100"
      hash41 = "7e5c63d2b9287f31a7679055f9172ec449b230663573296db502954c9677ab3b"
      hash42 = "a3626c6cd21e1be1ed25bc1939aa9922a7aaa4bc2ffdd1bfbd9952bb3c357a97"
      hash43 = "543da2c0830049b84d8e6667d05f802cf1a3f65eaccc96b5efb4c17538f233bd"
   strings:
      $s1 = "Invalid characters in path The specified file was not found*Windows socket error: %s (%d), on API '%s'" fullword wide
      $s2 = "SelectedNotFocusedHot$tlGroupHeaderLineCloseMixedSelection'tlGroupHeaderLineCloseMixedSelectionHot" fullword ascii
      $s3 = "C:\\ProgramData\\testyy.txt" fullword wide
      $s4 = "ExecuteMacroLines" fullword ascii
      $s5 = "ctedNotFocusedHot#tlGroupHeaderLineOpenMixedSelection&tlGroupHeaderLineOpenMixedSelectionHot" fullword ascii
      $s6 = "MTDelegatedComparer<System.Rtti.TPair<System.TypInfo.PTypeInfo,System.string>>" fullword ascii
      $s7 = "FOnExecuteMacro" fullword ascii
      $s8 = "OnExecuteMacro" fullword ascii
      $s9 = "ExecuteMacro" fullword ascii
      $s10 = "?TDelegatedComparer<System.Rtti.TMethodImplementation.TParamLoc>" fullword ascii
      $s11 = "System.Win.ScktComp" fullword ascii
      $s12 = " - Host: " fullword wide
      $s13 = "OnGetPassword" fullword ascii
      $s14 = "4TDelegatedComparer<System.HelpIntfs.THelpViewerNode>" fullword ascii
      $s15 = "WSAASyncGetHostByName" fullword wide
      $s16 = "Error setting %s.Count8Listbox (%s) style must be virtual in order to set Count%Cannot remove shell notification
icon\"%s requir" wide
      $s17 = "CTDictionary<System.string,System.TypInfo.PTypeInfo>.TPairEnumerator" fullword ascii
      $s18 = "FGetHostData" fullword ascii
      $s19 = "~System.Win.ScktComp" fullword ascii
      $s20 = "GetCookieByNameAndDomain" fullword ascii
   condition:
      ( uint16(0) == 0x5a4d and
        filesize < 28000KB and ( 8 of them )
      ) or ( all of them )
}
```

Malicious samples from VirusTotal with the same PE Signature

```
007cb339f4314da51f34f46d51adb9537229750e6112f4cf192db872042793b6
033489ac01edb282a139a19058fb746db01f62c5c70bf49cc34e5cc35130cd4b
067fb3fbfaee68e825af3b184bf61b7997ed3b0a1cf833aecba40b00d00fcb04
0dca0f585bb175dc5b248cbf2d32651647736199999d3af533f917e009ea9f11
1386c27973e299b5fb07bb6ad065e02c6bcac5d2b29da15a068be9f6d29dfa30
19ea016096b35c7af9a4b7b4f586070e3203f4b91be329d26783c6b1f3ec8346
1eef0649b231f9ce0838f1a04658ad4f6c8563b9ad7358397db09d21ac744a52
2b6b1c4de97695c278348f3e34e274f3ba6328a210f9e2dbe3035c9eff595cfd
30ddce8086742c014e3a796c4406262a0696bdf8703cf0996a32f8fa27449c2a
35db30e5b9577d52ddedb3ac0fc402b0a49c406c3d231330bebfc52ecba2a828
3ae06cbc3ae679d9eb19a03277c9c87258c4607fd3c561523afae89742b6895e
425a4bfcb429761781550117cc95f4cb3778279bb9535caeb3824e086593faa1
4a18f1e284fd06ef9b96bdc4b0b1666810f4868fc8d1edfbaf46727dfce416eb
4ac058056fa965b6a9ae5efa8a4af44827952ae3c64af9352250f48eeefda0a2
543da2c0830049b84d8e6667d05f802cf1a3f65eaccc96b5efb4c17538f233bd
60a4801983780a0b2b971bfe906e8ab2204323538ce964ddc093ed493136177b
6aaf83ff0b2deda98eb39150ff90c47b4a5f5f78d1eb0f185017ede95bfdedf5
6e82426990e76b0847fd0446c54c92a0b5833f65aee2d135fea183c67badd944
77426ec69ec283fb561022e32c37768da6ae08e5ea24bb8aae134af971ded426
78ded77048f73d94a6bee27ef9a229c2e07de616732b8ce0d990f38a158e5ada
7acb19b31a431ba3ca05acff9c1b378eb1658585761ff84ca762e2b5f16098d0
7e5c63d2b9287f31a7679055f9172ec449b230663573296db502954c9677ab3b
8418c5026cc9a1656859bac2c5f504561c76559d5ebcbdf3c0a7a74cf3b4458c
94c69d61e769cbd0229af67461749121f02d067ec4c1b1d14f95f35af2576243
97e10f669c1094838f3814e8f850d8cf9479db3a8b5fc7fe2bcde1edf1977dfa
985c6d89543563901c131c5cc7143f680fd957b8aa74c0f5dd25f7e462e354e9
a02e430146b555b68db882aeb26a02fcc044bcb27c23f2dc80d579e41775b7d5
a3626c6cd21e1be1ed25bc1939aa9922a7aaa4bc2ffdd1bfbd9952bb3c357a97
a4c94417cb5f33054feee449de342d6afb7c1836194259af3b5048d3e06cf4c3
a617e7d9c5066ad2e125297257f92fcf1ba2106adde60571799c07c0ce55f96f
a8dcda65baf611c2a7c35a129eb6903c779e45a90f91d4515db5d4af72bbebf5
b04e2037771923747ff98afb6cfd1d6769b6d266f781e66ebe7d92fd43e7b92b
b101e15184908561673ebc20ca4464789d363bdc8f5a3d54f4ca127fac57e100
b96a43bfbf8a03b019fb3cd82834576b23299d12ebe985ea19684829b6be22ce
ba1ccda1cb3e73b95f75b014abb5d078510b3ad71fde21164494714d9bb776e9
bc5662a336871a35ead6522dd17a83861338cf354b4baa62a672ffdc111c9f96
c02bb98803992ee6b1babb86f1b165bad99dd1eecdb01fab00cebe45cbae5860
d0e232ac6602d7c09e5ee233fb5865c1b9286e90973058665e0c76917a50c95d
e133c2134604d5ae038583f848df13cc8ab42be77e25554d78a938f2ff078437
e65948d6caefa012741edda1f9f99b56abfed5e66d101178cd846679717c6b29
f00724324df876d30b5b708301c4179b67f39927276e9c9a17d24e769d5b8152
f233313327906be03487c3f20732f5cf8f8e1150d19a9dc30e68e9db2d85a0ca
f2f645c0864cf536c9461339633a3ede4bd9f58dead05d10fafbd18429bba206
f45355992af923ac4bdb49e691a2d7e3d590cfd368678b7a78add63681b03583
f65a4dfeef0a7b5d539ab889d8badf0100017ebe11a9cb784882813ddbf3a00c
```

Author: @51ddh4r7h4

- August 20, 2018 11:34
- Permalink