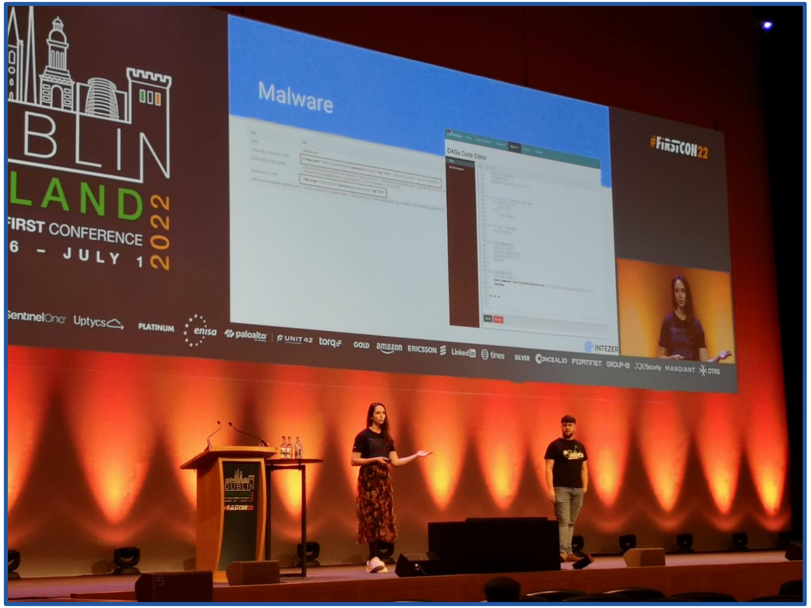
Unpacking Packers - So What? Does it ever get easier? No.

Nicole Fishbein

INTEZER

About Me

- Senior Security Researcher and Malware Analyst @ Intezer.
- Researcher of embedded devices for 5 years.
- Presented at BlackHat USA, BlackHat EU, FIRSTCon Dublin, LABScon, BSidesSF, RECON and more.



Agenda

1. Crypters, loaders and packers
2. Why crypters are used by malware developers?
3. How do we unpack packed* files?
    a. Static analysis
    b. Emulator
    c. Sandbox
4. Can we solve it with AI?
5. Conclusions and reflections

Crypters, Loaders and Packers

01

INTEZER

Crypters, Loaders, Packers...

Let's define them

🔒Crypters

- Crypters are tools used to encrypt or obfuscate code to avoid detection by antivirus software.
- Examples:
  1. CyberSeal - Known for encrypting executable files to evade antivirus detection.
  2. Armadillo Crypter - Uses multiple layers of encryption for payload protection.

🚀 Loaders

- Loaders are programs designed to load and execute payloads into memory, often used as the first stage of an attack to bypass security controls.
- Examples:

  1. IcedID Loader – Primarily used to load banking Trojans.
  2. Smoke Loader - A lightweight tool to inject and execute malware in a target system.

📦 Packers

- Packers compress executable files to make them harder to detect and reverse-engineer, often used to hide malware's true size and structure.
- Examples:

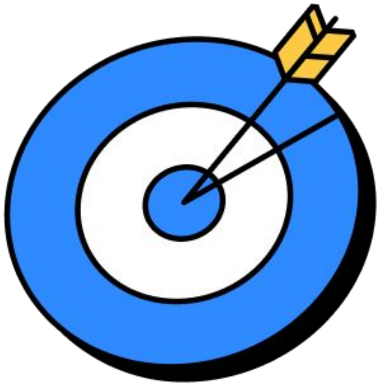
  1. UPX (Ultimate Packer for eXecutables) - Popular for compressing executable files.
  2. ASPack - Often used to compress and obfuscate executables.

Why Do Malware Developers Use Crypters, Loaders, and Packers?

02

Crypters in Malware Distribution

- To make the analysis more complicated
- To evade detection
- To prolong malware lifespan
- To make attacks modular



Crypters in Malware Distribution

Banking Trojans & Info-Stealers

Emotet, TrickBot, Dridex.

Use crypters and packers to stay stealthy during credential harvesting.

RATs (Remote Access Trojans)

NjRAT, QuasarRAT.

Often delivered by loaders that decrypt the RAT in memory.

Ransomware

Conti, LockBit.

Use packers and loaders to bypass defenses and ensure initial execution.

Initial Access Brokers

These are groups that specialize in gaining entry (often using loaders) and then sell access to ransomware operators.

Stats?

How do we Unpack Files?

03

START A FIGHT

The Battlefield

Defenders

Static analysi

Emulators

Sandbox

Detect the attempt to evade detection!!

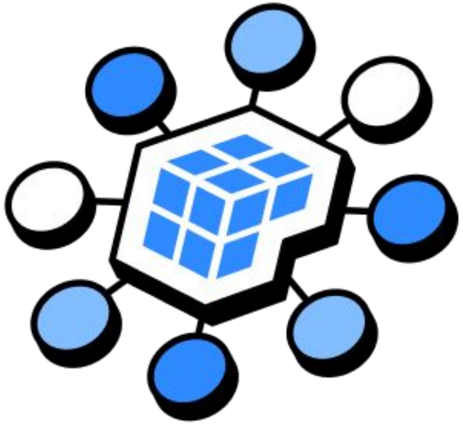
Threat Actors

ers, crypters

Detect and evade analysis

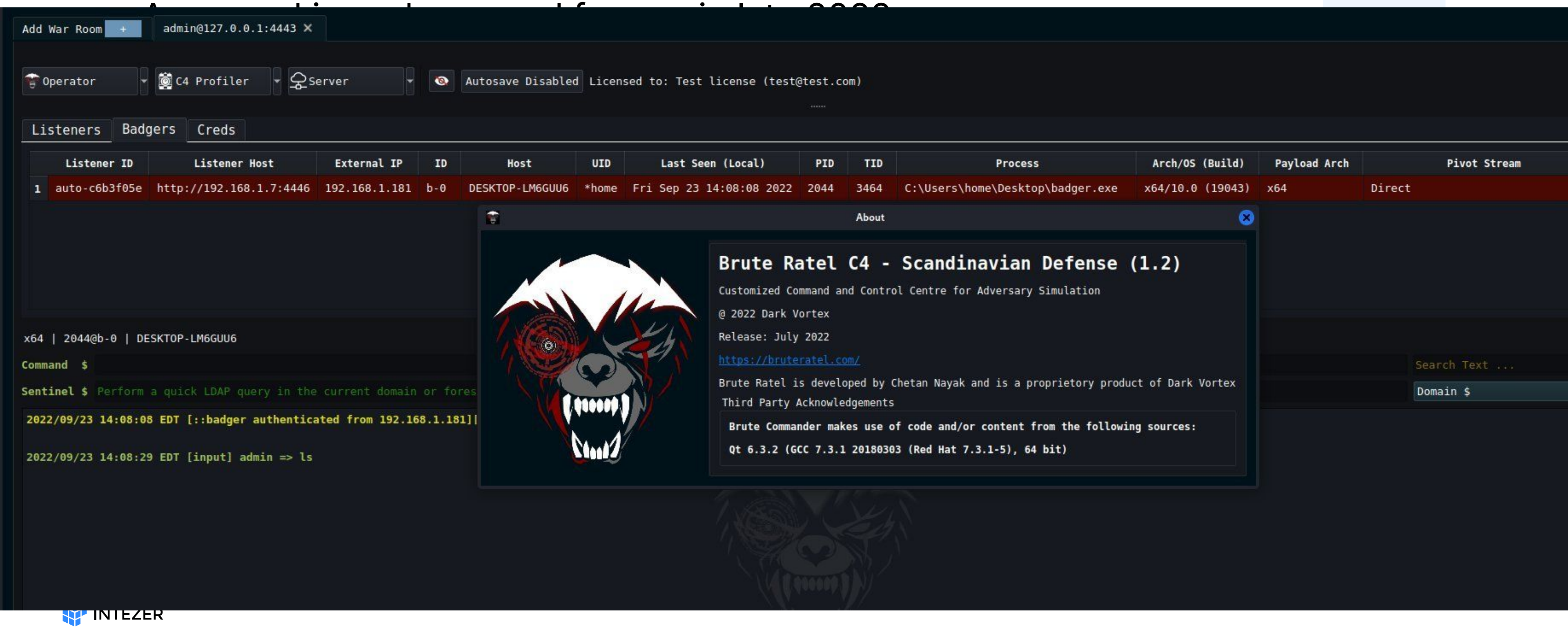
Static Extraction

Writing scripts or pieces of code to extract the payload using the knowledge we gained in the RE process



Brute Ratel?

- C2 framework similar to Cobalt Strike



Licensed to: Test license (test@test.com)

	Listener ID	Listener Host	External IP	ID	Host	UID	Last Seen (Local)	PID	TID	Process	Arch/OS (Build)	Payload Arch	Pivot Stream
1	auto-c6b3f05e	http://192.168.1.7:4446	192.168.1.181	b-0	DESKTOP-LM6GUU6	*home	Fri Sep 23 14:08:08 2022	2044	3464	C:\Users\home\Desktop\badger.exe	x64/10.0 (19043)	x64	Direct

Brute Ratel C4 - Scandinavian Defense (1.2)

Customized Command and Control Centre for Adversary Simulation

@ 2022 Dark Vortex

Release: July 2022

https://bruteratel.com/

Brute Ratel is developed by Chetan Nayak and is a proprietory product of Dark Vortex

Third Party Acknowledgements

Brute Commander makes use of code and/or content from the following sources:

Qt 6.3.2 (GCC 7.3.1 20180303 (Red Hat 7.3.1-5), 64 bit)

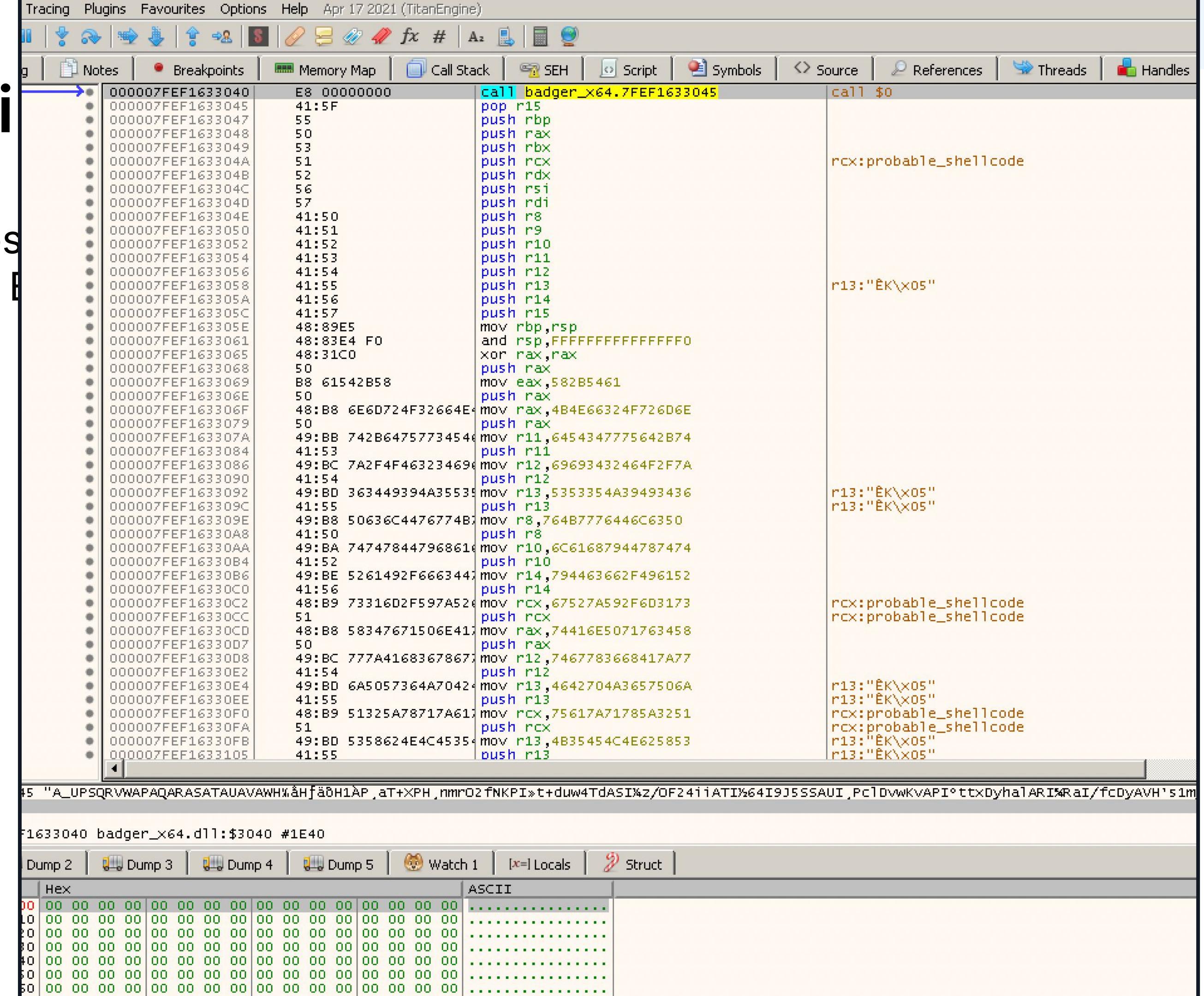
```
x64 | 2044@b-0 | DESKTOP-LM6GUU6
Command  $
Sentinel $ Perform a quick LDAP query in the current domain or fores
Domain $

2022/09/23 14:08:08 EDT [::badger authenticated from 192.168.1.181]|

2022/09/23 14:08:29 EDT [input] admin => ls
```

Analyzing Brute Ratel

- Bager = Beacon
- GUI = Commander
- Cracked version enables defenders build a badger payload
- Configuration and payload - similar to Cobalt Strike

Analyzi

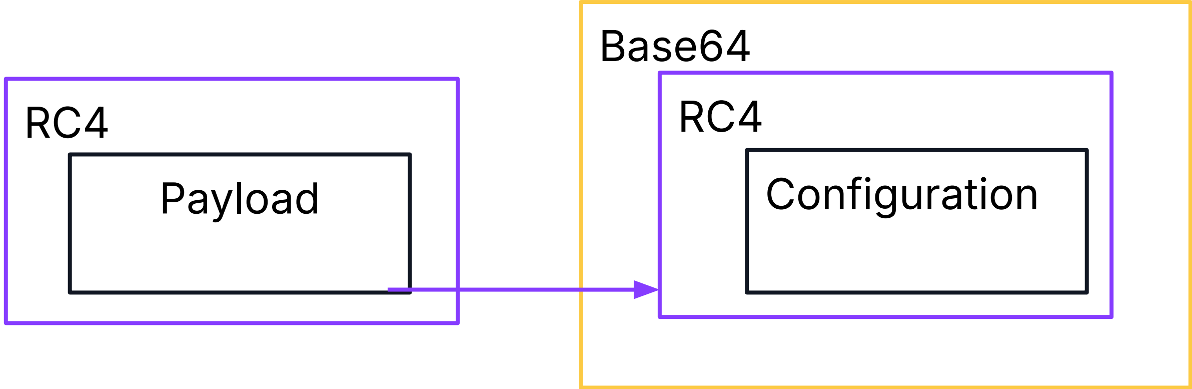
- A series
- Build a E


Tracing Plugins Favourites Options Help Apr 17 2021 (TitanEngine)
g Notes Breakpoints Memory Map Call Stack SEH Script Symbols Source References Threads Handles
Dump 2 Dump 3 Dump 4 Dump 5 Watch 1 [x=] Locals Struct
Hex
ASCII

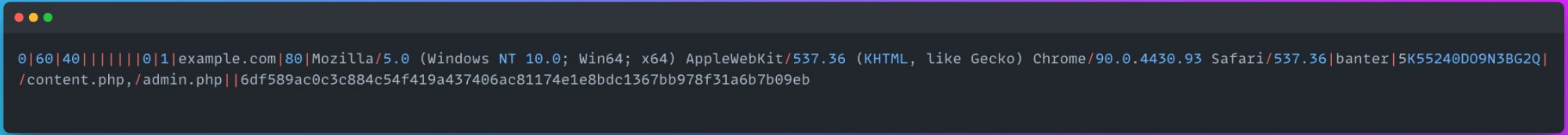
```
000007FEF1633040     E8 00000000          call badger_x64.7FEF1633045    call $0
000007FEF1633045     41:5F                pop r15
000007FEF1633047     55                   push rbp
000007FEF1633048     50                   push rax
000007FEF1633049     53                   push rbx
000007FEF163304A     51                   push rcx                       rcx:probable_shellcode
000007FEF163304B     52                   push rdx
000007FEF163304C     56                   push rsi
000007FEF163304D     57                   push rdi
000007FEF163304E     41:50                push r8
000007FEF1633050     41:51                push r9
000007FEF1633052     41:52                push r10
000007FEF1633054     41:53                push r11
000007FEF1633056     41:54                push r12
000007FEF1633058     41:55                push r13                       r13:"ÊK\x05"
000007FEF163305A     41:56                push r14
000007FEF163305C     41:57                push r15
000007FEF163305E     48:89E5              mov rbp,rsp
000007FEF1633061     48:83E4 F0           and rsp,FFFFFFFFFFFFFFF0
000007FEF1633065     48:31C0              xor rax,rax
000007FEF1633068     50                   push rax
000007FEF1633069     B8 61542B58          mov eax,582B5461
000007FEF163306E     50                   push rax
000007FEF163306F     48:B8 6E6D724F32664E mov rax,4B4E66324F726D6E
000007FEF1633079     50                   push rax
000007FEF163307A     49:BB 742B6475773454 mov r11,6454347775642B74
000007FEF1633084     41:53                push r11
000007FEF1633086     49:BC 7A2F4F46323469 mov r12,69693432464F2F7A
000007FEF1633090     41:54                push r12
000007FEF1633092     49:BD 363449394A3553 mov r13,5353354A39493436  r13:"ÊK\x05"
000007FEF163309C     41:55                push r13                       r13:"ÊK\x05"
000007FEF163309E     49:B8 50636C4476774B mov r8,764B7776446C6350
000007FEF16330A8     41:50                push r8
000007FEF16330AA     49:BA 74747844796861 mov r10,6C61687944787474
000007FEF16330B4     41:52                push r10
000007FEF16330B6     49:BE 5261492F666344 mov r14,794463662F496152
000007FEF16330C0     41:56                push r14
000007FEF16330C2     48:B9 73316D2F597A52 mov rcx,67527A592F6D3173  rcx:probable_shellcode
000007FEF16330CC     51                   push rcx                       rcx:probable_shellcode
000007FEF16330CD     48:B8 58347671506E41 mov rax,74416E5071763458
000007FEF16330D7     50                   push rax
000007FEF16330D8     49:BC 777A4168366786 mov r12,7467783668417A77
000007FEF16330E2     41:54                push r12
000007FEF16330E4     49:BD 6A5057364A7042 mov r13,4642704A3657506A  r13:"ÊK\x05"
000007FEF16330EE     41:55                push r13                       r13:"ÊK\x05"
000007FEF16330F0     48:B9 51325A78717A61 mov rcx,75617A71785A3251  rcx:probable_shellcode
000007FEF16330FA     51                   push rcx                       rcx:probable_shellcode
000007FEF16330FB     49:BD 5358624E4C4535 mov r13,4B35454C4E625853  r13:"ÊK\x05"
000007FEF1633105     41:55                push r13                       r13:"ÊK\x05"

45 "A_UPSQRVWAPAQARASATAUAVAWH‰åHƒäðH1ÀP¸aT+XPH¸nmrO2fNKPI»t+duw4TdASI¼z/OF24iiATI½64I9J5SSAUI¸PclDvwKvAPI°ttxDyhalARI¾RaI/fcDyAVH¹s1m

F1633040 badger_x64.dll:$3040 #1E40

00  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
10  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
20  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
30  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
40  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
50  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
60  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

INTEZER

Analyzing Brute Ratel - The Loader

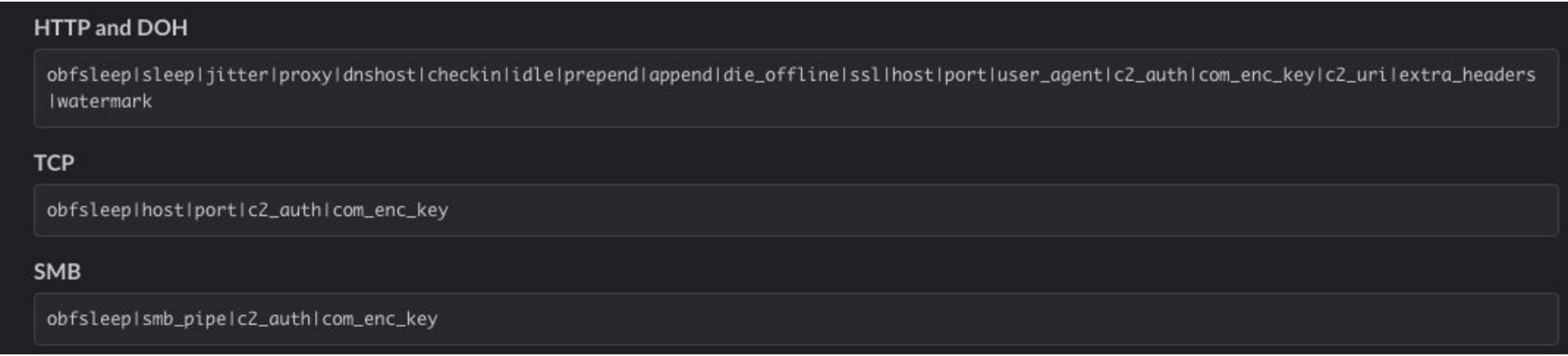


Decoded Configuration



```
0|60|40|||||||0|1|example.com|80|Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.93 Safari/537.36|banter|5K55240DO9N3BG2Q|
/content.php,/admin.php||6df589ac0c3c884c54f419a437406ac81174e1e8bdc1367bb978f31a6b7b09eb
```

The Structure of the Configuration



HTTP and DOH

```
obfsleep|sleep|jitter|proxy|dnshost|checkin|idle|prepend|append|die_offline|ssl|host|port|user_agent|c2_auth|com_enc_key|c2_uri|extra_headers
|watermark
```

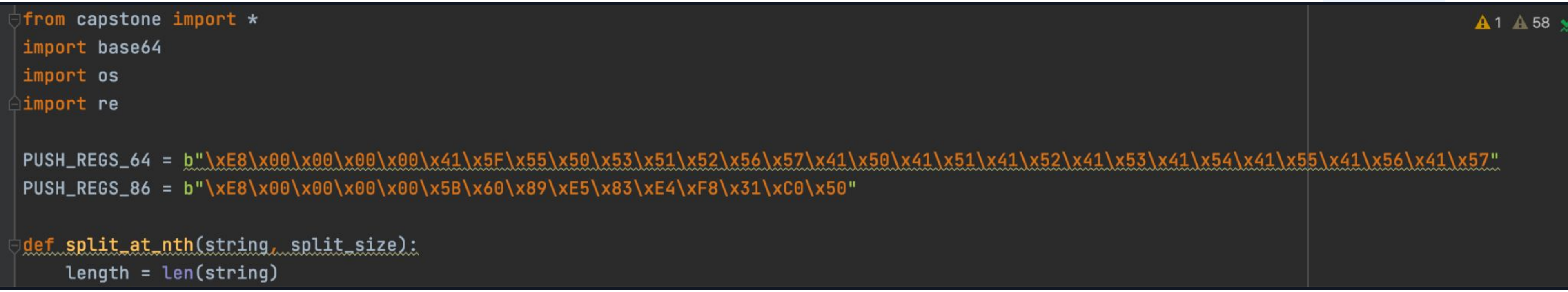
TCP

