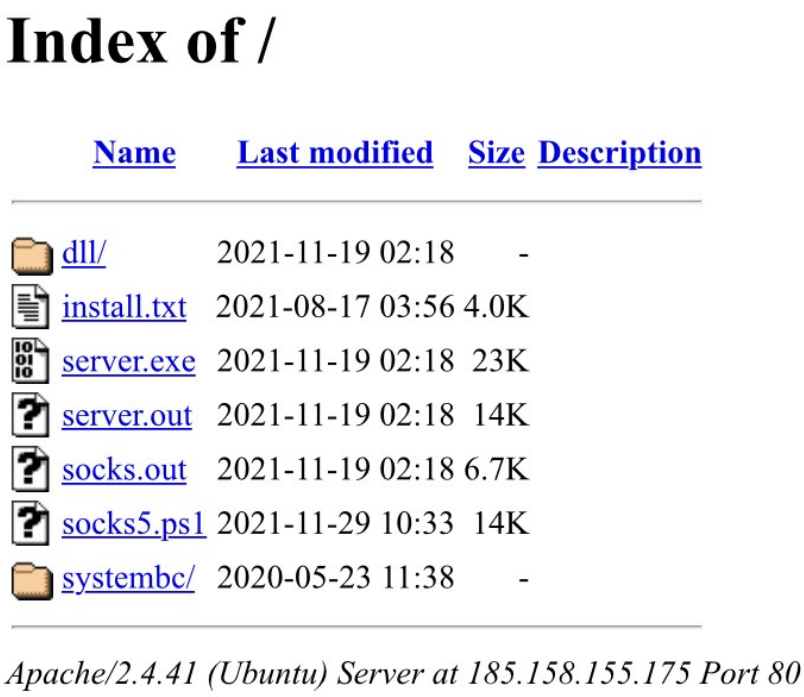
SystemBC, PowerShell version

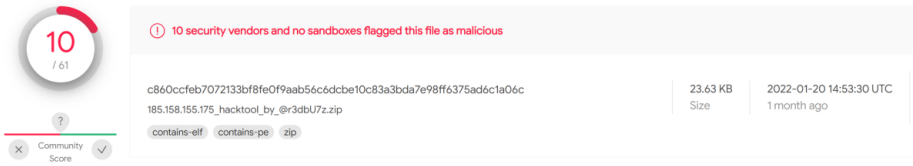
Jason Reaves ⋮⋮ 3/4/2022

By: Jason Reaves and Joshua Platt

Some of the most effective malware leveraged over the past few years against enterprise environments has incorporated scripting. AV detections for script based malware have historically lagged behind those of binary based detections. The SystemBC Malware-as-a-Service we previously outlined[1], has been leveraged by prolific crimeware groups involved in ransomware operations against enterprises[1,3,4,5] for a while now. Earlier this year a researcher on twitter[2] found and uploaded a copy of an open directory containing a SystemBC package containing the elements of a SystemBC package along with an interesting powershell file:

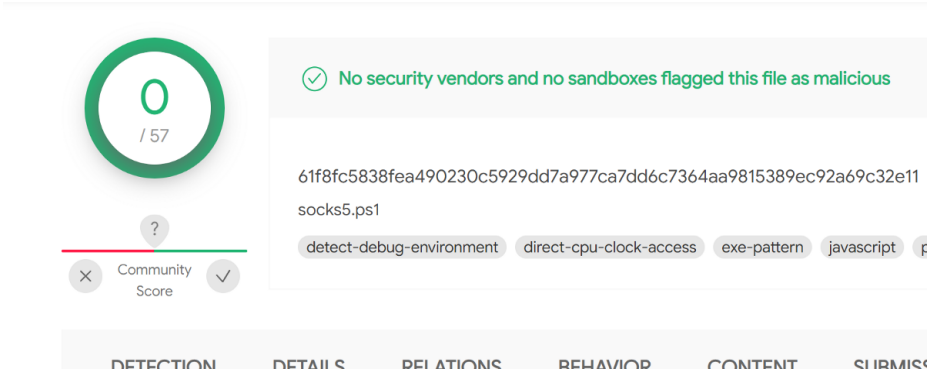


The uploaded packaged can be found on VirusTotal:



Ref:

The PowerShell script ‘socks5.ps1’ has no detections:



Ref:

The powershell script has a header containing a C2 server and a port number to connect to before then setting up a block of 50 bytes called 'xordata' which will be later passed to the 'Rc4_crypt' function

```
$xordata = New-Object byte[] 50For ($i=0; $i -ne 50; $i++) { $xordata[$i] =  $i }
```

Using a traffic example from VirusTotal:

```
00000000  00 01 02 03 04 05 06 07  08 09 0a 0b 0c 0d 0e 0f   ........ ........
00000010  10 11 12 13 14 15 16 17  18 19 1a 1b 1c 1d 1e 1f   ........ ........
00000020  20 21 22 23 24 25 26 27  28 29 2a 2b 2c 2d 2e 2f    !"#$%&' ()*+,-./
00000030  30 31 82 2e d9 e1 18 f7  d1 66 21 f6 9b ae 10 32   01...... .f!....2
00000040  21 57 6f 07 58 08 75 e0  33 86 43 b2 9c c5 97 e5   !Wo.X.u. 3.C.....
00000050  07 fa aa 90 c8 72 6b 0e  e6 08 14 4b 18 e4 b5 7f   .....rk. ...K....
00000060  a5 64 f1 3a                                       .d.:
```