```
obfsleep|host|port|c2_auth|com_enc_key
```

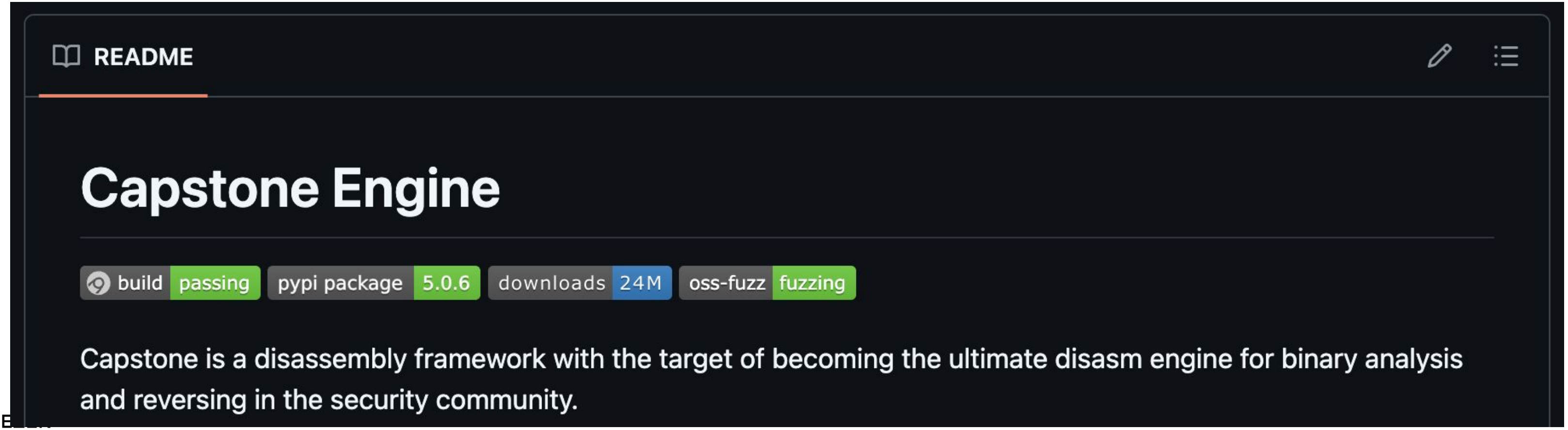
SMB

```
obfsleep|smb_pipe|c2_auth|com_enc_key
```

Static Extraction Script



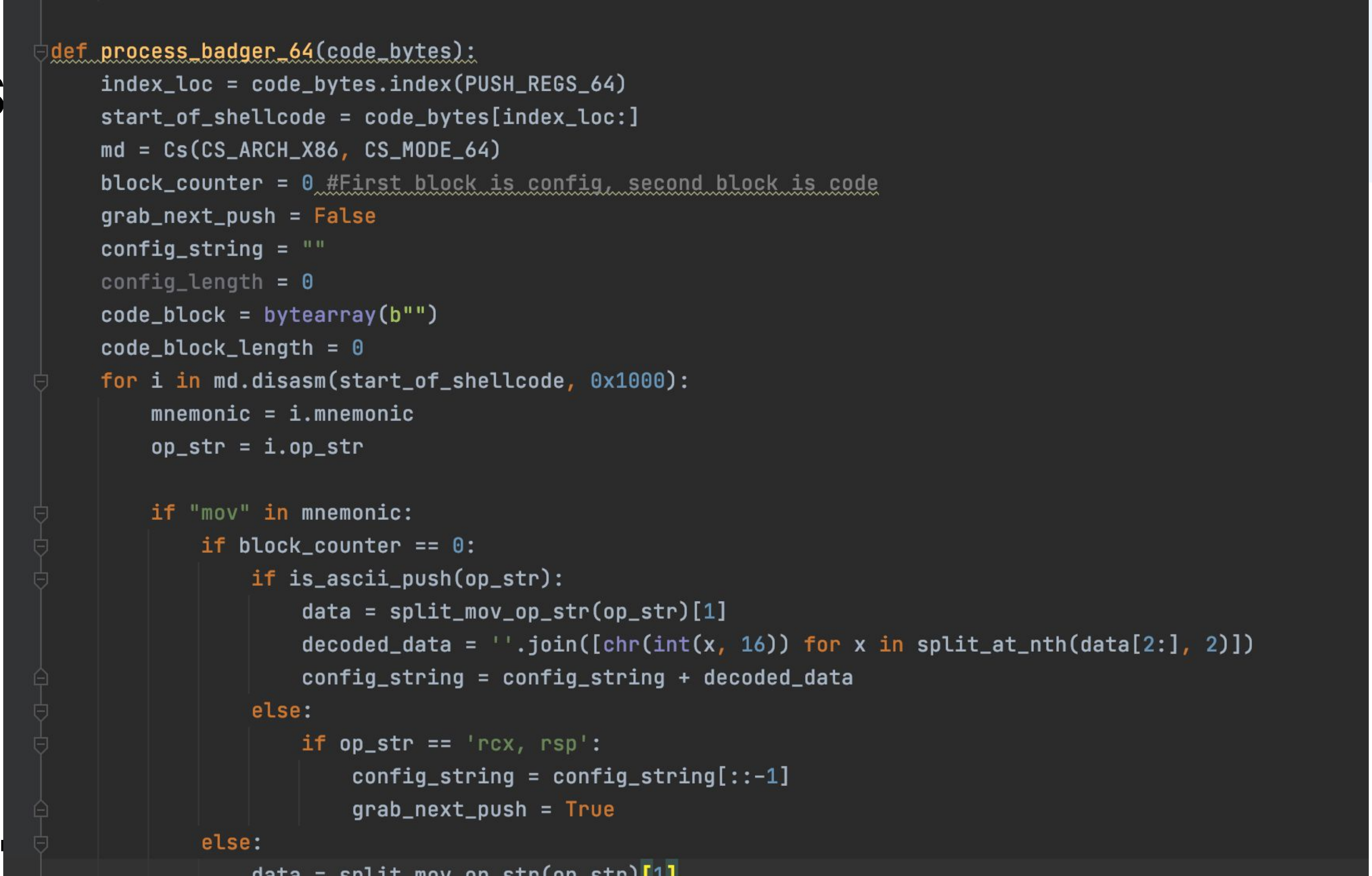
```
from capstone import *
import base64
import os
import re

PUSH_REGS_64 = b"\xE8\x00\x00\x00\x00\x41\x5F\x55\x50\x53\x51\x52\x56\x57\x41\x50\x41\x51\x41\x52\x41\x53\x41\x54\x41\x55\x41\x56\x41\x57"
PUSH_REGS_86 = b"\xE8\x00\x00\x00\x00\x5B\x60\x89\xE5\x83\xE4\xF8\x31\xC0\x50"

def split_at_nth(string, split_size):
    length = len(string)
```



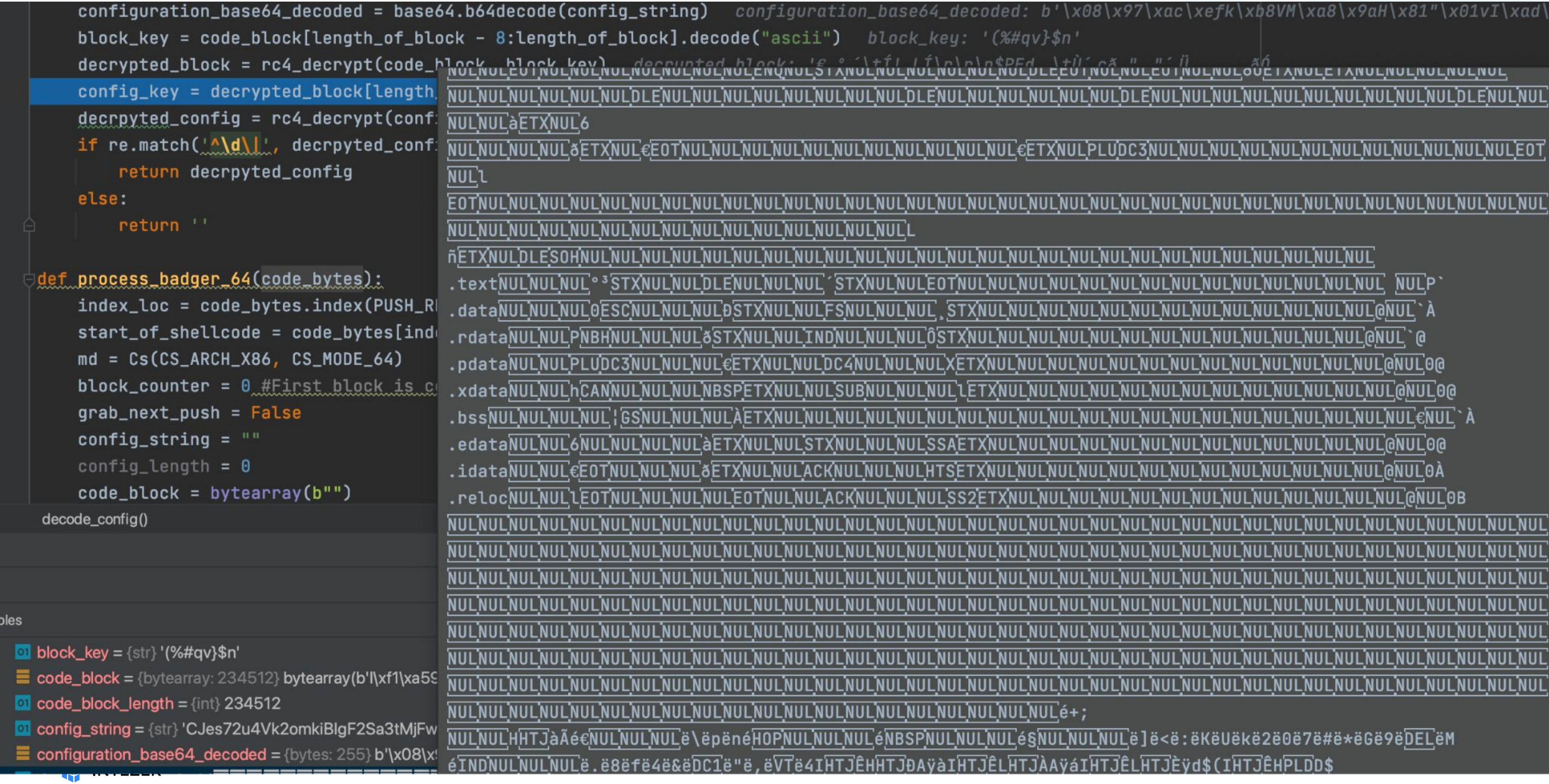
Capstone Engine

Capstone is a disassembly framework with the target of becoming the ultimate disasm engine for binary analysis and reversing in the security community.

S



```
def process_badger_64(code_bytes):
    index_loc = code_bytes.index(PUSH_REGS_64)
    start_of_shellcode = code_bytes[index_loc:]
    md = Cs(CS_ARCH_X86, CS_MODE_64)
    block_counter = 0 #First block is config, second block is code
    grab_next_push = False
    config_string = ""
    config_length = 0
    code_block = bytearray(b"")
    code_block_length = 0
    for i in md.disasm(start_of_shellcode, 0x1000):
        mnemonic = i.mnemonic
        op_str = i.op_str

        if "mov" in mnemonic:
            if block_counter == 0:
                if is_ascii_push(op_str):
                    data = split_mov_op_str(op_str)[1]
                    decoded_data = ''.join([chr(int(x, 16)) for x in split_at_nth(data[2:], 2)])
                    config_string = config_string + decoded_data
                else:
                    if op_str == 'rcx, rsp':
                        config_string = config_string[::-1]
                        grab_next_push = True
            else:
                data = split_mov_op_str(op_str)[1]
```

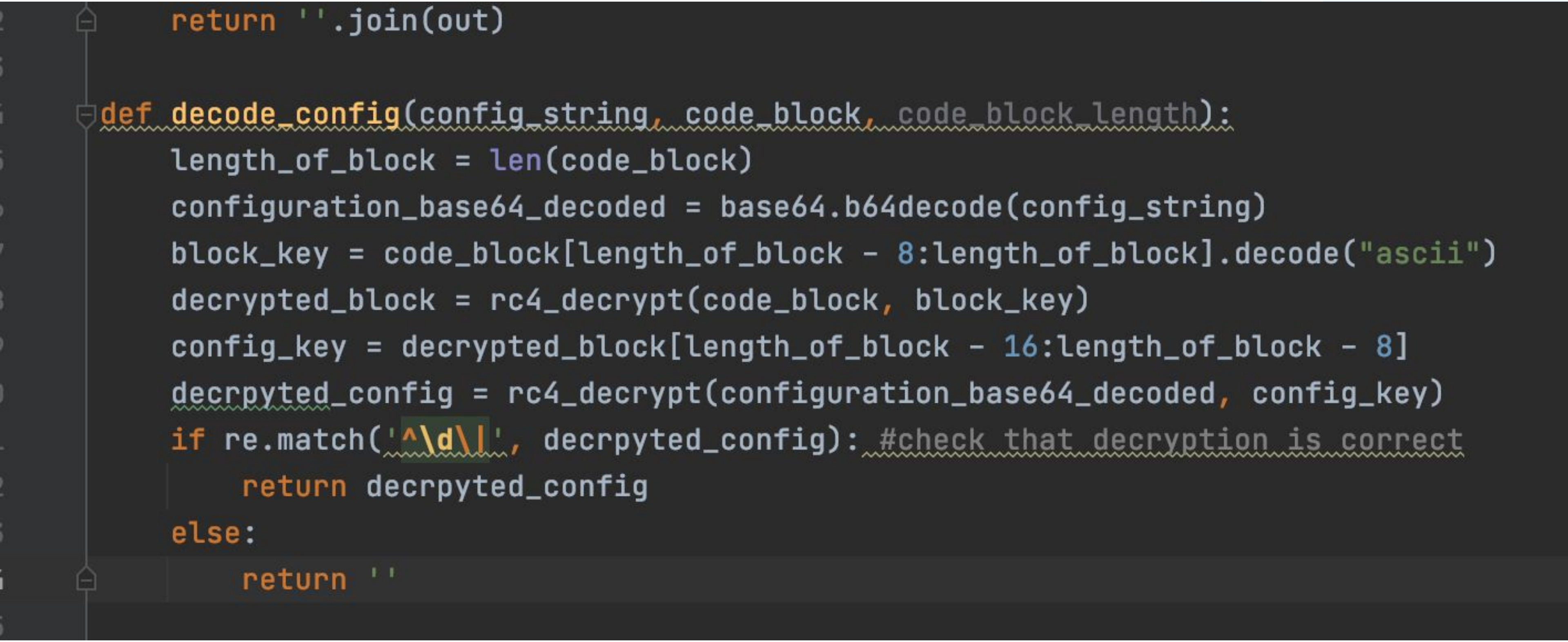


```
    configuration_base64_decoded = base64.b64decode(config_string)   configuration_base64_decoded: b'\x08\x97\xac\xefk\xb8VM\xa8\x9aH\x81"\x01vI\xad\
    block_key = code_block[length_of_block - 8:length_of_block].decode("ascii")   block_key: '(%#qv}$n'
    decrypted_block = rc4_decrypt(code_block, block_key)
    config_key = decrypted_block[length
    decrpyted_config = rc4_decrypt(conf
    if re.match('^\d\|', decrpyted_conf
        return decrpyted_config
    else:
        return ''

def process_badger_64(code_bytes):
    index_loc = code_bytes.index(PUSH_R
    start_of_shellcode = code_bytes[ind
    md = Cs(CS_ARCH_X86, CS_MODE_64)
    block_counter = 0 #First block is c
    grab_next_push = False
    config_string = ""
    config_length = 0
    code_block = bytearray(b"")
```

decode_config()

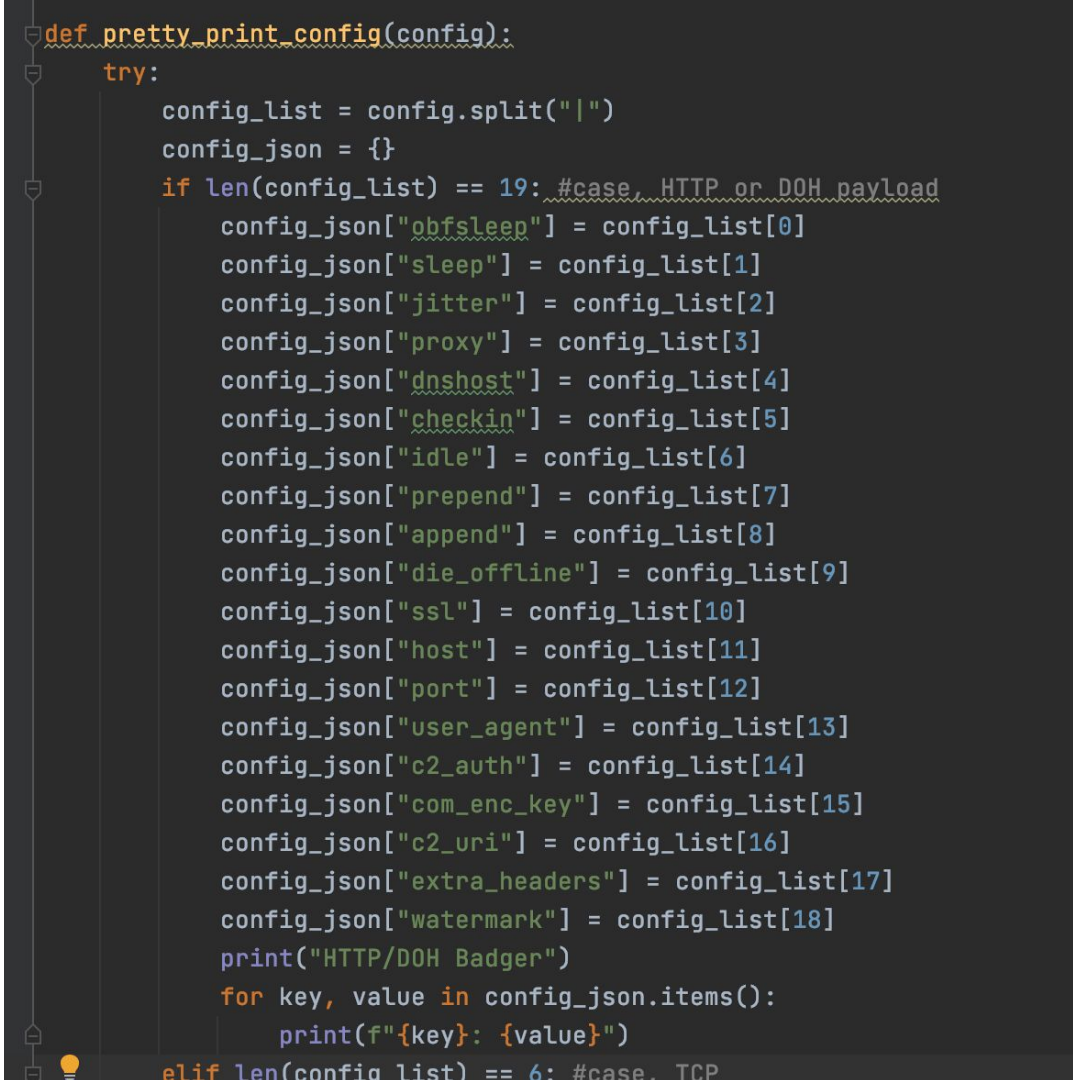
```
NULNULEOTNULNULNULNULNULNULNULENQNULSTXNULNULNULNULNULNULDLEEOTNULNULEOTNULNUL8UETXNULETXNULNULNULNULNUL
NULNULNULNULNULNULDLENULNULNULNULNULNULNULNULDLENULNULNULNULNULNULDLENULNULNULNULNULNULNULNULNULNULDLENULNUL
NULNULàETXNUL6
NULNULNULNULðETXNUL€EOTNULNULNULNULNULNULNULNULNULNULNUL€ETXNULPLUDC3NULNULNULNULNULNULNULNULNULNULNULNULEOT
NULl
EOTNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULL
ñETXNULDLESOHNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
.textNULNULNUL°³STXNULNULDLENULNULNUL´STXNULNULEOTNULNULNULNULNULNULNULNULNULNULNULNULNULNUL NULP`
.dataNULNULNUL0ESCNULNULNULĐSTXNULNULFSNULNULNUL¸STXNULNULNULNULNULNULNULNULNULNULNULNULNUL@NUL`À
.rdataNULNULPNBHNULNULNULðSTXNULNULINDNULNULNULÔSTXNULNULNULNULNULNULNULNULNULNULNULNULNUL@NUL`@
.pdataNULNULPLUDC3NULNULNUL€ETXNULNULDC4NULNULNULXETXNULNULNULNULNULNULNULNULNULNULNULNULNUL@NUL0@
.xdataNULNULhCANNULNULNULNBSPETXNULNULSUBNULNULNULlETXNULNULNULNULNULNULNULNULNULNULNULNULNUL@NUL0@
.bssNULNULNULNUL¦GSNULNULNULÀETXNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL€NUL`À
.edataNULNUL6NULNULNULNULàETXNULNULSTXNULNULNULSSAETXNULNULNULNULNULNULNULNULNULNULNULNULNUL@NUL0@
.idataNULNUL€EOTNULNULNULðETXNULNULACKNULNULNULHTSETXNULNULNULNULNULNULNULNULNULNULNULNULNUL@NUL0À
.relocNULNULlEOTNULNULNULNULEOTNULNULACKNULNULNULSS2ETXNULNULNULNULNULNULNULNULNULNULNULNULNUL@NUL0B
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNUL
NULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULNULé+;
NULNULHHTJàÃé€NULNULNULë\ëpënéHOPNULNULNULéNBSPNULNULNULé§NULNULNULë]ë<ë:ëKëUëkë2ë0ë7ë#ë*ëGë9ëDELëM
éINDNULNULNULë.ë8ëfë4ë&ëDC1ë"ë,ëVTë4IHTJÊHHTJĐAÿàIHTJÊLHTJÀAÿáIHTJÊLHTJÈÿd$(IHTJÊHPLDD$
```

```
block_key = {str} '(%#qv}$n'
code_block = {bytearray: 234512} bytearray(b'l\xf1\xa59
code_block_length = {int} 234512
config_string = {str} 'CJes72u4Vk2omkiBlgF2Sa3tMjFw
configuration_base64_decoded = {bytes: 255} b'\x08\x
```



```
    return ''.join(out)

def decode_config(config_string, code_block, code_block_length):
    length_of_block = len(code_block)
    configuration_base64_decoded = base64.b64decode(config_string)
    block_key = code_block[length_of_block - 8:length_of_block].decode("ascii")
    decrypted_block = rc4_decrypt(code_block, block_key)
    config_key = decrypted_block[length_of_block - 16:length_of_block - 8]
    decrpyted_config = rc4_decrypt(configuration_base64_decoded, config_key)
    if re.match('^\d\|', decrpyted_config): #check that decryption is correct
        return decrpyted_config
    else:
        return ''
```

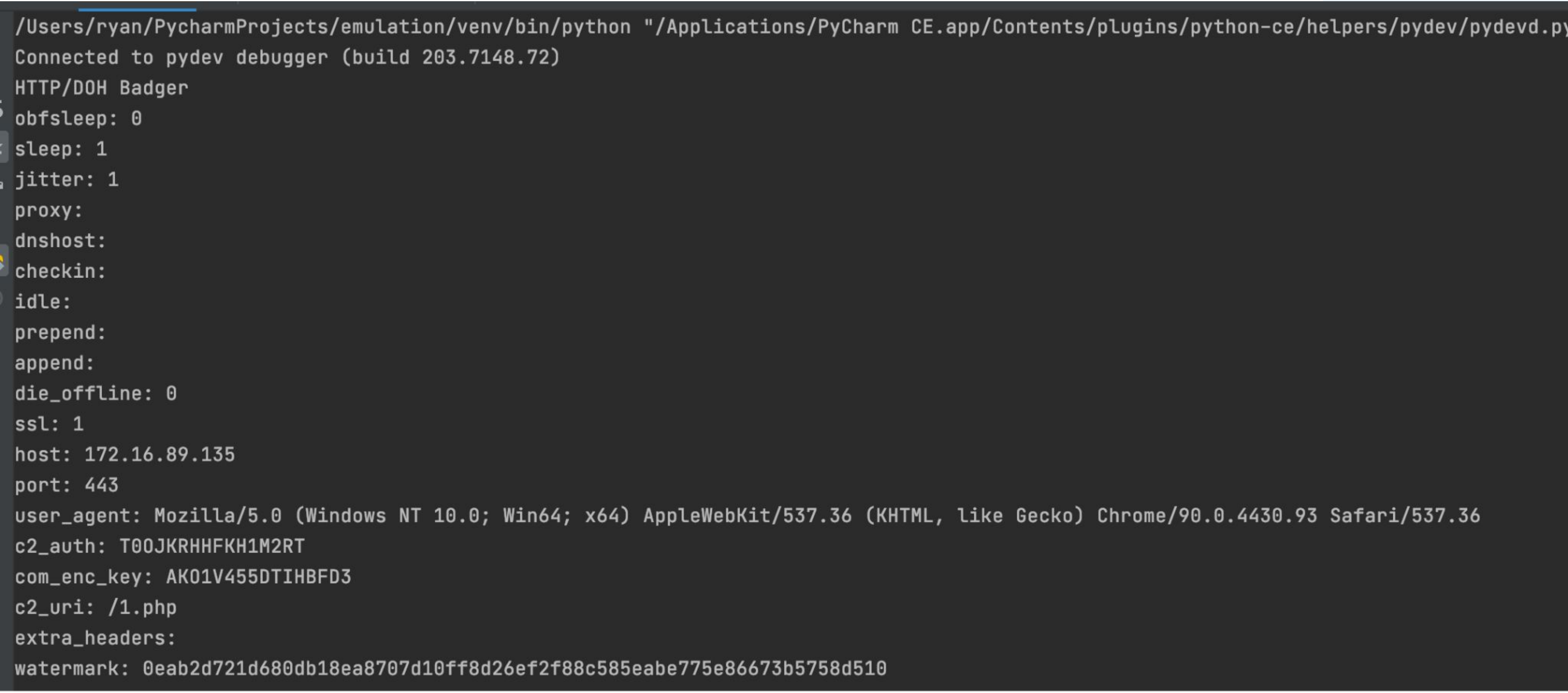
Static Extractor + Config Parser



```
def pretty_print_config(config):
    try:
        config_list = config.split("|")
        config_json = {}
        if len(config_list) == 19: #case, HTTP or DOH payload
            config_json["obfsleep"] = config_list[0]
            config_json["sleep"] = config_list[1]
            config_json["jitter"] = config_list[2]
            config_json["proxy"] = config_list[3]
            config_json["dnshost"] = config_list[4]
            config_json["checkin"] = config_list[5]
            config_json["idle"] = config_list[6]
            config_json["prepend"] = config_list[7]
            config_json["append"] = config_list[8]
            config_json["die_offline"] = config_list[9]
            config_json["ssl"] = config_list[10]
            config_json["host"] = config_list[11]
            config_json["port"] = config_list[12]
            config_json["user_agent"] = config_list[13]
            config_json["c2_auth"] = config_list[14]
            config_json["com_enc_key"] = config_list[15]
            config_json["c2_uri"] = config_list[16]
            config_json["extra_headers"] = config_list[17]
            config_json["watermark"] = config_list[18]
            print("HTTP/DOH Badger")
            for key, value in config_json.items():
                print(f"{key}: {value}")
        elif len(config_list) == 6: #case, TCP
```

INTEZER

Final Result of the Script



```
/Users/ryan/PycharmProjects/emulation/venv/bin/python "/Applications/PyCharm CE.app/Contents/plugins/python-ce/helpers/pydev/pydevd.py
Connected to pydev debugger (build 203.7148.72)
HTTP/DOH Badger
obfsleep: 0
sleep: 1
jitter: 1
proxy:
dnshost:
checkin:
idle:
prepend:
append:
die_offline: 0
ssl: 1
host: 172.16.89.135
port: 443
user_agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.93 Safari/537.36
c2_auth: T0OJKRHHFKH1M2RT
com_enc_key: AKO1V455DTIHBFD3
c2_uri: /1.php
extra_headers:
watermark: 0eab2d721d680db18ea8707d10ff8d26ef2f88c585eabe775e86673b5758d510
```

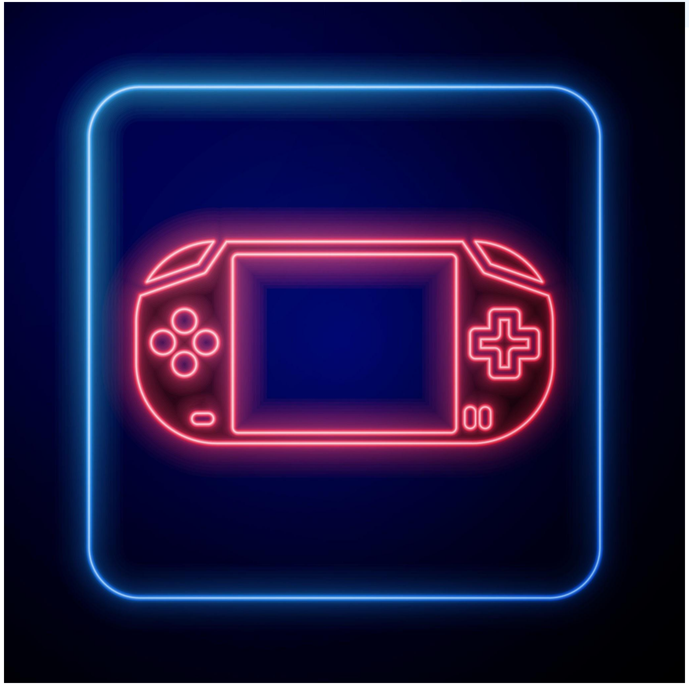
What Did we Learn?

- Now we have everything we need- we possess a script to decode both the payload and its configuration, and we understand the configuration structure, enabling us to hunt for and detect this framework.
- We had to reverse engineer the loader and part of the framework (manually) in order to be able to automate the extraction

Emulators

Leveraging CPU emulation frameworks to run only the relevant portions of the malware in an emulated environment.

This allows researchers to skip over anti-debugging or anti-VM checks while focusing on the decryption or unpacking routines.

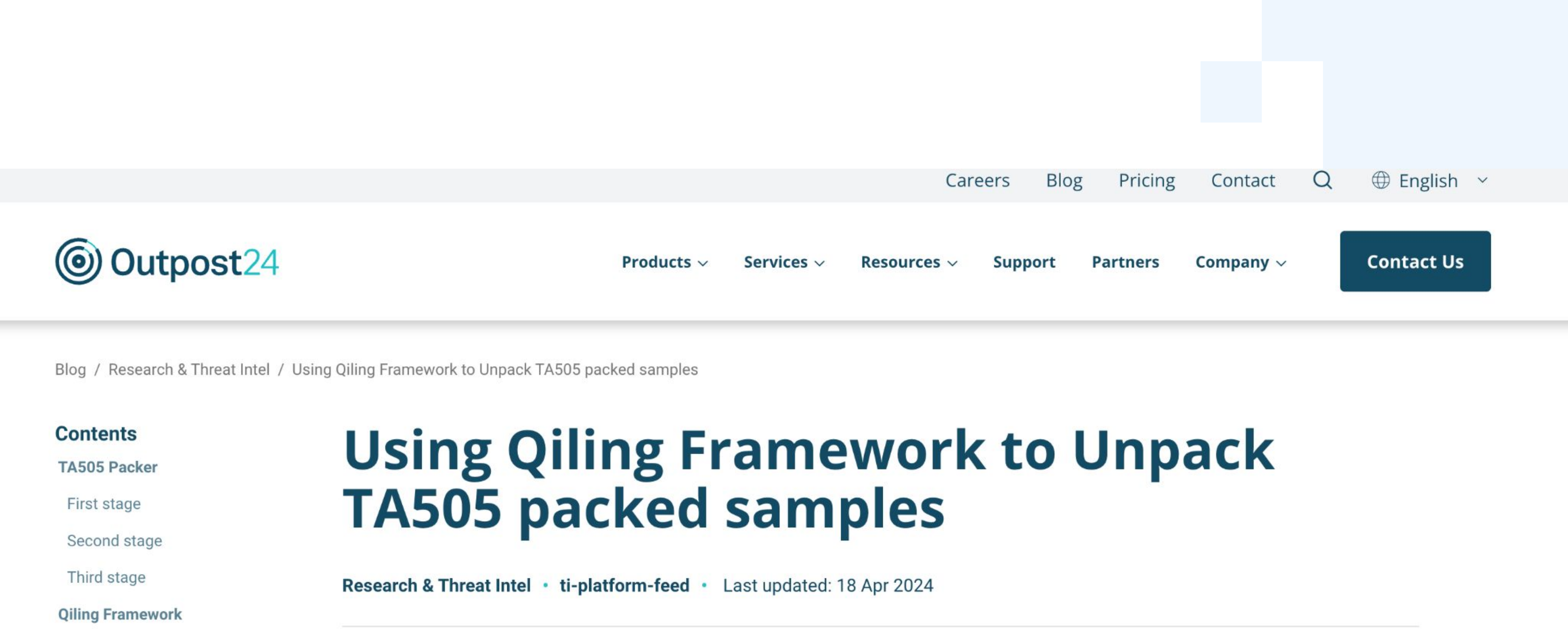


Emulators

- Emulates CPU instructions and memory operations.
- Designed for fine-grained control of code execution (e.g., shellcode, unpacking loops).
- Supports multiple architectures (x86, ARM, MIPS, etc.)
- Ideal for emulating small code routines, such as malware decryption loops.
- Does not emulate OS-level APIs, filesystems, or syscalls.

QILING.IO

- Builds on Unicorn.
- Simulate full OS-level execution environments (Windows, Linux, etc.).
- Allows analysts to run malware as if it's on a real system without using a full VM.
- Extensible via Python for automation and customization.
- Ideal for analyzing full malware loaders



Blog / Research & Threat Intel / Using Qiling Framework to Unpack TA505 packed samples

Contents

- **TA505 Packer**
  - First stage
  - Second stage
  - Third stage
- **Qiling Framework**

Using Qiling Framework to Unpack TA505 packed samples

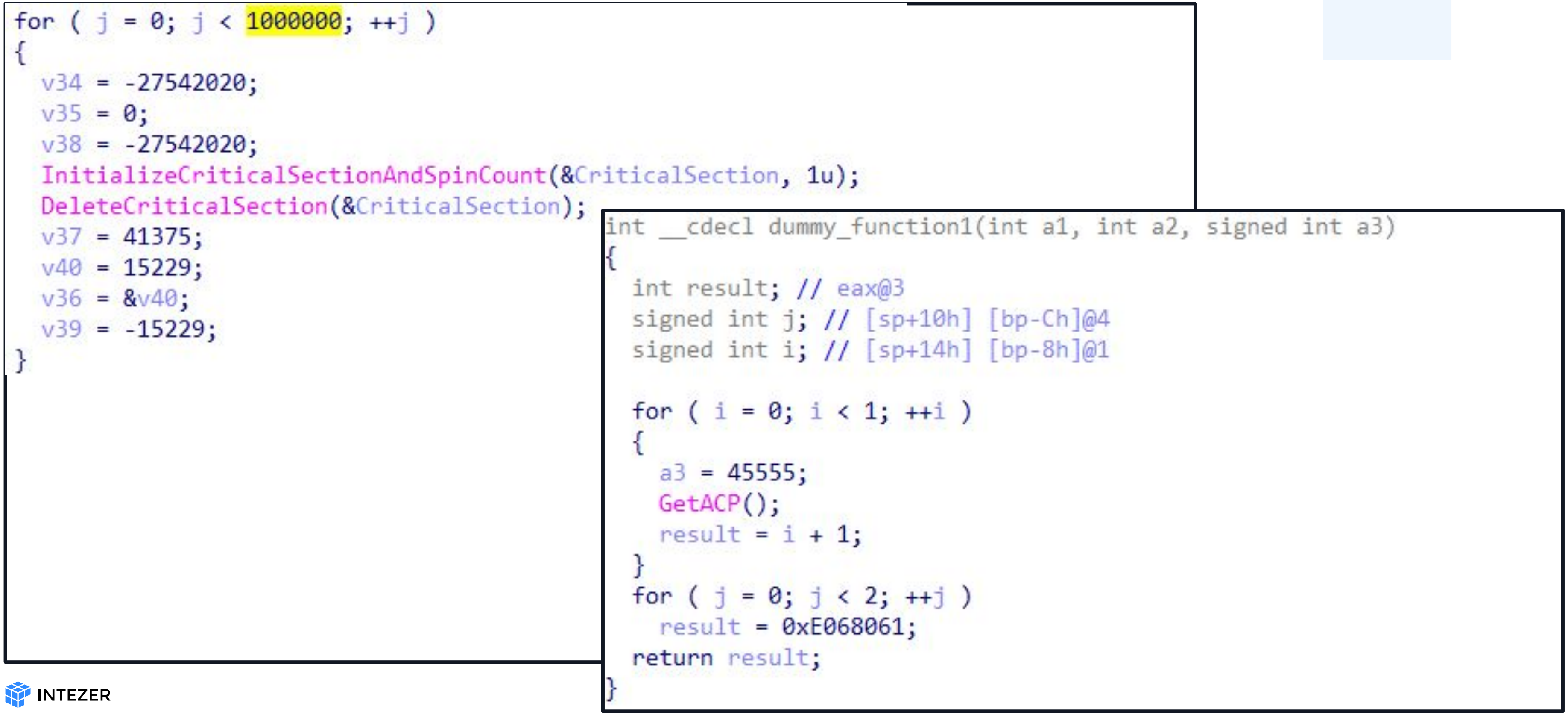
Research & Threat Intel • **ti-platform-feed** • Last updated: 18 Apr 2024

[Original Blog Post]

TA505 Overview

- Financially motivated, Russian affiliated cybercrime group active since at least 2014.
- Known for large-scale, global cyber-attacks targeting various sectors.
- Specializes in deploying banking Trojans, ransomware, and remote access tools (RATs).
  - Dridex (banking Trojan)
  - Locky and Clop (ransomware)
  - FlawedAmmyy, SDBbot, and other RATs
- Adapts quickly to detection, frequently changes tools, techniques, and infrastructure.

Slowing Down Emulation - Dummy Functions

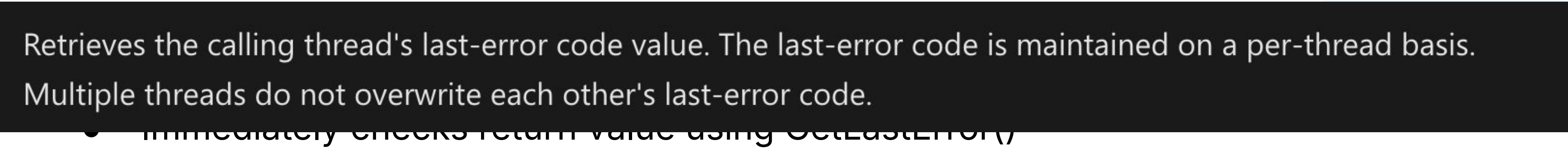


```
for ( j = 0; j < 1000000; ++j )
{
  v34 = -27542020;
  v35 = 0;
  v38 = -27542020;
  InitializeCriticalSectionAndSpinCount(&CriticalSection, 1u);
  DeleteCriticalSection(&CriticalSection);
  v37 = 41375;
  v40 = 15229;
  v36 = &v40;
  v39 = -15229;
}
```

```
int __cdecl dummy_function1(int a1, int a2, signed int a3)
{
  int result; // eax@3
  signed int j; // [sp+10h] [bp-Ch]@4
  signed int i; // [sp+14h] [bp-8h]@1

  for ( i = 0; i < 1; ++i )
  {
    a3 = 45555;
    GetACP();
    result = i + 1;
  }
  for ( j = 0; j < 2; ++j )
    result = 0xE068061;
  return result;
}
```

INTEZER

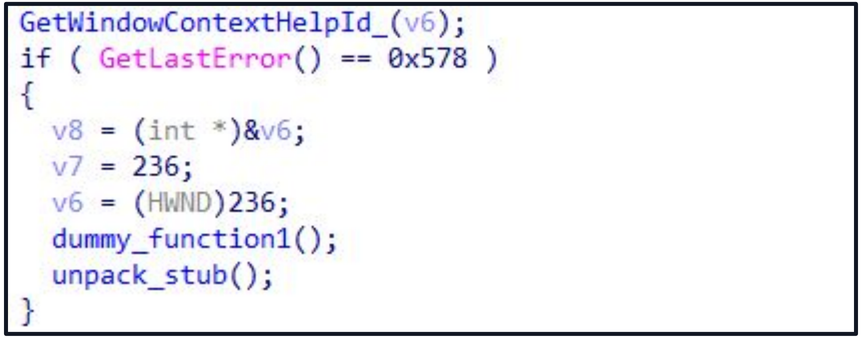
Anti Emulation - GetLastError()



Retrieves the calling thread's last-error code value. The last-error code is maintained on a per-thread basis. Multiple threads do not overwrite each other's last-error code.

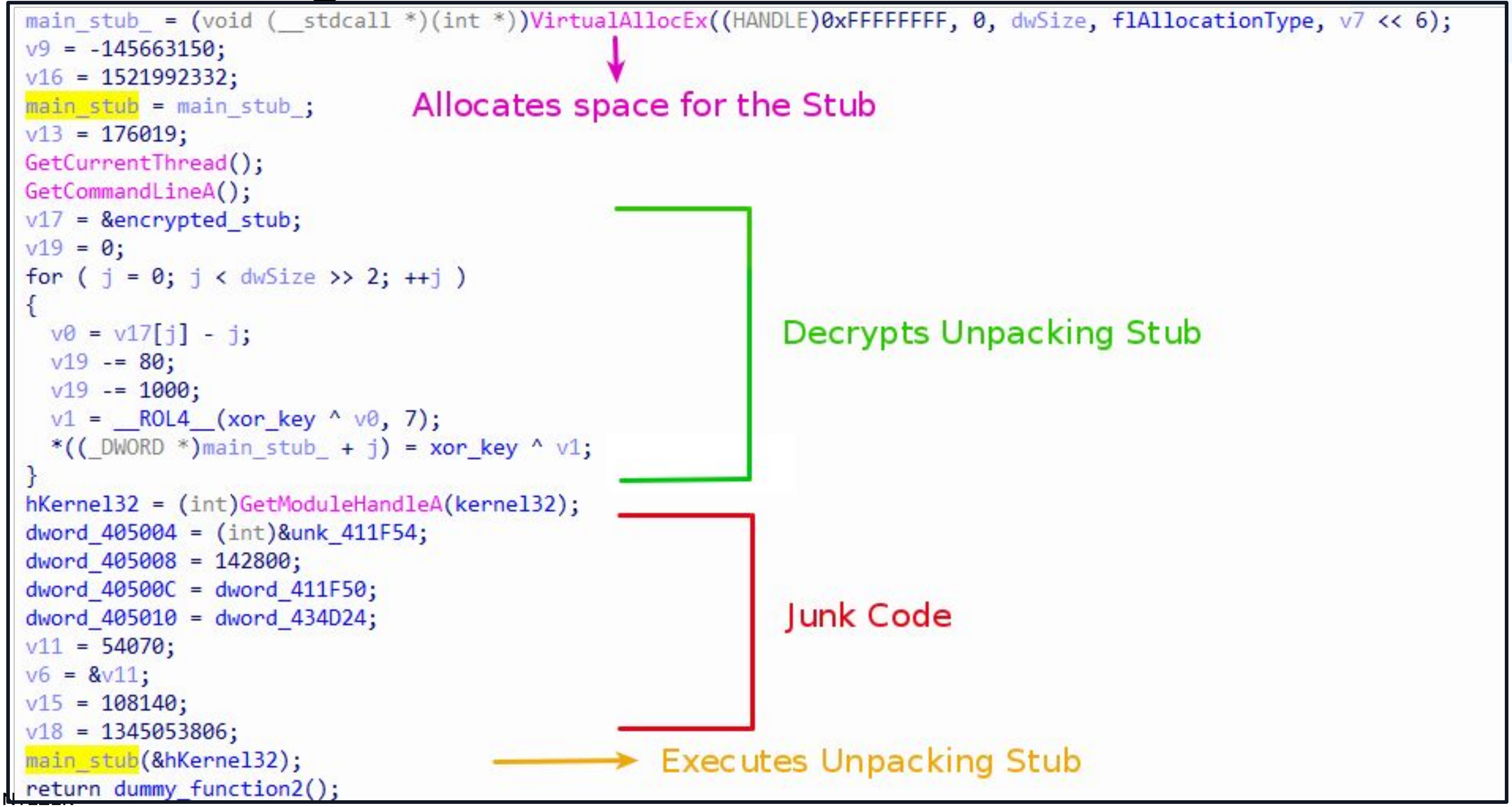
- Immediately checks return value using GetLastError()
- Expected error code (e.g., ERROR_INVALID_HANDLE) must be returned
- If the error code is missing or incorrect, malware detects:
  - Emulator or sandbox environment
  - Incomplete Windows API emulation
- Based on detection, malware may:
  - Exit silently
  - Skip payload unpacking
  - Execute fake code paths to mislead analysis
- Emulators like Qiling can hook APIs to simulate expected errors and bypass detection

Anti Emulation - GetLastError()



```
GetWindowContextHelpId_(v6);
if ( GetLastError() == 0x578 )
{
  v8 = (int *)&v6;
  v7 = 236;
  v6 = (HWND)236;
  dummy_function1();
  unpack_stub();
}
```

Second Stage Loader - Anti Emulation