Decrypting:

```
>>> a =
'000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f202122232425262728292a2b2c2d2e2f303182
 import binascii>>> b = binascii.unhexlify(a)>>>
b'\x00\x01\x02\x03\x04\x05\x06\x07\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a

!"#$%&\'()*+,-./01\x82.\xd9\xe1\x18\xf7\xd1f!\xf6\x9b\xae\x102!Wo\x07X\x08u\xe03\x86C\xb2\x9c\xc5\x97\xe
 from Crypto.Cipher import ARC4>>> len(b)100>>>
b[:50]'\x00\x01\x02\x03\x04\x05\x06\x07\x08\t\n\x0b\x0c\r\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x1
 !"#$%&\'()*+,-./01'>>> rc4 = ARC4.new(b[:50])>>>
rc4.decrypt(b[50:])'\xb1\x1d\x00\x01PS\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
```

The first word is the build number for Windows:

```
$osn = [system.environment]::osversion.version.build


$os0 = $osn -band 0x000000ff
$os1 = [math]::Floor(($osn -band 0x0000ff00) * [math]::Pow(2,-8))

$buffer0[50] = $os0 -as[byte]
$buffer0[51] = $os1 -as[byte]
```

In our decrypted example this is '7601', the next word value is bit check:

```
$int64 = 0


if ([IntPtr]::Size -eq 8) {$int64 = 1}

$buffer0[53] = $int64 -as[byte]
```

The PS value is hardcoded:

```
$buffer0[54] = 0x50 -as[byte]                $buffer0[55] = 0x53 -
as[byte]
```

After checking in, the bot receives IPs and port numbers and each one is assigned to their own job in a pool thread which will handle proxying traffic.

```
[void]$ps.AddScript($new_connection)[void]$ps.AddParameter("stream", $stream)
[void]$ps.AddParameter("writer", $writer)[void]$ps.AddParameter("reader", $reader)
[void]$ps.AddParameter("SocketArray", $SocketArray)[void]$ps.AddParameter("ebx", $ebx)
[void]$ps.AddParameter("domain", $domain)
[void]$ps.AddParameter("port_", $port_)             [void]$ps.AddParameter("xordata_",
$xordata)          [void]$ps.AddParameter("Rc4_crypt", $Rc4_crypt)    $jobs[$i] =
[PSCustomObject]@{        PowerShell = $ps        AsyncResult = $ps.BeginInvoke()    }
```

With the current method chosen by the developer (to hardcode the key generation), we can assume this version is still in a developmental stage. This makes network and endpoint detections easier for the time being.

IOCs

Powershell version:

c860ccfeb7072133bf8fe0f9aab56c6dcbe10c83a3bda7e98ff6375ad6c1a06c

185.158.155[.]175

SystemBC Full C2 list:

185.61.138.59172.106.86.12sweetcloud.linkasdfghjkl.hostbitdesk.onlineordercouldhost.comhcwakententx2.cor
socat01.xyztvtmhltd.org5.132.191.105185.215.113.78179.43.178.96protoukt.comsocksbswfjhofnbu.onionadmex1
correios.com188.212.22.165arbetfrolli.pwreserveupdate.comstatistiktrafiktrubest.nettbueguicsrwo64i7.onic
networking.com74.125.46.143109.201.140.54verguliosar.comxxxxxxtnuhffpbep.onion185.193.91.23437.49.229.1
server.comtik-
tak.clubjjj.rop.devbljxlgj4h4yuxkju.onion45.141.87.6063bwf6zdrgsmagpt.onion92.63.197.143fragrant.digital
socks.cc139.60.161.5823hfdne.xyzbrabulco.ac.ug80.233.248.1094renewdmn.biz5.206.224.199ncordercreatetest
tak-super-
puper.xyz135.181.37.14493.187.129.249185.197.74.227lisnm.comscserv1.infos.avluboy.xyz217.8.117.65149.28
records.life185.191.32.191aitchchewcdn.online176.111.174.63ns1.vic.au.dns.opennic.gluejoiasbella.com.br
lab.comjlayxnzzin5y335h.onionzghiexdgwfzi44b5.onion84.38.129.162masonksmith.tech46.166.176.24737.1.204.
socat01.com45.153.186.2435.79.124.201fhaaaggs.ml176.123.8.226217.8.117.42adobeupd.hosthuxere.xyz37.1.22(

Detections

Endpoint:

```
Run key:"HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" - socks5_powershell
```

Network:

```
alert tcp $HOME_NET any -> $EXTERNAL_NET any ( msg:"SystemBC Powershell bot
registration"; dsize:100; content: "|00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 26 27 28 29 2a 2b 2c
2d 2e 2f 30 31|"; offset: 0; depth: 50; classtype:trojan-activity; sid:9000011;
rev:1;)
```

References

1: https://medium.com/walmartglobaltech/inside-the-systembc-malware-as-a-service-9aa03afd09c6

2: https://twitter.com/r3dbU7z

3: https://news.sophos.com/en-us/2020/10/14/inside-a-new-ryuk-ransomware-attack/

4: https://twitter.com/vk_intel/status/1234891766924484609?lang=en

5: https://blogs.blackberry.com/en/2021/06/threat-thursday-systembc-a-rat-in-the-pipeline