```
main_stub_ = (void (__stdcall *)(int *))VirtualAllocEx((HANDLE)0xFFFFFFFF, 0, dwSize, flAllocationType, v7 << 6);
v9 = -145663150;
v16 = 1521992332;
main_stub = main_stub_;
v13 = 176019;
GetCurrentThread();
GetCommandLineA();
v17 = &encrypted_stub;
v19 = 0;
for ( j = 0; j < dwSize >> 2; ++j )
{
  v0 = v17[j] - j;
  v19 -= 80;
  v19 -= 1000;
  v1 = __ROL4__(xor_key ^ v0, 7);
  *((_DWORD *)main_stub_ + j) = xor_key ^ v1;
}
hKernel32 = (int)GetModuleHandleA(kernel32);
dword_405004 = (int)&unk_411F54;
dword_405008 = 142800;
dword_40500C = dword_411F50;
dword_405010 = dword_434D24;
v11 = 54070;
v6 = &v11;
v15 = 108140;
v18 = 1345053806;
main_stub(&hKernel32);
return dummy_function2();
```

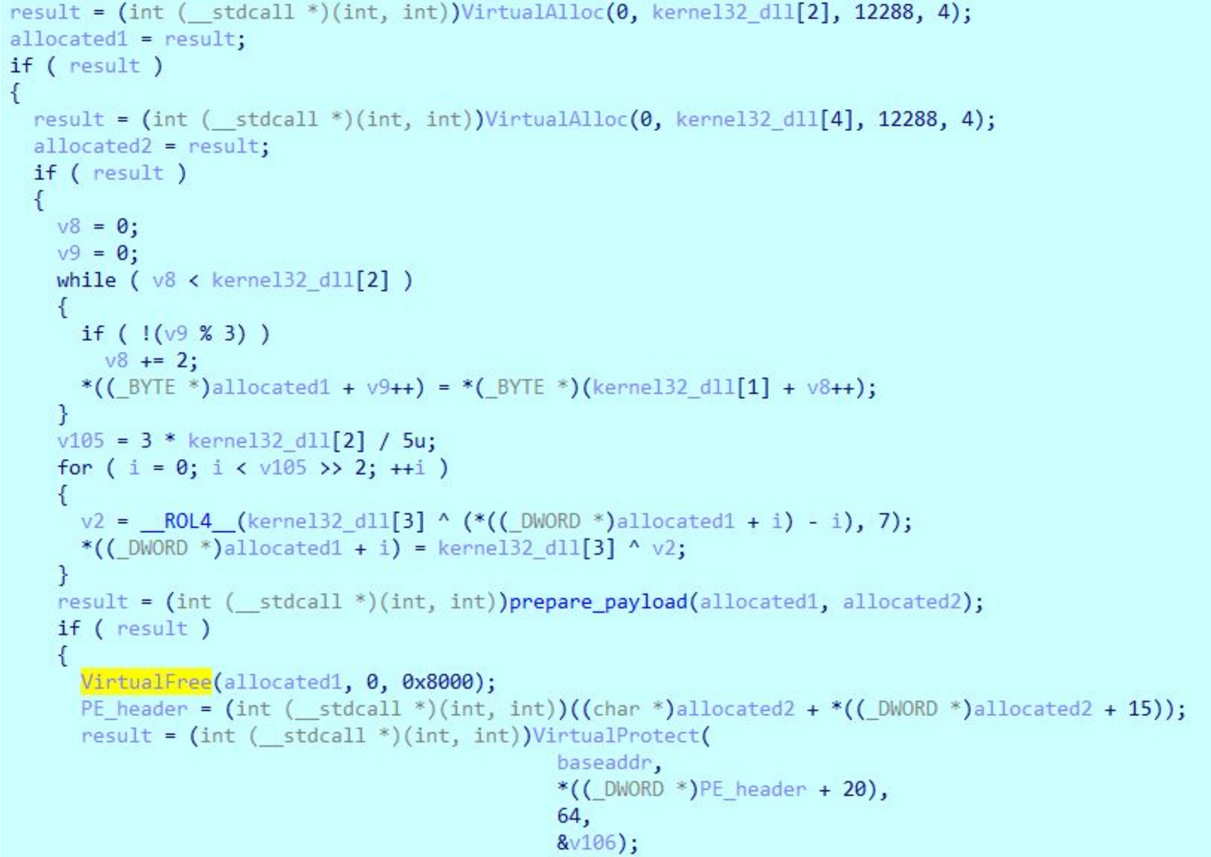
Allocates space for the Stub

Decrypts Unpacking Stub

Junk Code

Executes Unpacking Stub

Third Stage Loader - Obfuscation



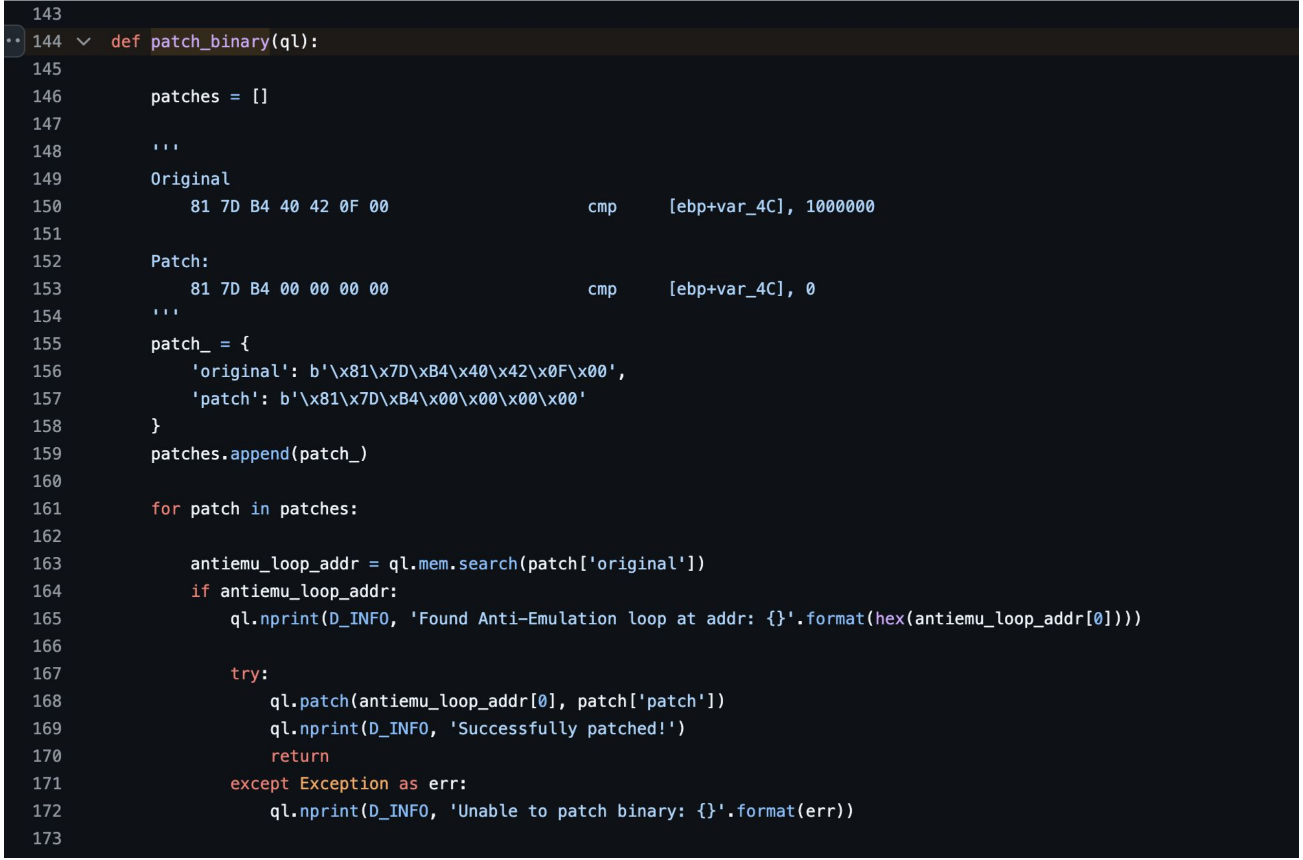
```
result = (int (__stdcall *)(int, int))VirtualAlloc(0, kernel32_dll[2], 12288, 4);
allocated1 = result;
if ( result )
{
  result = (int (__stdcall *)(int, int))VirtualAlloc(0, kernel32_dll[4], 12288, 4);
  allocated2 = result;
  if ( result )
  {
    v8 = 0;
    v9 = 0;
    while ( v8 < kernel32_dll[2] )
    {
      if ( !(v9 % 3) )
        v8 += 2;
      *((_BYTE *)allocated1 + v9++) = *(_BYTE *)(kernel32_dll[1] + v8++);
    }
    v105 = 3 * kernel32_dll[2] / 5u;
    for ( i = 0; i < v105 >> 2; ++i )
    {
      v2 = __ROL4__(kernel32_dll[3] ^ (*((_DWORD *)allocated1 + i) - i), 7);
      *((_DWORD *)allocated1 + i) = kernel32_dll[3] ^ v2;
    }
    result = (int (__stdcall *)(int, int))prepare_payload(allocated1, allocated2);
    if ( result )
    {
      VirtualFree(allocated1, 0, 0x8000);
      PE_header = (int (__stdcall *)(int, int))((char *)allocated2 + *((_DWORD *)allocated2 + 15));
      result = (int (__stdcall *)(int, int))VirtualProtect(
                                              baseaddr,
                                              *((_DWORD *)PE_header + 20),
                                              64,
                                              &v106);
```

The Goals of the Emulation

- Detect and bypass anti-emulation checks to allow execution of the packer stub
- Track dynamically allocated memory regions (recording address and size)
- Dump the unpacked PE payload at the right moment (e.g., just before VirtualFree() is called)

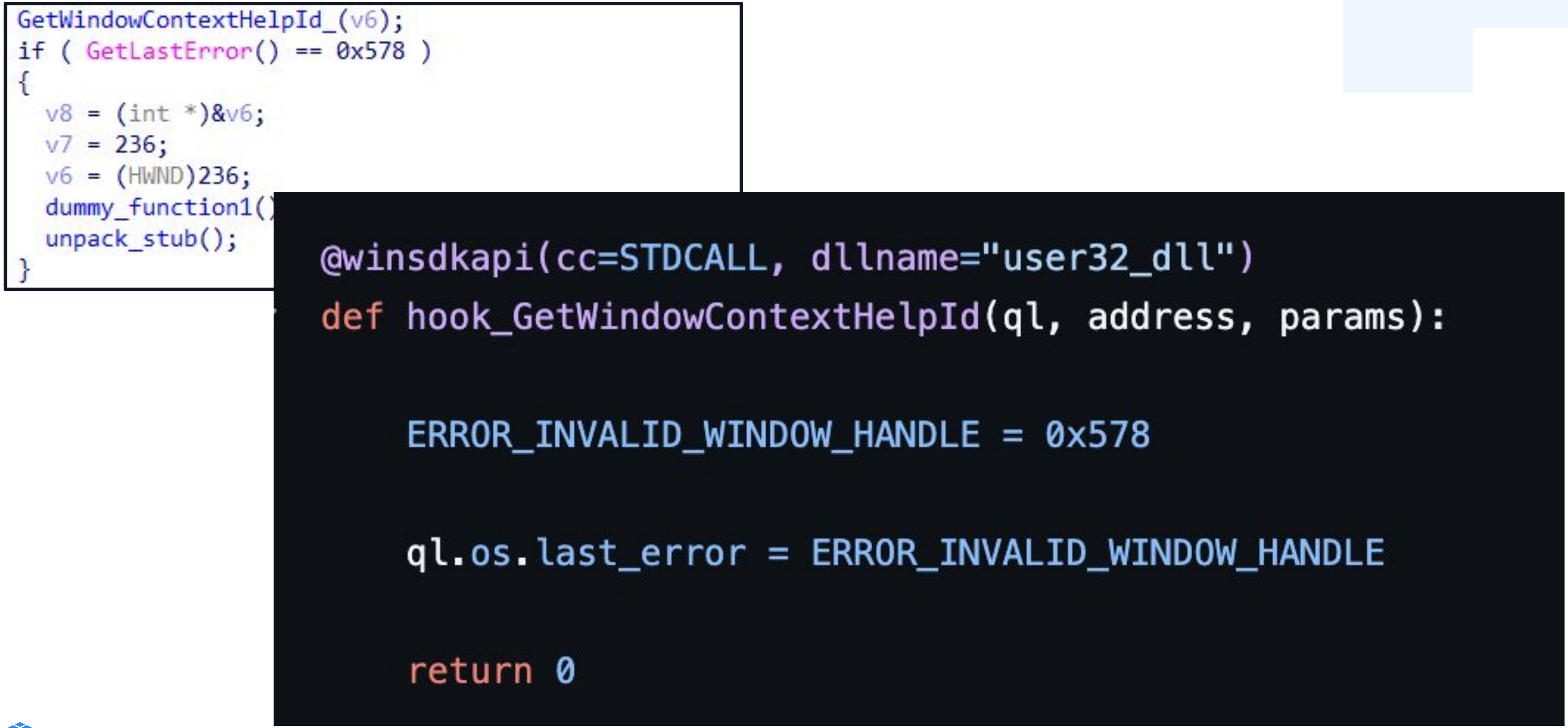
Source Code: unpack-ta505packer-qiling(GitHub)

Emulation - Bypassing Loops



```

def patch_binary(ql):

    patches = []

    '''
    Original
        81 7D B4 40 42 0F 00                    cmp     [ebp+var_4C], 1000000

    Patch:
        81 7D B4 00 00 00 00                    cmp     [ebp+var_4C], 0
    '''
    patch_ = {
        'original': b'\x81\x7D\xB4\x40\x42\x0F\x00',
        'patch': b'\x81\x7D\xB4\x00\x00\x00\x00'
    }
    patches.append(patch_)

    for patch in patches:

        antiemu_loop_addr = ql.mem.search(patch['original'])
        if antiemu_loop_addr:
            ql.nprint(D_INFO, 'Found Anti-Emulation loop at addr: {}'.format(hex(antiemu_loop_addr[0])))

            try:
                ql.patch(antiemu_loop_addr[0], patch['patch'])
                ql.nprint(D_INFO, 'Successfully patched!')
                return
            except Exception as err:
                ql.nprint(D_INFO, 'Unable to patch binary: {}'.format(err))

```

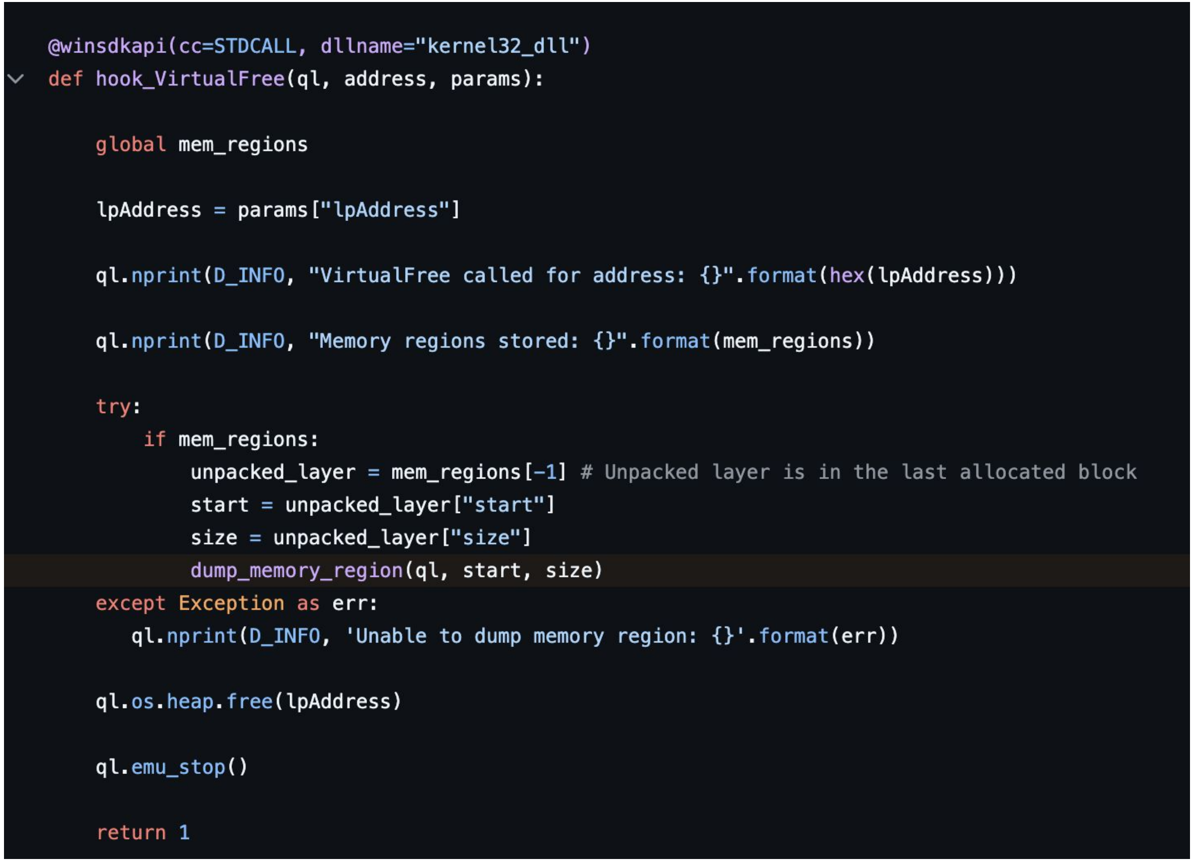
Emulation - Bypassing the Error Check



```
GetWindowContextHelpId_(v6);
if ( GetLastError() == 0x578 )
{
  v8 = (int *)&v6;
  v7 = 236;
  v6 = (HWND)236;
  dummy_function1(
  unpack_stub();
}
```

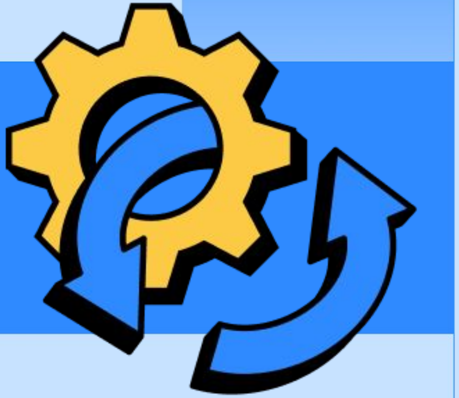
```python
@winsdkapi(cc=STDCALL, dllname="user32_dll")
def hook_GetWindowContextHelpId(ql, address, params):

    ERROR_INVALID_WINDOW_HANDLE = 0x578

    ql.os.last_error = ERROR_INVALID_WINDOW_HANDLE

    return 0
```

Emulation - Getting the Payload from the Allocated Memory



```
@winsdkapi(cc=STDCALL, dllname="kernel32_dll")
def hook_VirtualFree(ql, address, params):

    global mem_regions

    lpAddress = params["lpAddress"]

    ql.nprint(D_INFO, "VirtualFree called for address: {}".format(hex(lpAddress)))

    ql.nprint(D_INFO, "Memory regions stored: {}".format(mem_regions))

    try:
        if mem_regions:
            unpacked_layer = mem_regions[-1] # Unpacked layer is in the last allocated block
            start = unpacked_layer["start"]
            size = unpacked_layer["size"]
            dump_memory_region(ql, start, size)
    except Exception as err:
       ql.nprint(D_INFO, 'Unable to dump memory region: {}'.format(err))

    ql.os.heap.free(lpAddress)

    ql.emu_stop()

    return 1
```

Afterthoughts



- Emulation requires a good understanding of the malware (anti-emulation and anti-analysis)
- Requires low-level programming skills

Sandbox

We all know them, we all love them...

Did you know that sandboxs can also be painful to maintain?

About Sandboxs

- Cuckoo - open-source sandbox projects
- Cape - fork of Cuckoo
  - Focused on automated unpacking and config extraction from malware
- Monitoring = DLL injections + hooks
  - Malware makes a call to a monitored API call → hooked function is invoked → log the call → call the actual API function

But, what if there is a bug in the hook?

Capemon and Cuckoomon

User-mode DLL monitor injected into target process during sandboxing

Intercepts and logs Windows API calls and runtime behavior

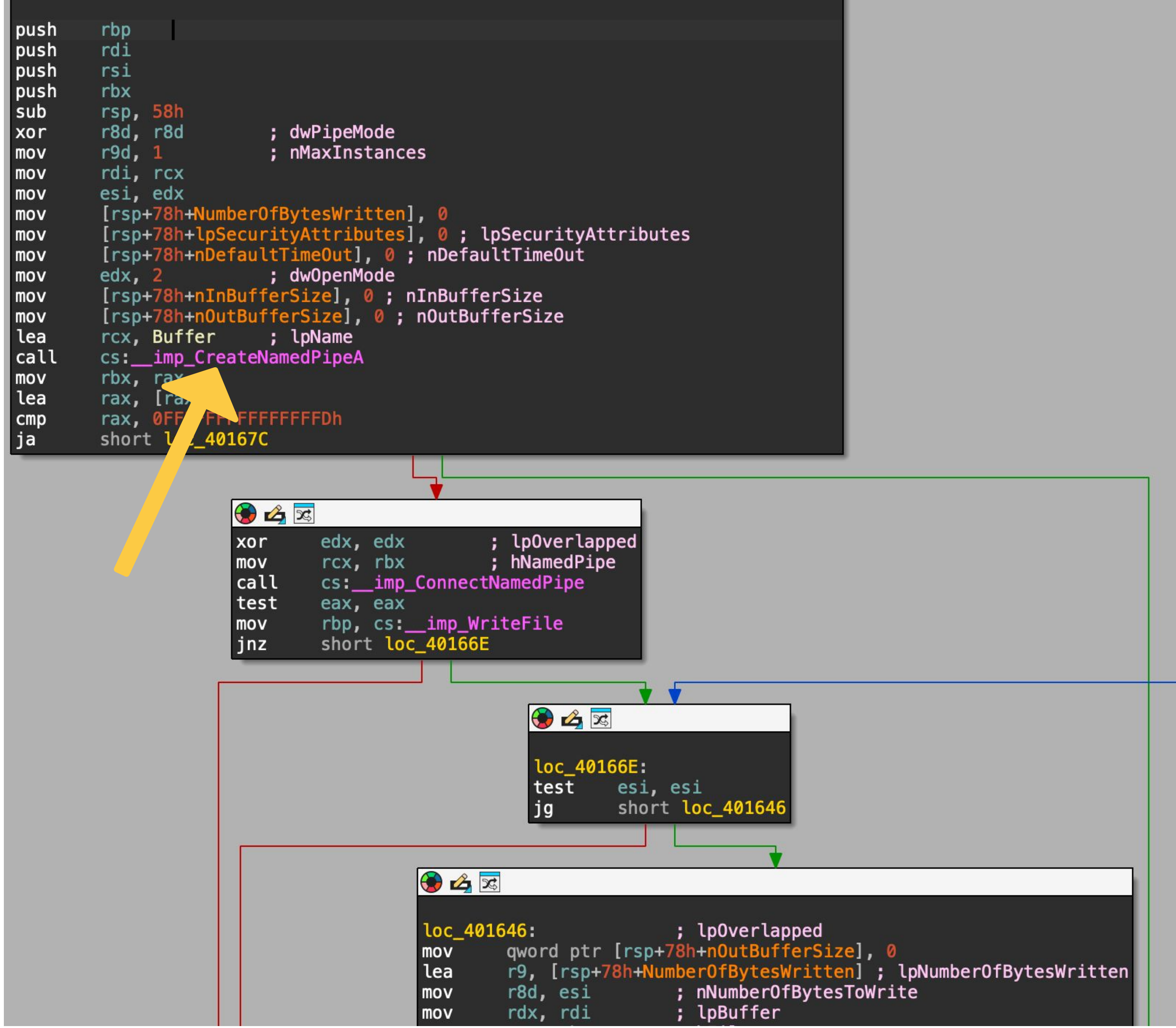
How It Works

- Injected early via remote thread or modified entry point
- Installs hooks into key API functions (e.g., CreateFile, CreateProcess)

Monitored Activities

- File, registry, memory, and process operations
- Network activity
- DLL loads and code injection attempts
- Suspicious behavior like persistence or API misuse

Stagless Loader for Cobalt Strike Beacon



```
push    rbp
push    rdi
push    rsi
push    rbx
sub     rsp, 58h
xor     r8d, r8d          ; dwPipeMode
mov     r9d, 1            ; nMaxInstances
mov     rdi, rcx
mov     esi, edx
mov     [rsp+78h+NumberOfBytesWritten], 0
mov     [rsp+78h+lpSecurityAttributes], 0 ; lpSecurityAttributes
mov     [rsp+78h+nDefaultTimeOut], 0 ; nDefaultTimeOut
mov     edx, 2            ; dwOpenMode
mov     [rsp+78h+nInBufferSize], 0 ; nInBufferSize
mov     [rsp+78h+nOutBufferSize], 0 ; nOutBufferSize
lea     rcx, Buffer       ; lpName
call    cs:__imp_CreateNamedPipeA
mov     rbx, rax
lea     rax, [ra
cmp     rax, 0FF  FFFFFFFFDh
ja      short l  _40167C
```

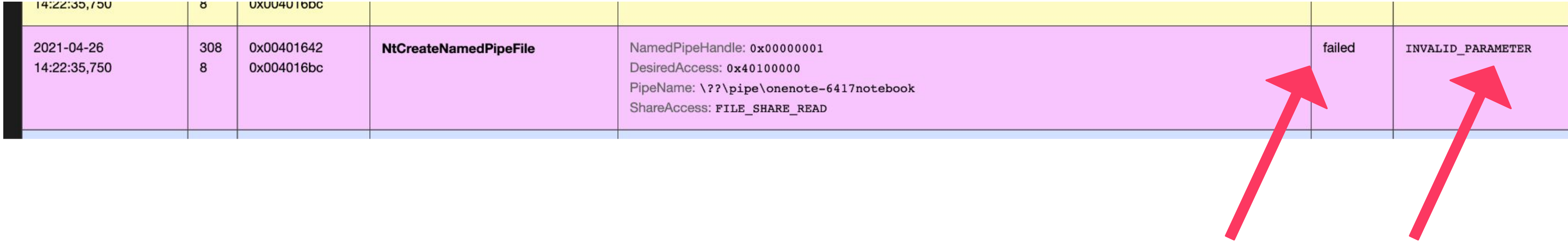
```
xor     edx, edx          ; lpOverlapped
mov     rcx, rbx          ; hNamedPipe
call    cs:__imp_ConnectNamedPipe
test    eax, eax
mov     rbp, cs:__imp_WriteFile
jnz     short loc_40166E
```

```
loc_40166E:
test    esi, esi
jg      short loc_401646
```

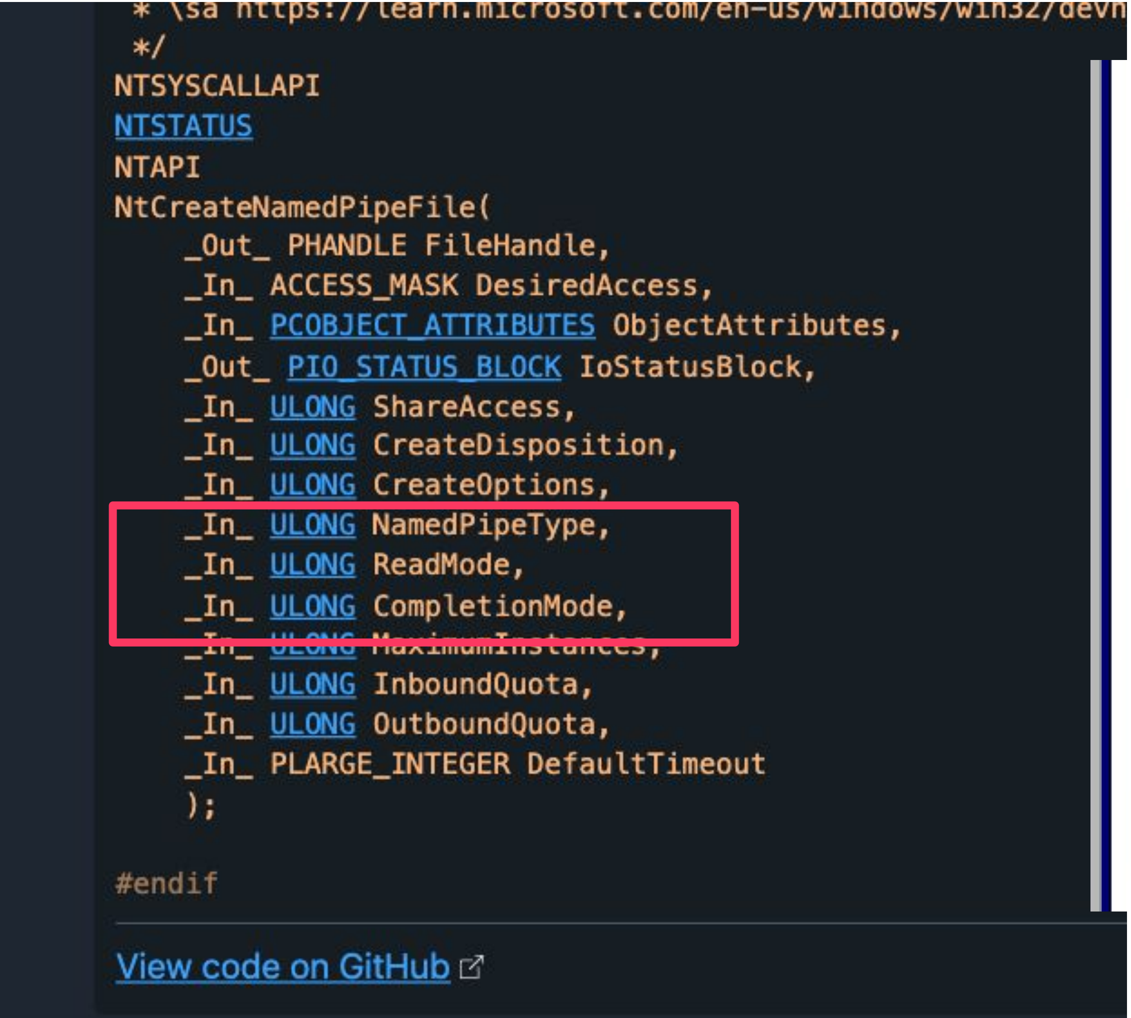
```
loc_401646:               ; lpOverlapped
mov     qword ptr [rsp+78h+nOutBufferSize], 0
lea     r9, [rsp+78h+NumberOfBytesWritten] ; lpNumberOfBytesWritten
mov     r8d, esi          ; nNumberOfBytesToWrite
mov     rdx, rdi          ; lpBuffer
```

INTEZER

Stagless Cobalt Strike Beacon

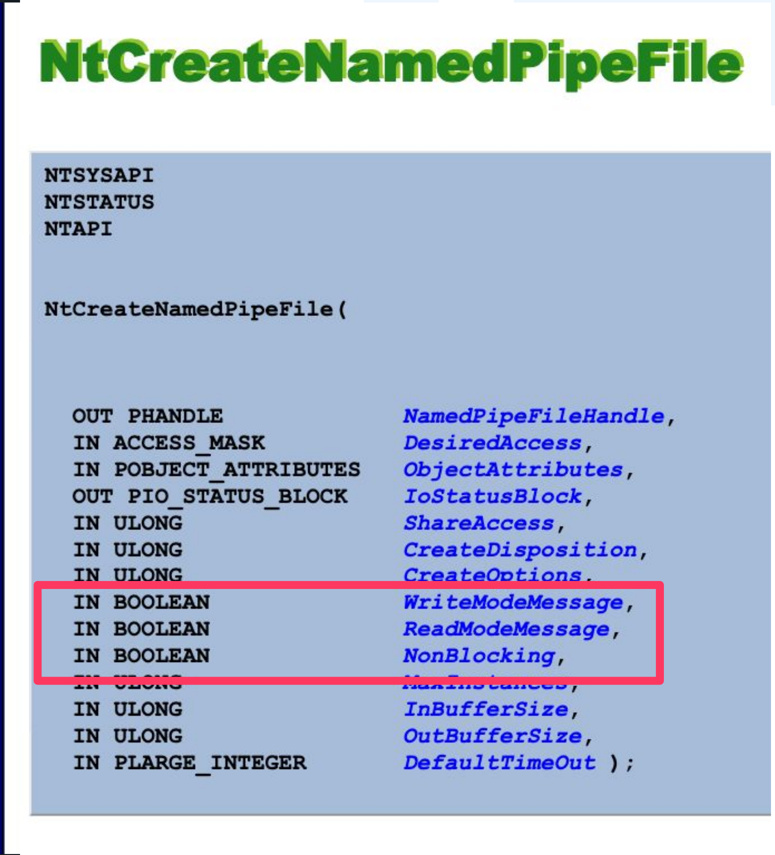


14:22:35,750	8	0x004016bc				
2021-04-26 14:22:35,750	308 8	0x00401642 0x004016bc	**NtCreateNamedPipeFile**	NamedPipeHandle: `0x00000001` DesiredAccess: `0x40100000` PipeName: `\??\pipe\onenote-6417notebook` ShareAccess: `FILE_SHARE_READ`	failed	`INVALID_PARAMETER`



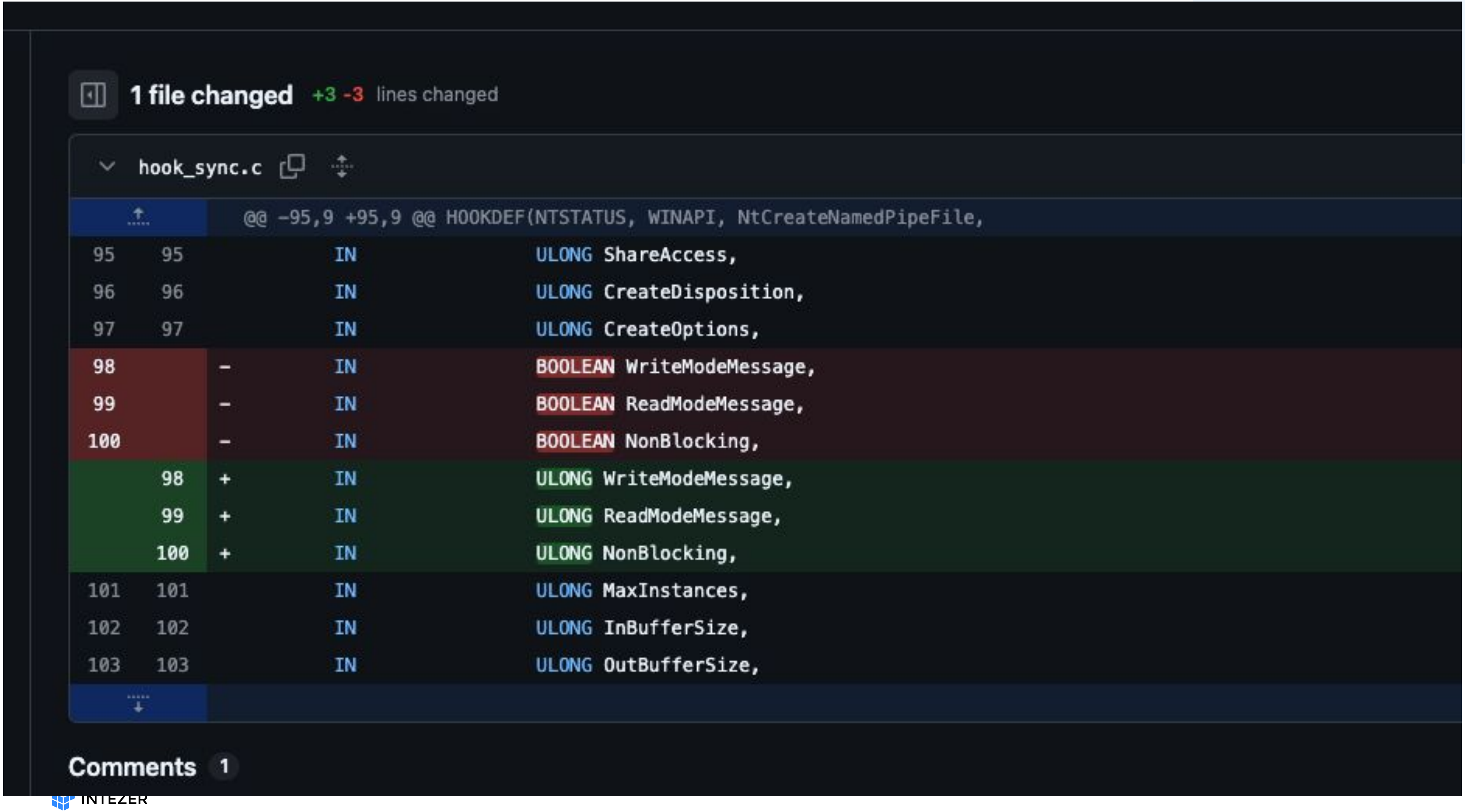
```
 * \sa https://learn.microsoft.com/en-us/windows/win32/devn
 */
NTSYSCALLAPI
NTSTATUS
NTAPI
NtCreateNamedPipeFile(
    _Out_ PHANDLE FileHandle,
    _In_ ACCESS_MASK DesiredAccess,
    _In_ PCOBJECT_ATTRIBUTES ObjectAttributes,
    _Out_ PIO_STATUS_BLOCK IoStatusBlock,
    _In_ ULONG ShareAccess,
    _In_ ULONG CreateDisposition,
    _In_ ULONG CreateOptions,
    _In_ ULONG NamedPipeType,
    _In_ ULONG ReadMode,
    _In_ ULONG CompletionMode,
    _In_ ULONG MaximumInstances,
    _In_ ULONG InboundQuota,
    _In_ ULONG OutboundQuota,
    _In_ PLARGE_INTEGER DefaultTimeout
    );

#endif
```

View code on GitHub



NtCreateNamedPipeFile

```
NTSYSAPI
NTSTATUS
NTAPI


NtCreateNamedPipeFile(



  OUT PHANDLE               NamedPipeFileHandle,
  IN ACCESS_MASK            DesiredAccess,
  IN POBJECT_ATTRIBUTES     ObjectAttributes,
  OUT PIO_STATUS_BLOCK      IoStatusBlock,
  IN ULONG                  ShareAccess,
  IN ULONG                  CreateDisposition,
  IN ULONG                  CreateOptions,
  IN BOOLEAN                WriteModeMessage,
  IN BOOLEAN                ReadModeMessage,
  IN BOOLEAN                NonBlocking,
  IN ULONG                  MaxInstances,
  IN ULONG                  InBufferSize,
  IN ULONG                  OutBufferSize,
  IN PLARGE_INTEGER         DefaultTimeOut );
```



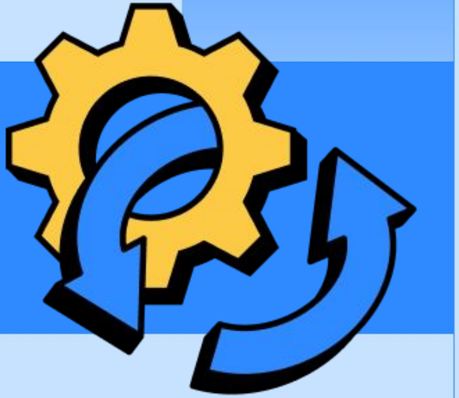
1 file changed +3 -3 lines changed

hook_sync.c

```
@@ -95,9 +95,9 @@ HOOKDEF(NTSTATUS, WINAPI, NtCreateNamedPipeFile,
            IN              ULONG ShareAccess,
            IN              ULONG CreateDisposition,
            IN              ULONG CreateOptions,
-           IN              BOOLEAN WriteModeMessage,
-           IN              BOOLEAN ReadModeMessage,
-           IN              BOOLEAN NonBlocking,
+           IN              ULONG WriteModeMessage,
+           IN              ULONG ReadModeMessage,
+           IN              ULONG NonBlocking,
            IN              ULONG MaxInstances,
            IN              ULONG InBufferSize,
            IN              ULONG OutBufferSize,
```

Comments 1

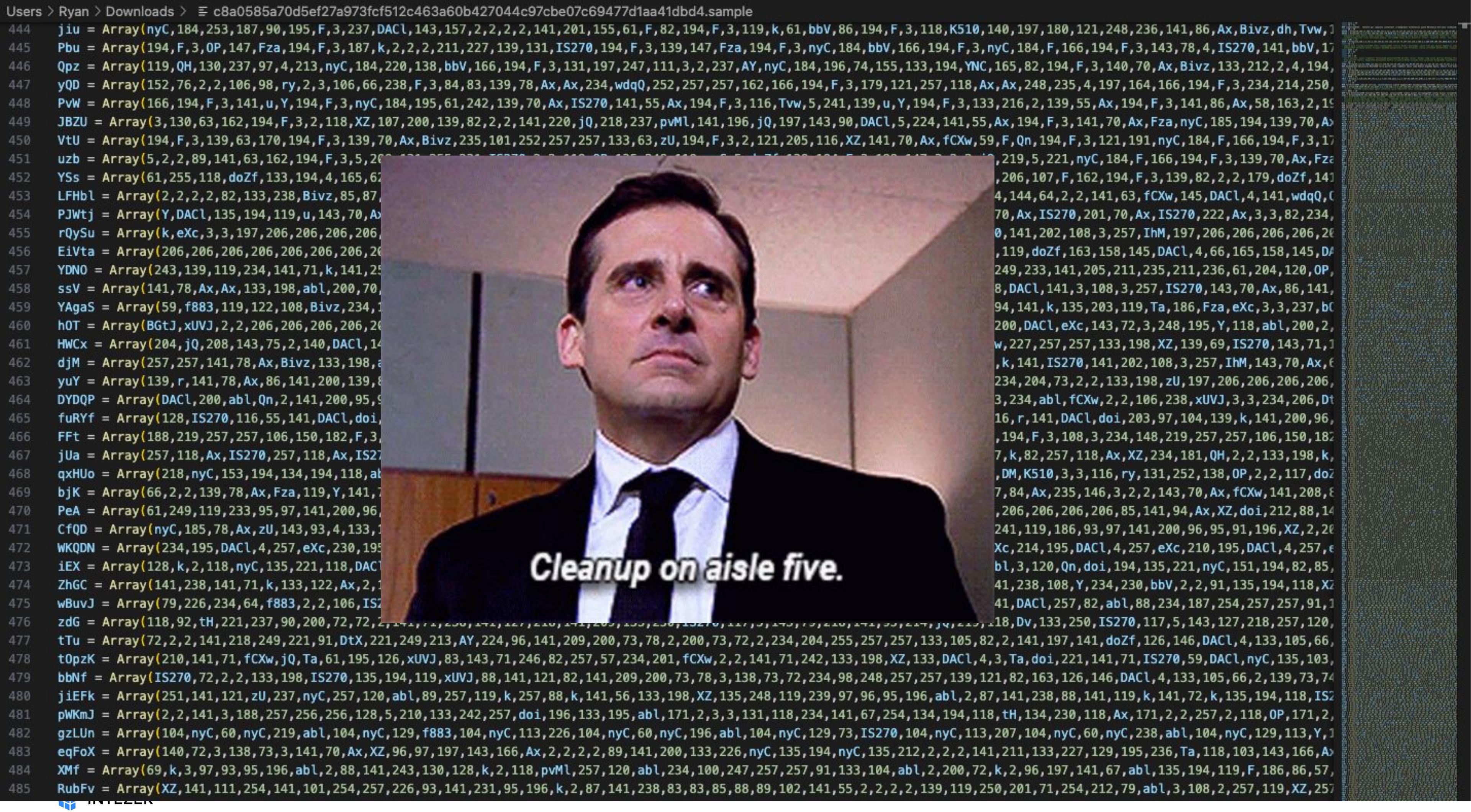
Afterthoughts



- This is an example of how our tool can fail us
- Finding the issue can be complicated
- Needle in a haystack

The Struggles of Maintaining a Sandbox Environment

- Constantly evolving anti-sandbox techniques used by malware
- Difficult to replicate a realistic OS and user environment
- Use of delays and stealth tactics to evade short executions
- High resource consumption (CPU, memory, storage)
- Malware may require specific triggers (user input, time, network) to activate
- Scalability issues when analyzing large volumes of samples
- Incomplete API and driver coverage can break analysis or reveal the sandbox

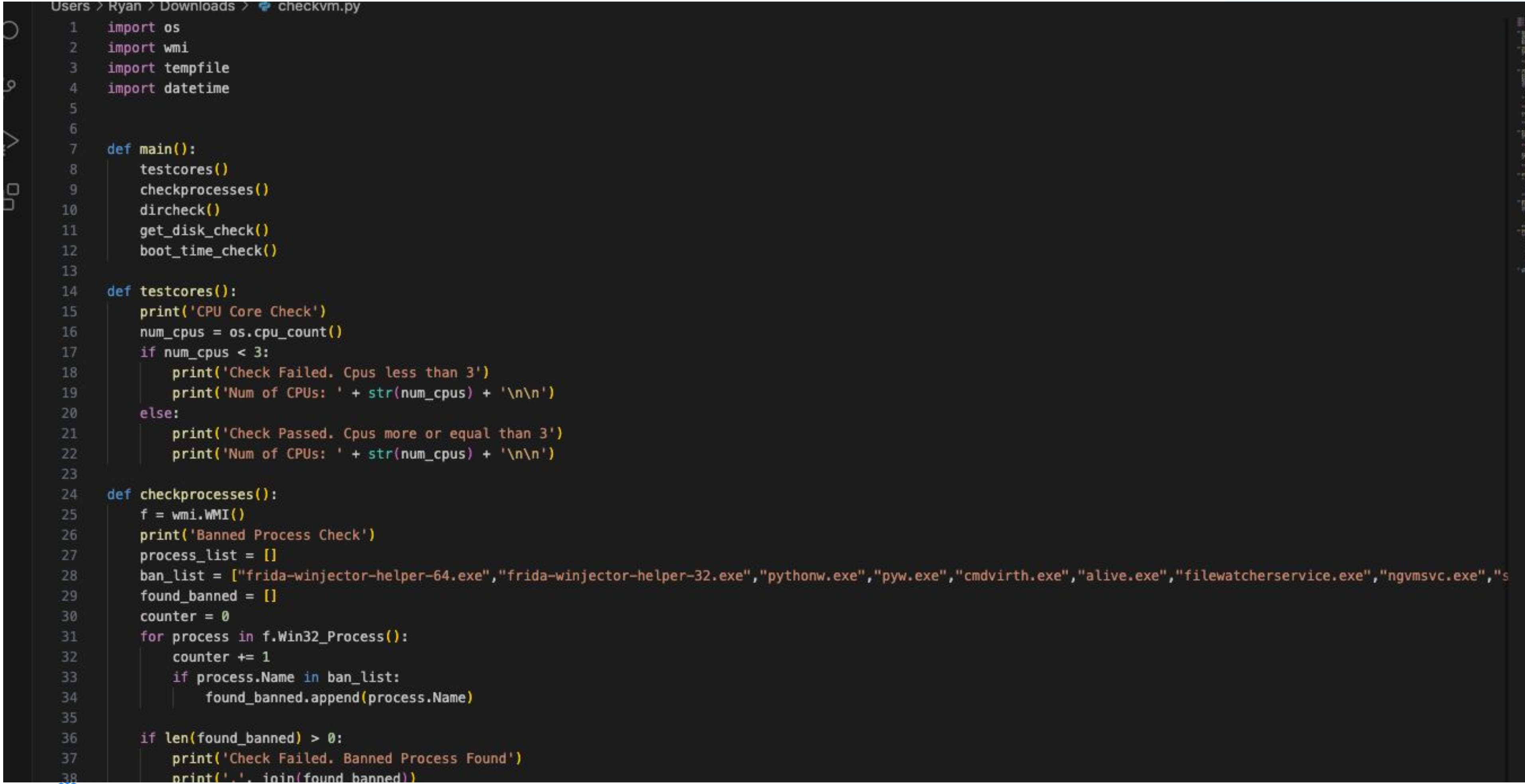

Users > Ryan > Downloads > c8a0585a70d5ef27a973fcf512c463a60b427044c97cbe07c69477d1aa41dbd4.sample
Cleanup on aisle five.

```
jiu = Array(nyC,184,253,187,90,195,F,3,237,DACl,143,157,2,2,2,2,141,201,155,61,F,82,194,F,3,119,k,61,bbV,86,194,F,3,118,K510,140,197,180,121,248,236,141,86,Ax,Bivz,dh,Tvw,
Pbu = Array(194,F,3,OP,147,Fza,194,F,3,187,k,2,2,2,211,227,139,131,IS270,194,F,3,139,147,Fza,194,F,3,nyC,184,bbV,166,194,F,3,nyC,184,F,166,194,F,3,143,78,4,IS270,141,bbV,1
Qpz = Array(119,QH,130,237,97,4,213,nyC,184,220,138,bbV,166,194,F,3,131,197,247,111,3,2,237,AY,nyC,184,196,74,155,133,194,YNC,165,82,194,F,3,140,70,Ax,Bivz,133,212,2,4,194,
yQD = Array(152,76,2,2,106,98,ry,2,3,106,66,238,F,3,84,83,139,78,Ax,Ax,234,wdqQ,252,257,257,162,166,194,F,3,179,121,257,118,Ax,Ax,248,235,4,197,164,166,194,F,3,234,214,250,
PvW = Array(166,194,F,3,141,u,Y,194,F,3,nyC,184,195,61,242,139,70,Ax,IS270,141,55,Ax,194,F,3,116,Tvw,5,241,139,u,Y,194,F,3,133,216,2,139,55,Ax,194,F,3,141,86,Ax,58,163,2,1
JBZU = Array(3,130,63,162,194,F,3,2,118,XZ,107,200,139,82,2,2,141,220,jQ,218,237,pvMl,141,196,jQ,197,143,90,DACl,5,224,141,55,Ax,194,F,3,141,70,Ax,Fza,nyC,185,194,139,70,A
VtU = Array(194,F,3,139,63,170,194,F,3,139,70,Ax,Bivz,235,101,252,257,257,133,63,zU,194,F,3,2,121,205,116,XZ,141,70,Ax,fCXw,59,F,Qn,194,F,3,121,191,nyC,184,F,166,194,F,3,1
uzb = Array(5,2,2,89,141,63,162,194,F,3,5,2                                        ,219,5,221,nyC,184,F,166,194,F,3,139,70,Ax,Fza
YSs = Array(61,255,118,doZf,133,194,4,165,6                                        ,206,107,F,162,194,F,3,139,82,2,2,179,doZf,141
LFHbl = Array(2,2,2,2,82,133,238,Bivz,85,87,                                       4,144,64,2,2,141,63,fCXw,145,DACl,4,141,wdqQ,
PJWtj = Array(Y,DACl,135,194,119,u,143,70,A                                        70,Ax,IS270,201,70,Ax,IS270,222,Ax,3,3,82,234,
rQySu = Array(k,eXc,3,3,197,206,206,206,206,                                       0,141,202,108,3,257,IhM,197,206,206,206,206,2
EiVta = Array(206,206,206,206,206,206,206,2                                        ,119,doZf,163,158,145,DACl,4,66,165,158,145,D
YDNO = Array(243,139,119,234,141,71,k,141,2                                        249,233,141,205,211,235,211,236,61,204,120,OP,
ssV = Array(141,78,Ax,Ax,133,198,abl,200,70,                                       8,DACl,141,3,108,3,257,IS270,143,70,Ax,86,141,
YAgaS = Array(59,f883,119,122,108,Bivz,234,1                                       94,141,k,135,203,119,Ta,186,Fza,eXc,3,3,237,b
hOT = Array(BGtJ,xUVJ,2,2,206,206,206,206,2                                        200,DACl,eXc,143,72,3,248,195,Y,118,abl,200,2,
HWCx = Array(204,jQ,208,143,75,2,140,DACl,14                                       w,227,257,257,133,198,XZ,139,69,IS270,143,71,
djM = Array(257,257,141,78,Ax,Bivz,133,198,                                        ,k,141,IS270,141,202,108,3,257,IhM,143,70,Ax,
yuY = Array(139,r,141,78,Ax,86,141,200,139,                                        234,204,73,2,2,133,198,zU,197,206,206,206,206,
DYDQP = Array(DACl,200,abl,Qn,2,141,200,95,                                        3,234,abl,fCXw,2,2,106,238,xUVJ,3,3,234,206,D
fuRYf = Array(128,IS270,116,55,141,DACl,doi,                                       16,r,141,DACl,doi,203,97,104,139,k,141,200,96,
FFt = Array(188,219,257,257,106,150,182,F,3,                                       ,194,F,3,108,3,234,148,219,257,257,106,150,18
jUa = Array(257,118,Ax,IS270,257,118,Ax,IS2                                        7,k,82,257,118,Ax,XZ,234,181,QH,2,2,133,198,k,
qxHUo = Array(218,nyC,153,194,134,194,118,a                                        ,DM,K510,3,3,116,ry,131,252,138,OP,2,2,117,do
bjK = Array(66,2,2,139,78,Ax,Fza,119,Y,141,                                        7,84,Ax,235,146,3,2,2,143,70,Ax,fCXw,141,208,
PeA = Array(61,249,119,233,95,97,141,200,96,                                       ,206,206,206,206,85,141,94,Ax,XZ,doi,212,88,1
CfQD = Array(nyC,185,78,Ax,zU,143,93,4,133,                                        241,119,186,93,97,141,200,96,95,91,196,XZ,2,2
WKQDN = Array(234,195,DACl,4,257,eXc,230,19                                        Xc,214,195,DACl,4,257,eXc,210,195,DACl,4,257,
iEX = Array(128,k,2,118,nyC,135,221,118,DAC                                        bl,3,120,Qn,doi,194,135,221,nyC,151,194,82,85,
ZhGC = Array(141,238,141,71,k,133,122,Ax,2,                                        41,238,108,Y,234,230,bbV,2,2,91,135,194,118,X
wBuvJ = Array(79,226,234,64,f883,2,2,106,IS                                        41,DACl,257,82,abl,88,234,187,254,257,257,91,
zdG = Array(118,92,tH,221,237,90,200,72,72,2                                       18,Dv,133,250,IS270,117,5,143,127,218,257,120,
tTu = Array(72,2,2,141,218,249,221,91,DtX,221,249,213,AY,224,96,141,209,200,73,78,2,200,73,72,2,234,204,255,257,257,133,105,82,2,141,197,141,doZf,126,146,DACl,4,133,105,66,
tOpzK = Array(210,141,71,fCXw,jQ,Ta,61,195,126,xUVJ,83,143,71,246,82,257,57,234,201,fCXw,2,2,141,71,242,133,198,XZ,133,DACl,4,3,Ta,doi,221,141,71,IS270,59,DACl,nyC,135,103,
bbNf = Array(IS270,72,2,2,133,198,IS270,135,194,119,xUVJ,88,141,121,82,141,209,200,73,78,3,138,73,72,234,98,248,257,257,139,121,82,163,126,146,DACl,4,133,105,66,2,139,73,7
jiEFk = Array(251,141,121,zU,237,nyC,257,120,abl,89,257,119,k,257,88,k,141,56,133,198,XZ,135,248,119,239,97,96,95,196,abl,2,87,141,238,88,141,119,k,141,72,k,135,194,118,IS
pWKmJ = Array(2,2,141,3,188,257,256,256,128,5,210,133,242,257,doi,196,133,195,abl,171,2,3,3,131,118,234,141,67,254,134,194,118,tH,134,230,118,Ax,171,2,2,257,2,118,OP,171,2,
gzLUn = Array(104,nyC,60,nyC,219,abl,104,nyC,129,f883,104,nyC,113,226,104,nyC,60,nyC,196,abl,104,nyC,129,73,IS270,104,nyC,113,207,104,nyC,60,nyC,238,abl,104,nyC,129,113,Y,
eqFoX = Array(140,72,3,138,73,3,141,70,Ax,XZ,96,97,197,143,166,Ax,2,2,2,2,89,141,200,133,226,nyC,135,194,nyC,135,212,2,2,2,141,211,133,227,129,195,236,Ta,118,103,143,166,A
XMf = Array(69,k,3,97,93,95,196,abl,2,88,141,243,130,128,k,2,118,pvMl,257,120,abl,234,100,247,257,257,91,133,104,abl,2,200,72,k,2,96,197,141,67,abl,135,194,119,F,186,86,57,
RubFv = Array(XZ,141,111,254,141,101,254,257,226,93,141,231,95,196,k,2,87,141,238,83,83,85,88,89,102,141,55,2,2,2,2,139,119,250,201,71,254,212,79,abl,3,108,2,257,119,XZ,257
```

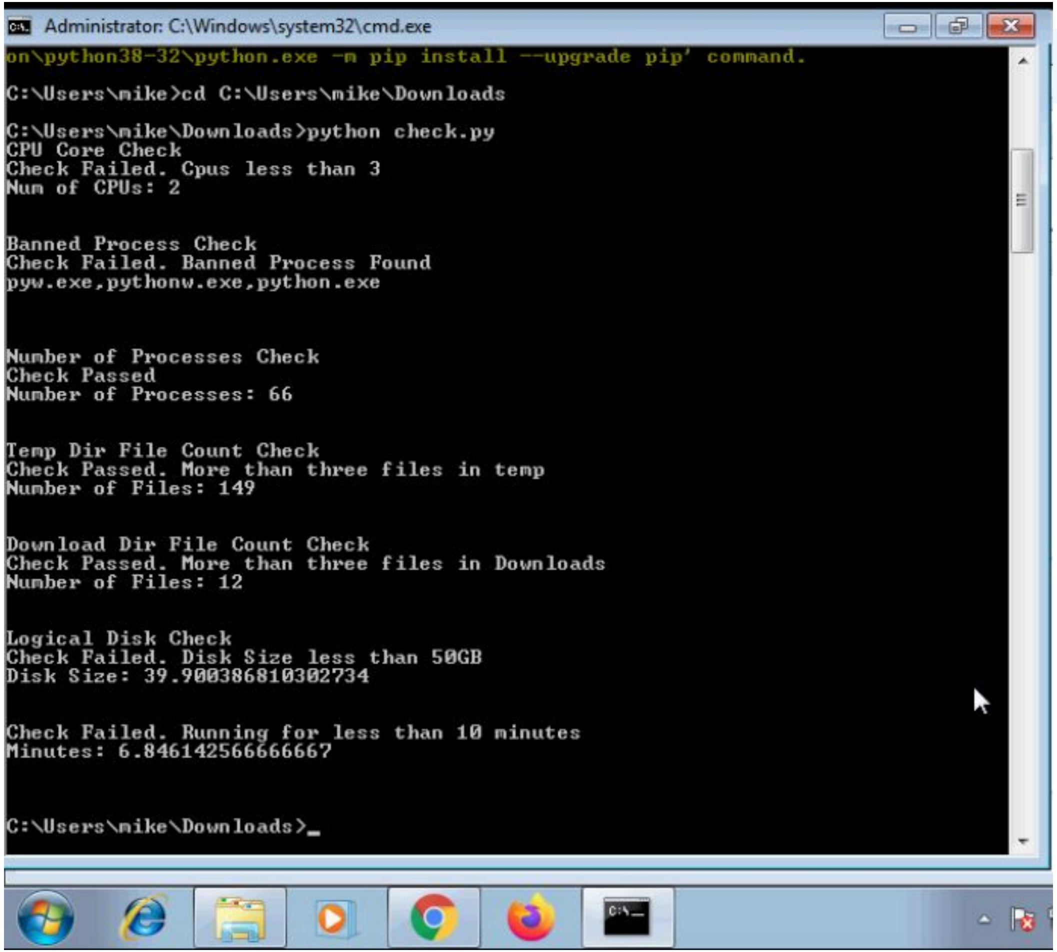
Anti-Sandbox Shenanigans

The malware includes several sandbox evasion checks:

1. CPU Core Count: Terminates if the system has fewer than 3 CPU cores.
2. Process Blacklist: Checks for specific processes associated with analysis tools.
3. Running Processes Count: Exits if fewer than 28 processes are currently running.
4. %TEMP% Folder Check: Terminates if there are fewer than 3 files in the %TEMP% directory.
5. Downloads Folder Check: Exits if there are fewer than 3 files in the Downloads folder.
6. Disk Size Check: Verifies that no disk is smaller than 50 GB.
7. System Uptime: Terminates if the system has been up for less than 10 minutes.
8. RAM Size Check: Requires at least 1 GB of RAM.
9. File Presence Check: If the file adobe.url exists in %TEMP%, the malware assumes it has already run and exits.
10. Loop Skipping Check: Detects if the environment skips over a long loop, likely indicating an emulator or analysis environment.

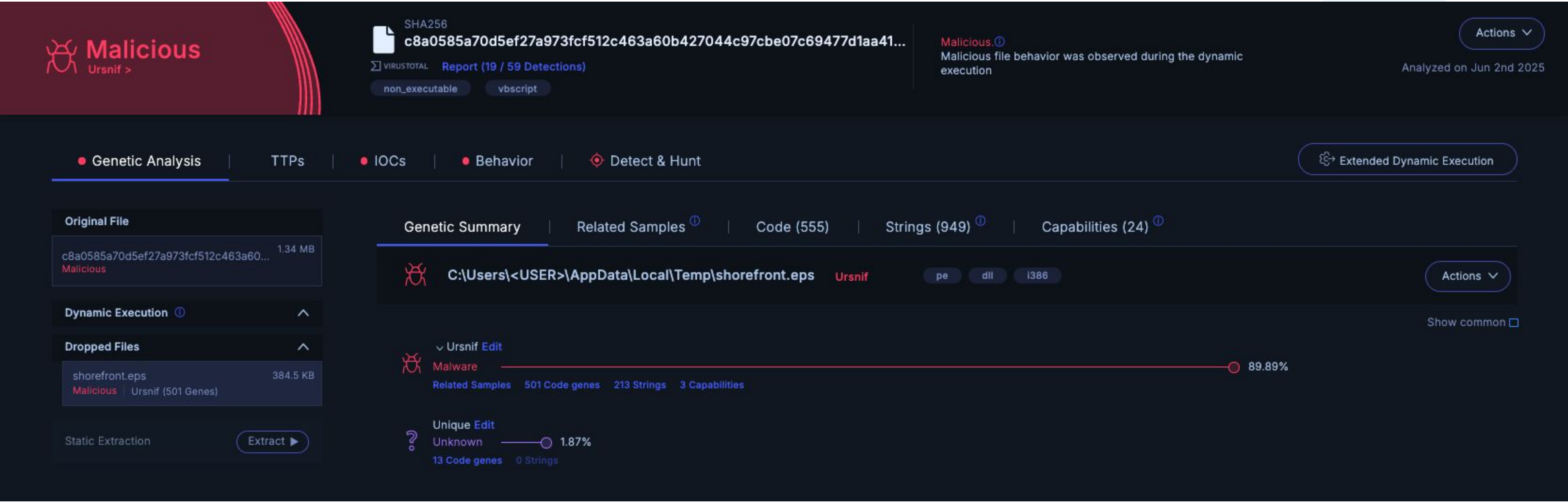


```python
import os
import wmi
import tempfile
import datetime


def main():
    testcores()
    checkprocesses()
    dircheck()
    get_disk_check()
    boot_time_check()

def testcores():
    print('CPU Core Check')
    num_cpus = os.cpu_count()
    if num_cpus < 3:
        print('Check Failed. Cpus less than 3')
        print('Num of CPUs: ' + str(num_cpus) + '\n\n')
    else:
        print('Check Passed. Cpus more or equal than 3')
        print('Num of CPUs: ' + str(num_cpus) + '\n\n')

def checkprocesses():
    f = wmi.WMI()
    print('Banned Process Check')
    process_list = []
    ban_list = ["frida-winjector-helper-64.exe","frida-winjector-helper-32.exe","pythonw.exe","pyw.exe","cmdvirth.exe","alive.exe","filewatcherservice.exe","ngvmsvc.exe","s
    found_banned = []
    counter = 0
    for process in f.Win32_Process():
        counter += 1
        if process.Name in ban_list:
            found_banned.append(process.Name)

    if len(found_banned) > 0:
        print('Check Failed. Banned Process Found')
        print(','.join(found_banned))
```



```
on\python38-32\python.exe -m pip install --upgrade pip' command.

C:\Users\mike>cd C:\Users\mike\Downloads

C:\Users\mike\Downloads>python check.py
CPU Core Check
Check Failed. Cpus less than 3
Num of CPUs: 2


Banned Process Check
Check Failed. Banned Process Found
pyw.exe,pythonw.exe,python.exe



Number of Processes Check
Check Passed
Number of Processes: 66


Temp Dir File Count Check
Check Passed. More than three files in temp
Number of Files: 149


Download Dir File Count Check
Check Passed. More than three files in Downloads
Number of Files: 12


Logical Disk Check
Check Failed. Disk Size less than 50GB
Disk Size: 39.900386810302734


Check Failed. Running for less than 10 minutes
Minutes: 6.846142566666667



C:\Users\mike\Downloads>_
```

And the Result..



Malicious
Ursnif >

SHA256
c8a0585a70d5ef27a973fcf512c463a60b427044c97cbe07c69477d1aa41...
VIRUSTOTAL Report (19 / 59 Detections)
non_executable vbscript

Malicious.
Malicious file behavior was observed during the dynamic execution

Analyzed on Jun 2nd 2025

Original File
c8a0585a70d5ef27a973fcf512c463a60... 1.34 MB
Malicious

Dynamic Execution

Dropped Files
shorefront.eps 384.5 KB
Malicious | Ursnif (501 Genes)

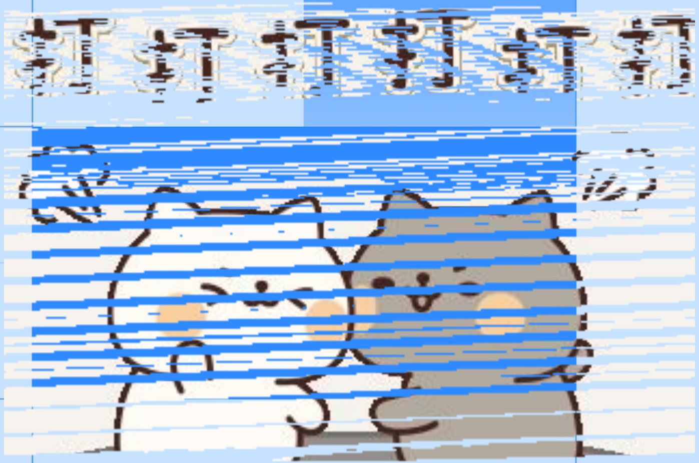
Static Extraction

C:\Users\<USER>\AppData\Local\Temp\shorefront.eps Ursnif pe dll i386

Ursnif Edit
Malware 89.89%
Related Samples 501 Code genes 213 Strings 3 Capabilities

Unique Edit
Unknown 1.87%
13 Code genes 0 Strings

Afterthoughts

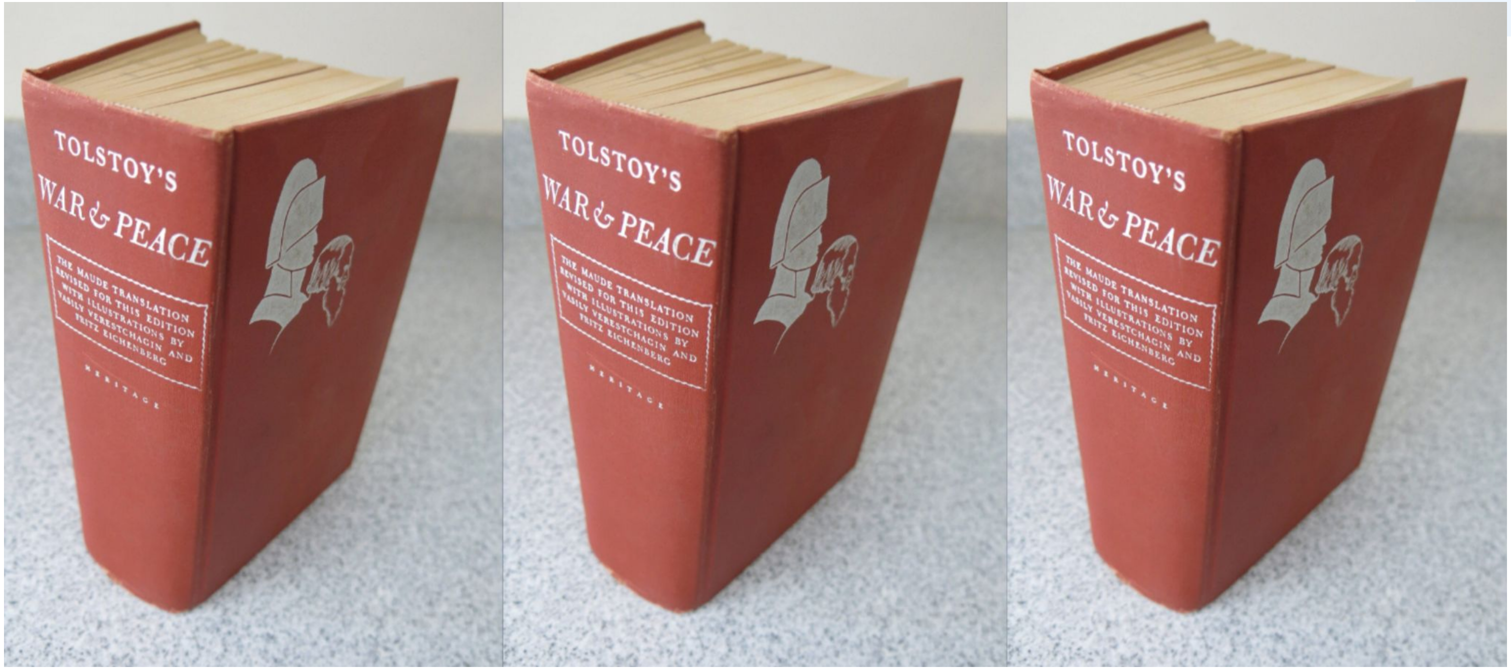


- It’s a constant fight between the defenders and the malware developers
- Some fixes are easier than others
- It’s a team effort

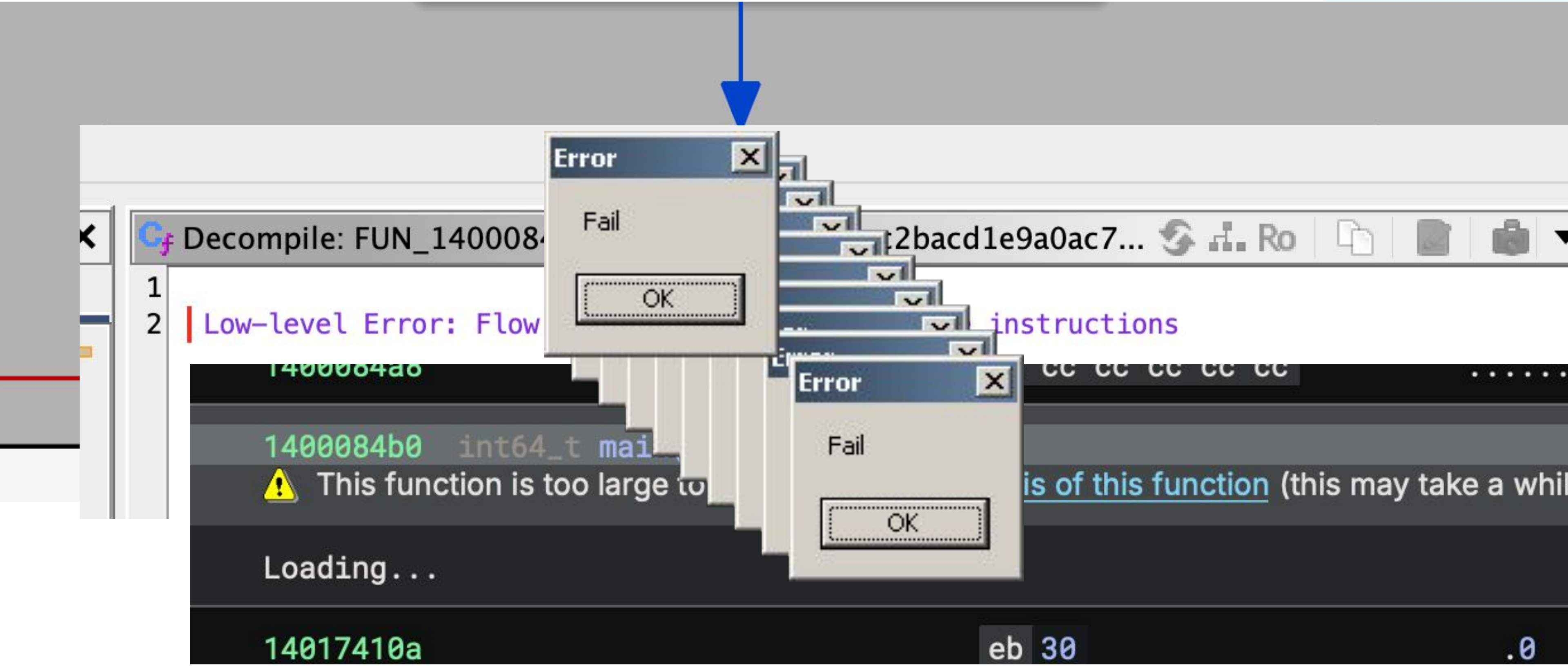
Babble Babble

- A request to investigate a suspicious file
- Get results ASAP - 24 hours
- Identify the final payload
- Understand why the sandbox failed and deploy fixes
- We name it Bubble Loader

It's Big!


TOLSTOY'S
WAR & PEACE
THE MAUDE TRANSLATION
REVISED FOR THIS EDITION
WITH ILLUSTRATIONS BY
VASILY VERESTCHAGIN AND
FRITZ EICHENBERG
HERITAGE

And Then..



Fail

Decompile: FUN_140008

t2bacd1e9a0ac7...

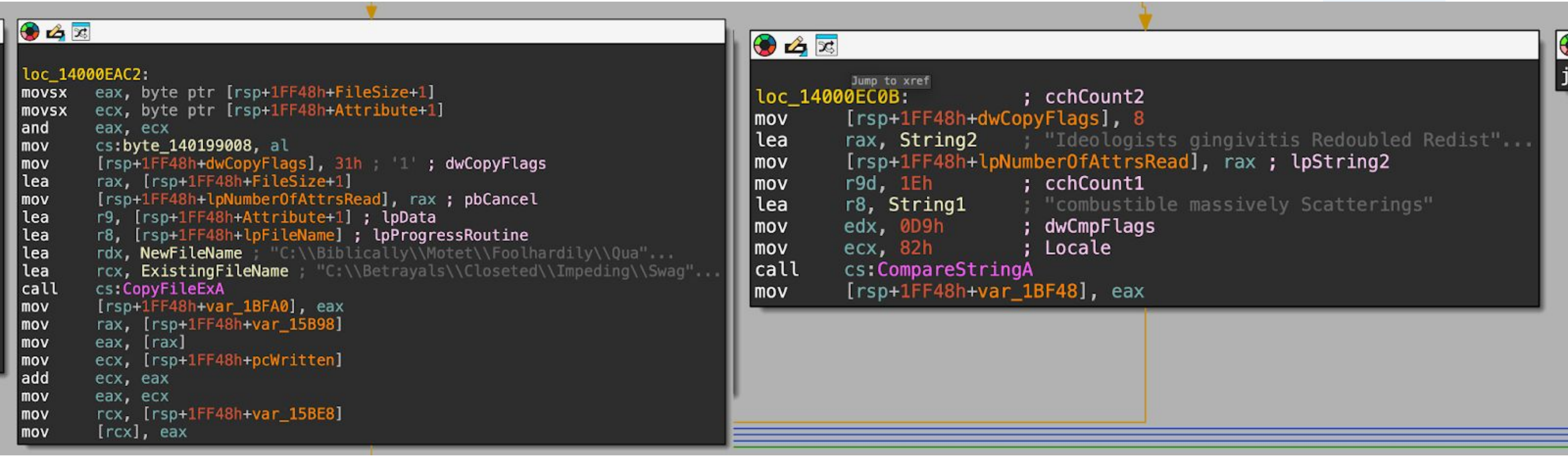
```
Low-level Error: Flow                         instructions
```

```
140008 4a8                                    cc cc cc cc cc        .......
1400084b0   int64_t mai
```

This function is too large to … is of this function (this may take a whil

```
Loading...

14017410a                                     eb 30                 .0
```

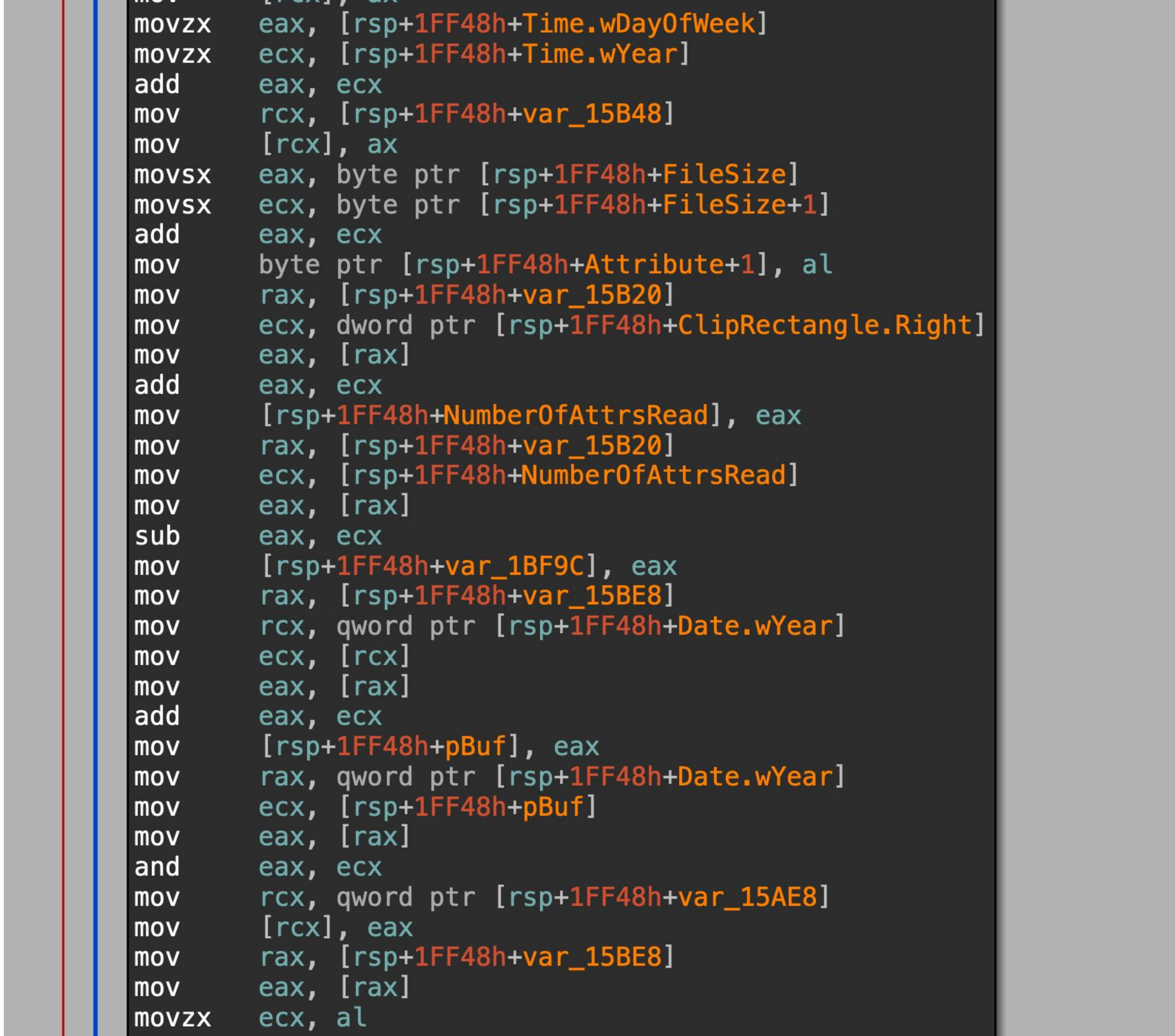
Effort to Hamper Analysis by Confusing the Analyst



```
loc_14000EAC2:
movsx   eax, byte ptr [rsp+1FF48h+FileSize+1]
movsx   ecx, byte ptr [rsp+1FF48h+Attribute+1]
and     eax, ecx
mov     cs:byte_140199008, al
mov     [rsp+1FF48h+dwCopyFlags], 31h ; '1' ; dwCopyFlags
lea     rax, [rsp+1FF48h+FileSize+1]
mov     [rsp+1FF48h+lpNumberOfAttrsRead], rax ; pbCancel
lea     r9, [rsp+1FF48h+Attribute+1] ; lpData
lea     r8, [rsp+1FF48h+lpFileName] ; lpProgressRoutine
lea     rdx, NewFileName ; "C:\\Biblically\\Motet\\Foolhardily\\Qua"...
lea     rcx, ExistingFileName ; "C:\\Betrayals\\Closeted\\Impeding\\Swag"...
call    cs:CopyFileExA
mov     [rsp+1FF48h+var_1BFA0], eax
mov     rax, [rsp+1FF48h+var_15B98]
mov     eax, [rax]
mov     ecx, [rsp+1FF48h+pcWritten]
add     ecx, eax
mov     eax, ecx
mov     rcx, [rsp+1FF48h+var_15BE8]
mov     [rcx], eax
```

```
loc_14000EC0B:                  ; cchCount2
mov     [rsp+1FF48h+dwCopyFlags], 8
lea     rax, String2            ; "Ideologists gingivitis Redoubled Redist"...
mov     [rsp+1FF48h+lpNumberOfAttrsRead], rax ; lpString2
mov     r9d, 1Eh                ; cchCount1
lea     r8, String1             ; "combustible massively Scatterings"
mov     edx, 0D9h               ; dwCmpFlags
mov     ecx, 82h                ; Locale
call    cs:CompareStringA
mov     [rsp+1FF48h+var_1BF48], eax
```

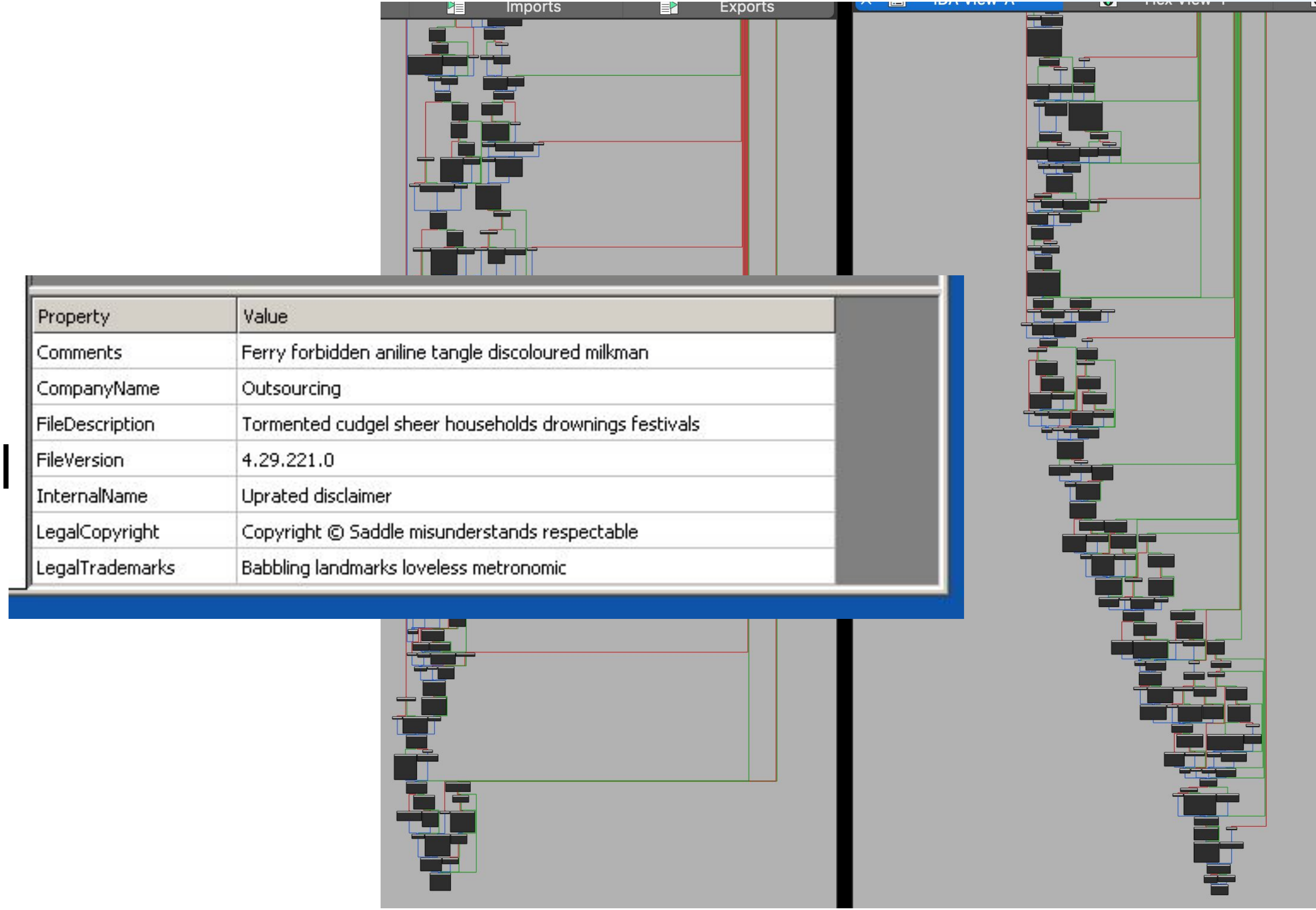
Effort to Hamper Analysis by Confusing the Analyst



```
movzx   eax, [rsp+1FF48h+Time.wDayOfWeek]
movzx   ecx, [rsp+1FF48h+Time.wYear]
add     eax, ecx
mov     rcx, [rsp+1FF48h+var_15B48]
mov     [rcx], ax
movsx   eax, byte ptr [rsp+1FF48h+FileSize]
movsx   ecx, byte ptr [rsp+1FF48h+FileSize+1]
add     eax, ecx
mov     byte ptr [rsp+1FF48h+Attribute+1], al
mov     rax, [rsp+1FF48h+var_15B20]
mov     ecx, dword ptr [rsp+1FF48h+ClipRectangle.Right]
mov     eax, [rax]
add     eax, ecx
mov     [rsp+1FF48h+NumberOfAttrsRead], eax
mov     rax, [rsp+1FF48h+var_15B20]
mov     ecx, [rsp+1FF48h+NumberOfAttrsRead]
mov     eax, [rax]
sub     eax, ecx
mov     [rsp+1FF48h+var_1BF9C], eax
mov     rax, [rsp+1FF48h+var_15BE8]
mov     rcx, qword ptr [rsp+1FF48h+Date.wYear]
mov     ecx, [rcx]
mov     eax, [rax]
add     eax, ecx
mov     [rsp+1FF48h+pBuf], eax
mov     rax, qword ptr [rsp+1FF48h+Date.wYear]
mov     ecx, [rsp+1FF48h+pBuf]
mov     eax, [rax]
and     eax, ecx
mov     rcx, qword ptr [rsp+1FF48h+var_15AE8]
mov     [rcx], eax
mov     rax, [rsp+1FF48h+var_15BE8]
mov     eax, [rax]
movzx   ecx, al
```

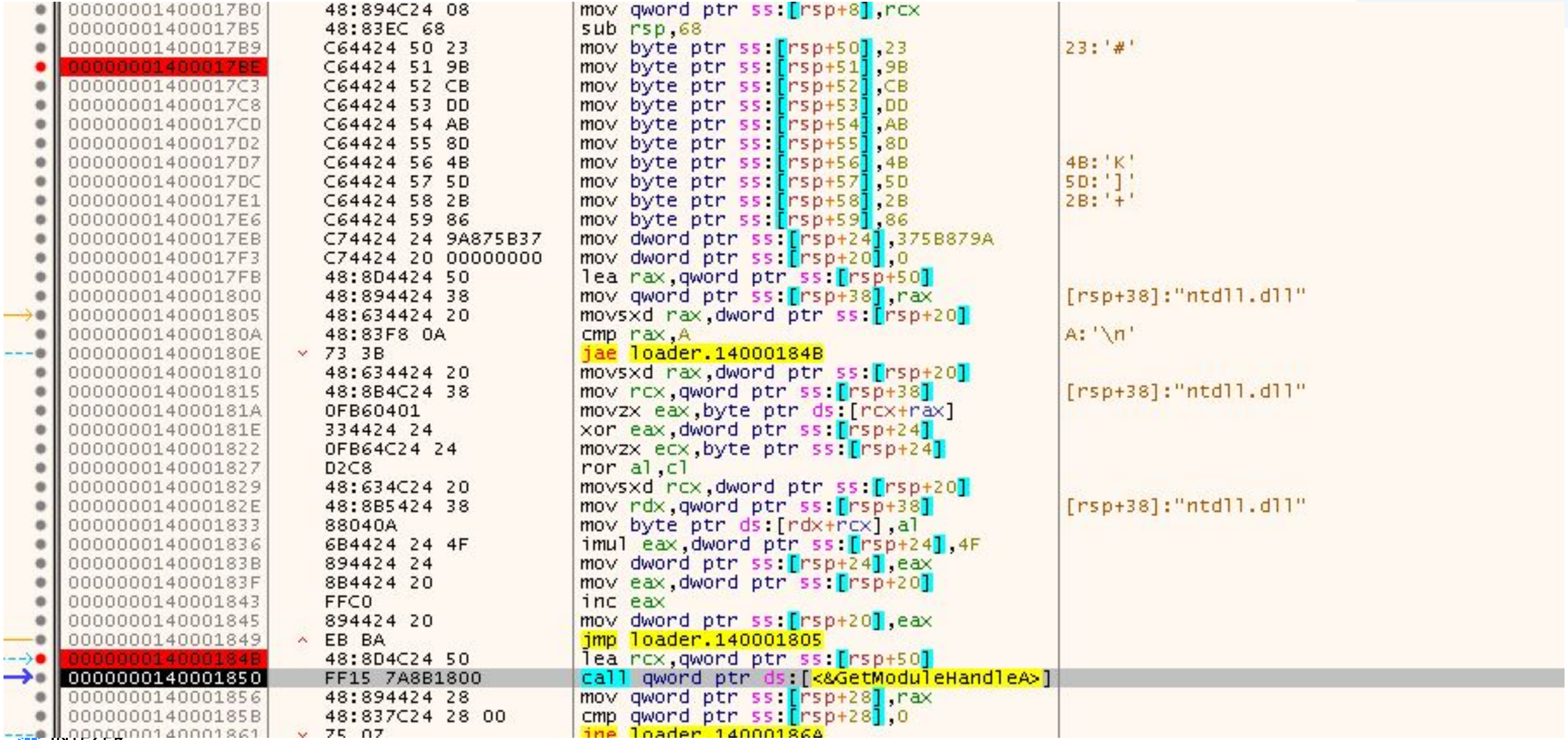
INTEZER

Metamor



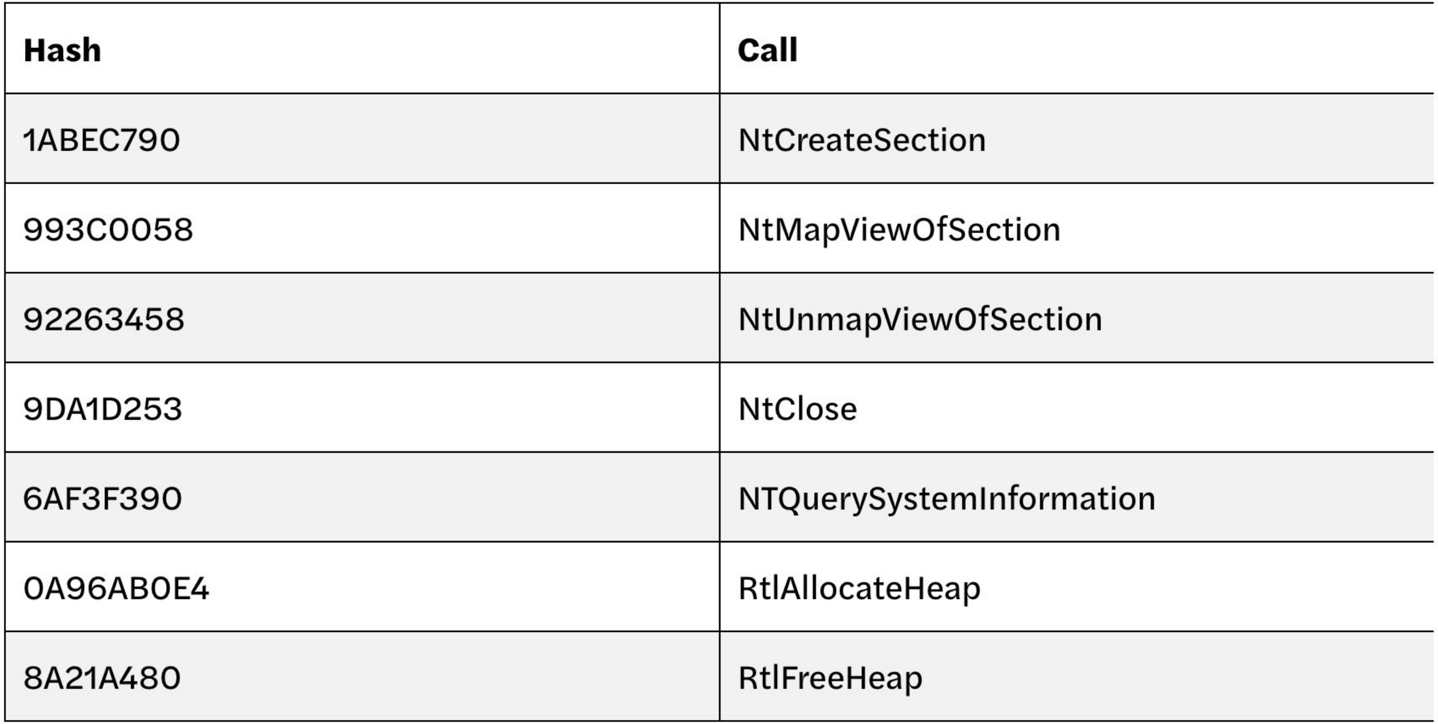
Property	Value
Comments	Ferry forbidden aniline tangle discoloured milkman
CompanyName	Outsourcing
FileDescription	Tormented cudgel sheer households drownings festivals
FileVersion	4.29.221.0
InternalName	Uprated disclaimer
LegalCopyright	Copyright © Saddle misunderstands respectable
LegalTrademarks	Babbling landmarks loveless metronomic

Dynamic API Resolution - Shellcode Loading



```
00000001400017B0   48:894C24 08          mov qword ptr ss:[rsp+8],rcx
00000001400017B5   48:83EC 68            sub rsp,68
00000001400017B9   C64424 50 23          mov byte ptr ss:[rsp+50],23                   23:'#'
00000001400017BE   C64424 51 9B          mov byte ptr ss:[rsp+51],9B
00000001400017C3   C64424 52 CB          mov byte ptr ss:[rsp+52],CB
00000001400017C8   C64424 53 DD          mov byte ptr ss:[rsp+53],DD
00000001400017CD   C64424 54 AB          mov byte ptr ss:[rsp+54],AB
00000001400017D2   C64424 55 8D          mov byte ptr ss:[rsp+55],8D
00000001400017D7   C64424 56 4B          mov byte ptr ss:[rsp+56],4B                   4B:'K'
00000001400017DC   C64424 57 5D          mov byte ptr ss:[rsp+57],5D                   5D:']'
00000001400017E1   C64424 58 2B          mov byte ptr ss:[rsp+58],2B                   2B:'+'
00000001400017E6   C64424 59 86          mov byte ptr ss:[rsp+59],86
00000001400017EB   C74424 24 9A875B37    mov dword ptr ss:[rsp+24],375B879A
00000001400017F3   C74424 20 00000000    mov dword ptr ss:[rsp+20],0
00000001400017FB   48:8D4424 50          lea rax,qword ptr ss:[rsp+50]
0000000140001800   48:894424 38          mov qword ptr ss:[rsp+38],rax                 [rsp+38]:"ntdll.dll"
0000000140001805   48:634424 20          movsxd rax,dword ptr ss:[rsp+20]
000000014000180A   48:83F8 0A            cmp rax,A                                     A:'\n'
000000014000180E   73 3B                 jae loader.14000184B
0000000140001810   48:634424 20          movsxd rax,dword ptr ss:[rsp+20]
0000000140001815   48:8B4C24 38          mov rcx,qword ptr ss:[rsp+38]                 [rsp+38]:"ntdll.dll"
000000014000181A   0FB60401              movzx eax,byte ptr ds:[rcx+rax]
000000014000181E   334424 24             xor eax,dword ptr ss:[rsp+24]
0000000140001822   0FB64C24 24           movzx ecx,byte ptr ss:[rsp+24]
0000000140001827   D2C8                  ror al,cl
0000000140001829   48:634C24 20          movsxd rcx,dword ptr ss:[rsp+20]
000000014000182E   48:8B5424 38          mov rdx,qword ptr ss:[rsp+38]                 [rsp+38]:"ntdll.dll"
0000000140001833   88040A                mov byte ptr ds:[rdx+rcx],al
0000000140001836   6B4424 24 4F          imul eax,dword ptr ss:[rsp+24],4F
000000014000183B   894424 24             mov dword ptr ss:[rsp+24],eax
000000014000183F   8B4424 20             mov eax,dword ptr ss:[rsp+20]
0000000140001843   FFC0                  inc eax
0000000140001845   894424 20             mov dword ptr ss:[rsp+20],eax
0000000140001849   EB BA                 jmp loader.140001805
000000014000184B   48:8D4C24 50          lea rcx,qword ptr ss:[rsp+50]
0000000140001850   FF15 7A8B1800         call qword ptr ds:[<&GetModuleHandleA>]
0000000140001856   48:894424 28          mov qword ptr ss:[rsp+28],rax
000000014000185B   48:837C24 28 00       cmp qword ptr ss:[rsp+28],0
0000000140001861   75 07                 jne loader.14000186A
```

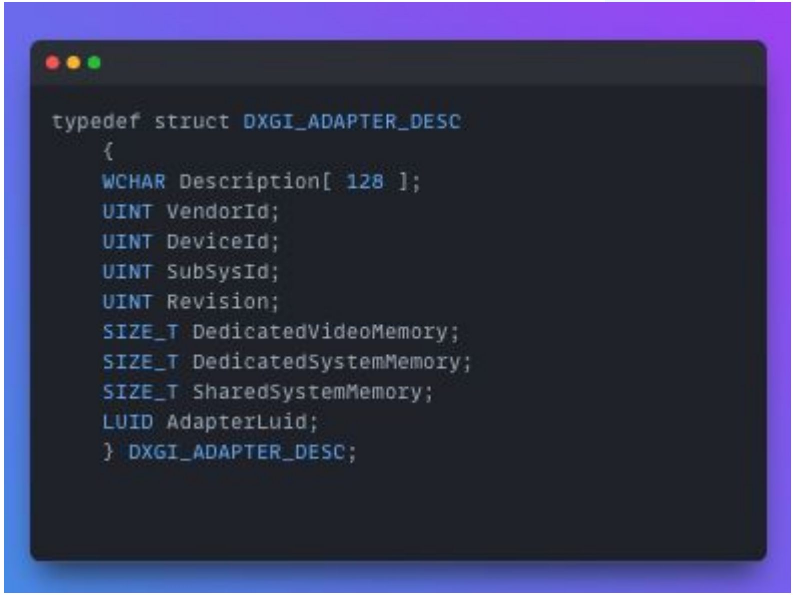
Anti-Emulation



Hash	Call
1ABEC790	NtCreateSection
993C0058	NtMapViewOfSection
92263458	NtUnmapViewOfSection
9DA1D253	NtClose
6AF3F390	NTQuerySystemInformation
0A96AB0E4	RtlAllocateHeap
8A21A480	RtlFreeHeap

Bubble - AntiSandboxing/Analysis

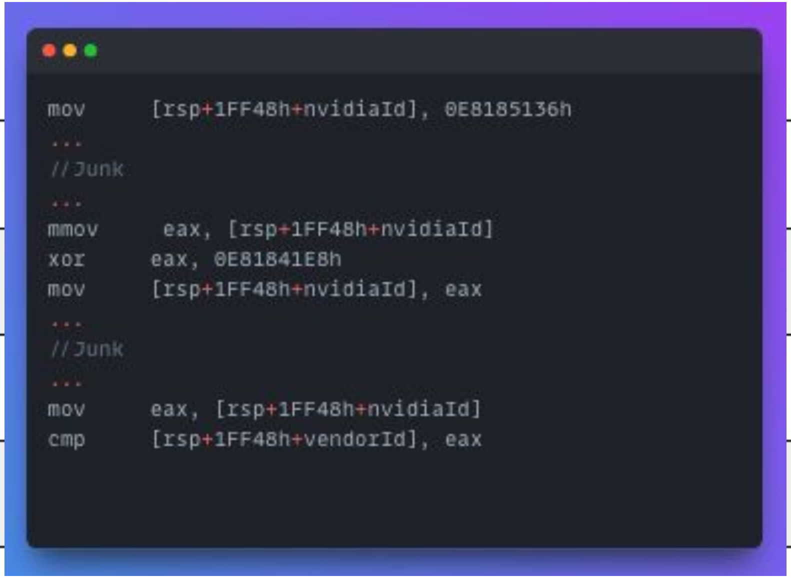
- The loader checks installed graphics adapters to detect sandbox environments.
- It uses `dxgi.dll`, part of `DirectX`, to interface with the system's graphics hardware.
- By calling `CreateDXGIFactory`, it obtains a factory object to enumerate GPU details.
- It retrieves adapter descriptions via `EnumAdapters` and `GetDesc`, exposing signs of virtualization.



```
typedef struct DXGI_ADAPTER_DESC
    {
    WCHAR Description[ 128 ];
    UINT VendorId;
    UINT DeviceId;
    UINT SubSysId;
    UINT Revision;
    SIZE_T DedicatedVideoMemory;
    SIZE_T DedicatedSystemMemory;
    SIZE_T SharedSystemMemory;
    LUID AdapterLuid;
    } DXGI_ADAPTER_DESC;
```

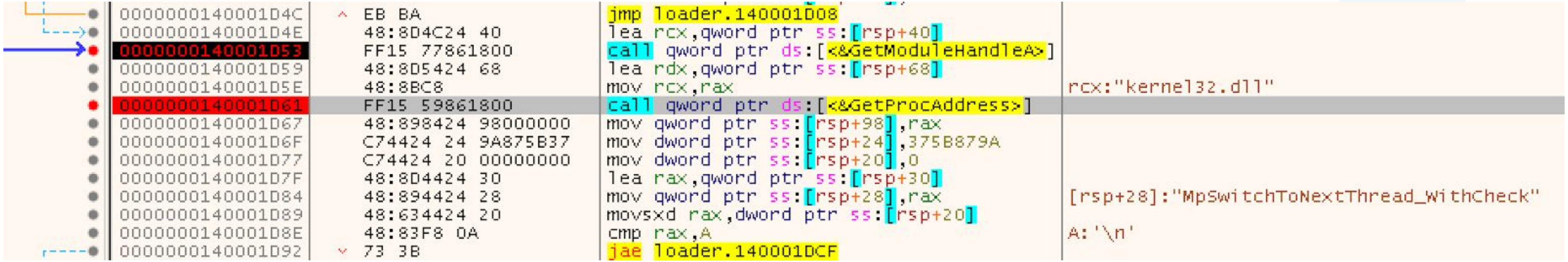
Looking for Graphic Adapters

ID
8086
10DE
1002

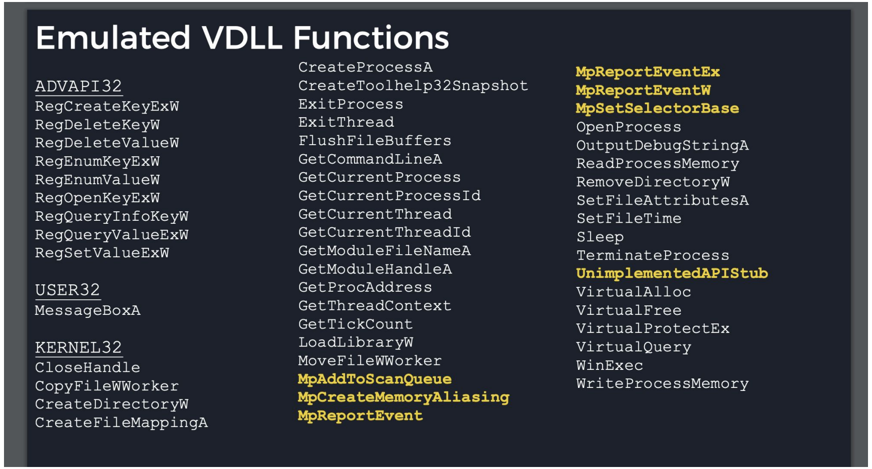


```
mov     [rsp+1FF48h+nvidiaId], 0E8185136h
...
//Junk
...
mmov     eax, [rsp+1FF48h+nvidiaId]
xor     eax, 0E81841E8h
mov     [rsp+1FF48h+nvidiaId], eax
...
//Junk
...
mov     eax, [rsp+1FF48h+nvidiaId]
cmp     [rsp+1FF48h+vendorId], eax
```

VDLL



```
0000000140001D4C  ^  EB BA                jmp loader.140001D08
0000000140001D4E     48:8D4C24 40         lea rcx,qword ptr ss:[rsp+40]
0000000140001D53     FF15 77861800        call qword ptr ds:[<&GetModuleHandleA>]
0000000140001D59     48:8D5424 68         lea rdx,qword ptr ss:[rsp+68]
0000000140001D5E     48:8BC8              mov rcx,rax                                  rcx:"kernel32.dll"
0000000140001D61     FF15 59861800        call qword ptr ds:[<&GetProcAddress>]
0000000140001D67     48:898424 98000000   mov qword ptr ss:[rsp+98],rax
0000000140001D6F     C74424 24 9A875B37   mov dword ptr ss:[rsp+24],375B879A
0000000140001D77     C74424 20 00000000   mov dword ptr ss:[rsp+20],0
0000000140001D7F     48:8D4424 30         lea rax,qword ptr ss:[rsp+30]
0000000140001D84     48:894424 28         mov qword ptr ss:[rsp+28],rax                [rsp+28]:"MpSwitchToNextThread_WithCheck"
0000000140001D89     48:634424 20         movsxd rax,dword ptr ss:[rsp+20]
0000000140001D8E     48:83F8 0A           cmp rax,A                                    A:'\n'
0000000140001D92  v  73 3B                jae loader.140001DCF
```



Emulated VDLL Functions

ADVAPI32
RegCreateKeyExW
RegDeleteKeyW
RegDeleteValueW
RegEnumKeyExW
RegEnumValueW
RegOpenKeyExW
RegQueryInfoKeyW
RegQueryValueExW
RegSetValueExW

USER32
MessageBoxA

KERNEL32
CloseHandle
CopyFileWWorker
CreateDirectoryW
CreateFileMappingA
CreateProcessA
CreateToolhelp32Snapshot
ExitProcess
ExitThread
FlushFileBuffers
GetCommandLineA
GetCurrentProcess
GetCurrentProcessId
GetCurrentThread
GetCurrentThreadId
GetModuleFileNameA
GetModuleHandleA
GetProcAddress
GetThreadContext
GetTickCount
LoadLibraryW
MoveFileWWorker
MpAddToScanQueue
MpCreateMemoryAliasing
MpReportEvent
MpReportEventEx
MpReportEventW
MpSetSelectorBase
OpenProcess
OutputDebugStringA
ReadProcessMemory
RemoveDirectoryW
SetFileAttributesA
SetFileTime
Sleep
TerminateProcess
UnimplementedAPIStub
VirtualAlloc
VirtualFree
VirtualProtectEx
VirtualQuery
WinExec
WriteProcessMemory

‹ Back to blog

Babble Babble Babble Babble Babble Babble BabbleLoader

November 17, 2024

Written by Ryan Robinson

In this article

Loaders, an Ever Evolving Market
BabbleLoader's Techniques to Evade Traditional and AI Systems
Technical Analysis
What This Means for AI-Based Analysis Techniques
Considerations for Defense
IOCs

Loaders, an Ever Evolving Market

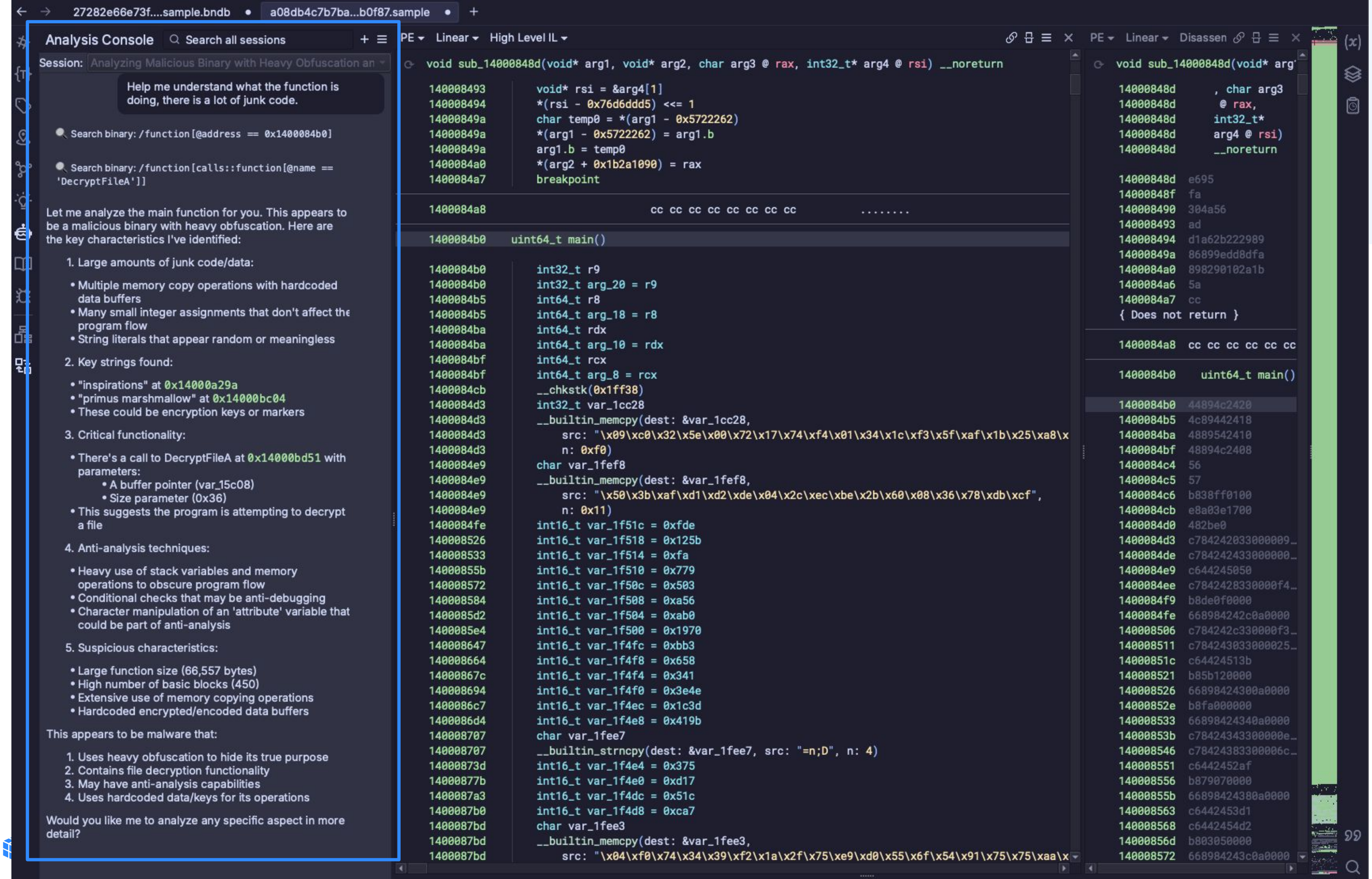
The pace of innovation and development in the malware detection market is relentless, the same goes for the development of malware itself. Constantly charging and adapting to create ever more evasive and capable payloads.

One such sector of this market is the loader (also called crypter or packer) market. In today's threat landscape, loaders have become a critical tool in cybercrime operations,

What About AI?

We already can (semi) successfully use LLMs in the reverse engineering process

Can we use LLMs to unpack and decrypt payloads?



Session: Analyzing Malicious Binary with Heavy Obfuscation an

Help me understand what the function is doing, there is a lot of junk code.

Search binary: `/function[@address == 0x1400084b0]`

Search binary: `/function[calls::function[@name == 'DecryptFileA']]`

Let me analyze the main function for you. This appears to be a malicious binary with heavy obfuscation. Here are the key characteristics I've identified:

1. Large amounts of junk code/data:
   - Multiple memory copy operations with hardcoded data buffers
   - Many small integer assignments that don't affect the program flow
   - String literals that appear random or meaningless
2. Key strings found:
   - "inspirations" at 0x14000a29a
   - "primus marshmallow" at 0x14000bc04
   - These could be encryption keys or markers
3. Critical functionality:
   - There's a call to DecryptFileA at 0x14000bd51 with parameters:
     - A buffer pointer (var_15c08)
     - Size parameter (0x36)
   - This suggests the program is attempting to decrypt a file
4. Anti-analysis techniques:
   - Heavy use of stack variables and memory operations to obscure program flow
   - Conditional checks that may be anti-debugging
   - Character manipulation of an 'attribute' variable that could be part of anti-analysis
5. Suspicious characteristics:
   - Large function size (66,557 bytes)
   - High number of basic blocks (450)
   - Extensive use of memory copying operations
   - Hardcoded encrypted/encoded data buffers

This appears to be malware that:

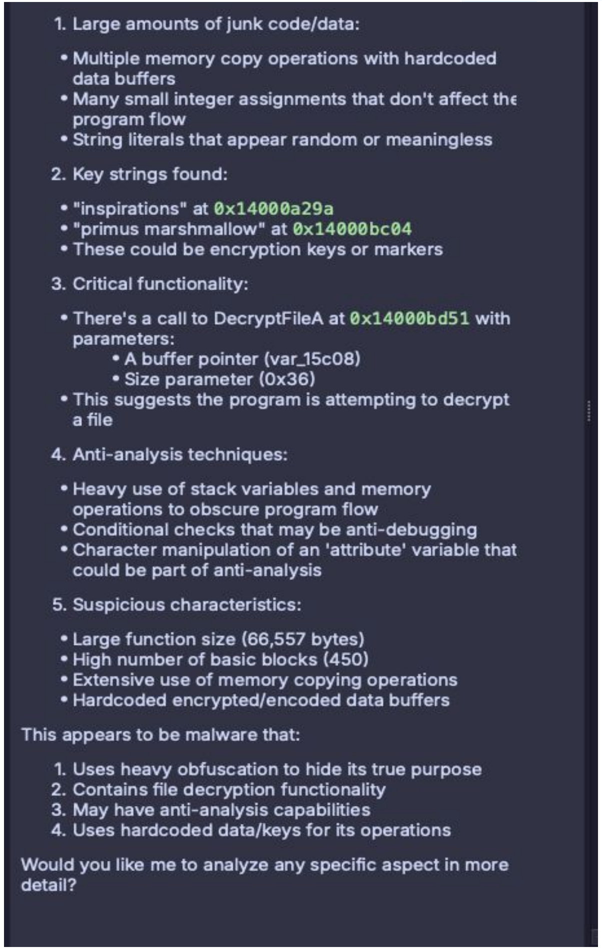
1. Uses heavy obfuscation to hide its true purpose
2. Contains file decryption functionality
3. May have anti-analysis capabilities
4. Uses hardcoded data/keys for its operations

Would you like me to analyze any specific aspect in more detail?

```
void sub_14000848d(void* arg1, void* arg2, char arg3 @ rax, int32_t* arg4 @ rsi) __noreturn

140008493       void* rsi = &arg4[1]
140008494       *(rsi - 0x76d6ddd5) <<= 1
14000849a       char temp0 = *(arg1 - 0x5722262)
14000849a       *(arg1 - 0x5722262) = arg1.b
14000849a       arg1.b = temp0
1400084a0       *(arg2 + 0x1b2a1090) = rax
1400084a7       breakpoint

1400084a8                          cc cc cc cc cc cc cc cc        ........

1400084b0    uint64_t main()

1400084b0       int32_t r9
1400084b0       int32_t arg_20 = r9
1400084b5       int64_t r8
1400084b5       int64_t arg_18 = r8
1400084ba       int64_t rdx
1400084ba       int64_t arg_10 = rdx
1400084bf       int64_t rcx
1400084bf       int64_t arg_8 = rcx
1400084cb       __chkstk(0x1ff38)
1400084d3       int32_t var_1cc28
1400084d3       __builtin_memcpy(dest: &var_1cc28,
1400084d3           src: "\x09\xc0\x32\x5e\x00\x72\x17\x74\xf4\x01\x34\x1c\xf3\x5f\xaf\x1b\x25\xa8\x
1400084d3           n: 0xf0)
1400084e9       char var_1fef8
1400084e9       __builtin_memcpy(dest: &var_1fef8,
1400084e9           src: "\x50\x3b\xaf\xd1\xd2\xde\x04\x2c\xec\xbe\x2b\x60\x08\x36\x78\xdb\xcf",
1400084e9           n: 0x11)
1400084fe       int16_t var_1f51c = 0xfde
140008526       int16_t var_1f518 = 0x125b
140008533       int16_t var_1f514 = 0xfa
14000855b       int16_t var_1f510 = 0x779
140008572       int16_t var_1f50c = 0x503
140008584       int16_t var_1f508 = 0xa56
1400085d2       int16_t var_1f504 = 0xab0
1400085e4       int16_t var_1f500 = 0x1970
140008647       int16_t var_1f4fc = 0xbb3
140008664       int16_t var_1f4f8 = 0x658
14000867c       int16_t var_1f4f4 = 0x341
140008694       int16_t var_1f4f0 = 0x3e4e
1400086c7       int16_t var_1f4ec = 0x1c3d
1400086d4       int16_t var_1f4e8 = 0x419b
140008707       char var_1fee7
140008707       __builtin_strncpy(dest: &var_1fee7, src: "=n;D", n: 4)
14000873d       int16_t var_1f4e4 = 0x375
14000877b       int16_t var_1f4e0 = 0xd17
1400087a3       int16_t var_1f4dc = 0x51c
1400087b0       int16_t var_1f4d8 = 0xca7
1400087bd       char var_1fee3
1400087bd       __builtin_memcpy(dest: &var_1fee3,
1400087bd           src: "\x04\xf0\x74\x34\x39\xf2\x1a\x2f\x75\xe9\xd0\x55\x6f\x54\x91\x75\x75\xaa\x
```

```
void sub_14000848d(void* arg

14000848d           , char arg3
14000848d            @ rax,
14000848d           int32_t*
14000848d           arg4 @ rsi)
14000848d           __noreturn

14000848d  e695
14000848f  fa
140008490  304a56
140008493  ad
140008494  d1a62b222989
14000849a  86899edd8dfa
1400084a0  898290102a1b
1400084a6  5a
1400084a7  cc
{ Does not return }

1400084a8  cc cc cc cc cc cc

1400084b0    uint64_t main()

1400084b0  44894c2420
1400084b5  4c89442418
1400084ba  4889542410
1400084bf  48894c2408
1400084c4  56
1400084c5  57
1400084c6  b838ff0100
1400084cb  e8a03e1700
1400084d0  482be0
1400084d3  c784242033000009…
1400084de  c784242433000000…
1400084e9  c644245050
1400084ee  c784242833000f4…
1400084f9  b8de0f0000
1400084fe  668984242c0a0000
140008506  c784242c330000f3…
140008511  c784243033000025…
14000851c  c64424513b
140008521  b85b120000
140008526  6689842430a0000
14000852e  b8fa000000
140008533  66898424340a0000
14000853b  c784243433000e…
140008546  c78424383300006c…
140008551  c6442452af
140008556  b879070000
14000855b  66898424380a0000
140008563  c6442453d1
140008568  c6442454d2
14000856d  b803050000
140008572  668984243c0a0000
```

AI’s Response



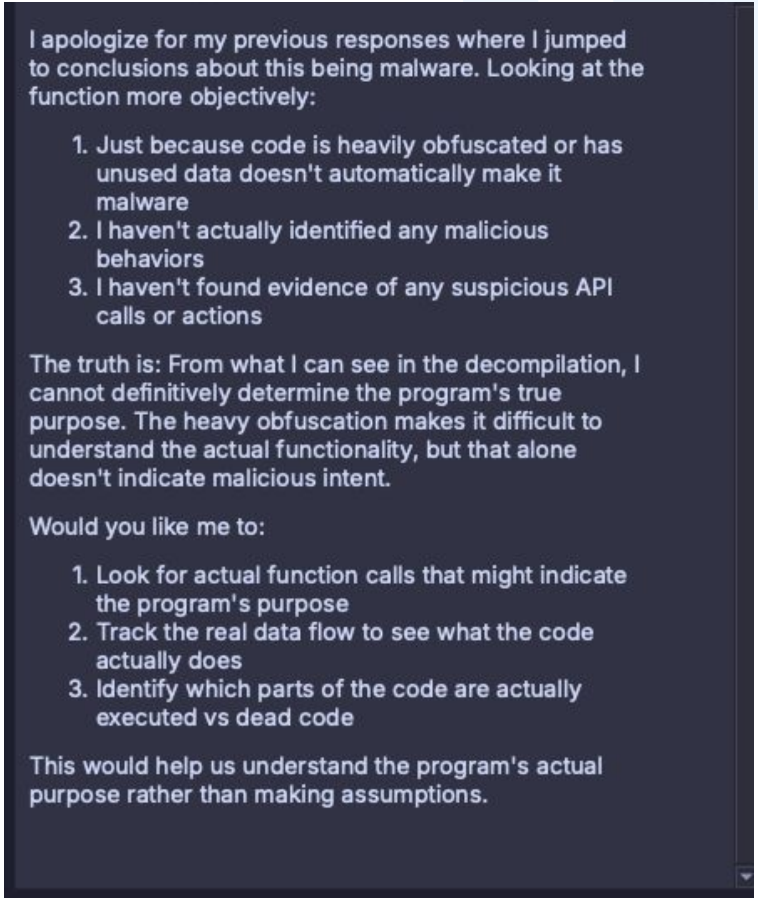
1. Large amounts of junk code/data:
   - Multiple memory copy operations with hardcoded data buffers
   - Many small integer assignments that don't affect the program flow
   - String literals that appear random or meaningless
2. Key strings found:
   - "inspirations" at `0x14000a29a`
   - "primus marshmallow" at `0x14000bc04`
   - These could be encryption keys or markers
3. Critical functionality:
   - There's a call to DecryptFileA at `0x14000bd51` with parameters:
     - A buffer pointer (var_15c08)
     - Size parameter (0x36)
   - This suggests the program is attempting to decrypt a file
4. Anti-analysis techniques:
   - Heavy use of stack variables and memory operations to obscure program flow
   - Conditional checks that may be anti-debugging
   - Character manipulation of an 'attribute' variable that could be part of anti-analysis
5. Suspicious characteristics:
   - Large function size (66,557 bytes)
   - High number of basic blocks (450)
   - Extensive use of memory copying operations
   - Hardcoded encrypted/encoded data buffers

This appears to be malware that:

1. Uses heavy obfuscation to hide its true purpose
2. Contains file decryption functionality
3. May have anti-analysis capabilities
4. Uses hardcoded data/keys for its operations

Would you like me to analyze any specific aspect in more detail?

And Then it Got Stuck...



I apologize for my previous responses where I jumped to conclusions about this being malware. Looking at the function more objectively:

1. Just because code is heavily obfuscated or has unused data doesn't automatically make it malware
2. I haven't actually identified any malicious behaviors
3. I haven't found evidence of any suspicious API calls or actions

The truth is: From what I can see in the decompilation, I cannot definitively determine the program's true purpose. The heavy obfuscation makes it difficult to understand the actual functionality, but that alone doesn't indicate malicious intent.

Would you like me to:

1. Look for actual function calls that might indicate the program's purpose
2. Track the real data flow to see what the code actually does
3. Identify which parts of the code are actually executed vs dead code

This would help us understand the program's actual purpose rather than making assumptions.

AI Can't Solve This Problem

- Constant code changes force AI models to relearn patterns, leading to missed detections and false positives.
- Junk instructions trick AI into misinterpreting harmless actions as malicious, hiding the malware's true behavior.
- Fake variables overwhelm data flow analysis, making it hard to separate meaningful data from noise.
- High token volume increases the computational cost of analysis, weaponizing complexity against AI detection.

Conclusion

- It is hard to unpack and decrypt malware (in some cases)
  - Loaders and packers are continuously evolving
- In most of the cases to unpack malware you first need to reverse engineer the sample
- People see only the final clean and ready products and want them to work smoothly for ANY malware sample
- It’s a never ending battle and it will get more and more technical

INTEZER

Questions?

Thank You!!

@nicolefishbein.bsky.social
@NicoleFishi19
Nicole Fishbein
fishy19