Please note this event may be photographed

Introduction to Binary Exploitation

Part 2: Stack Overflows

The Legal Bit

- The skills taught in these sessions allow identification and exploitation of security vulnerabilities in systems. We strive to give you a place to practice legally, and can point you to other places to practice. These skills should not be used on systems where you do not have explicit permission from the owner of the system. It is **VERY** easy to end up in breach of relevant laws, and we can accept no responsibility for anything you do with the skills learnt here.
- If we have reason to believe that you are utilising these skills against systems where you are not authorised you will be banned from our events, and if necessary the relevant authorities will be alerted.
- Remember, if you have any doubts as to if something is legal or authorised, just don't do it until you are able to confirm you are allowed to.

Code of Conduct

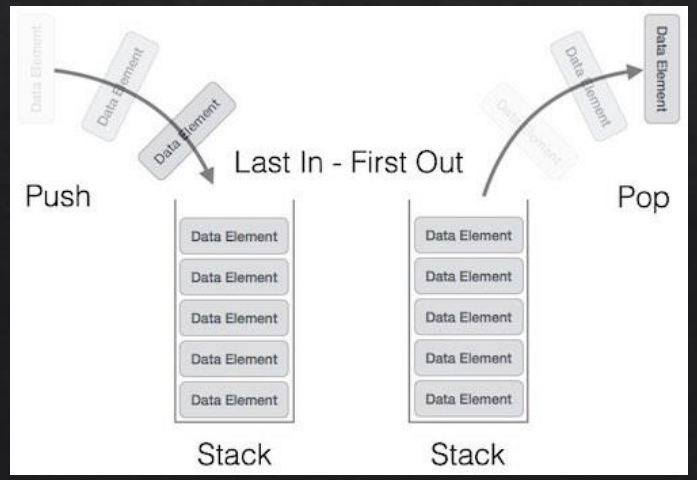
- Before proceeding past this point you must read and agree our Code of Conduct, this is a requirement from the University for us to operate as a society.
- If you have any doubts or need anything clarified, please ask a member of the committee.
- Breaching the Code of Conduct = immediate ejection and further consequences.

- Code of Conduct can be found at https://wiki.shefesh.com/doku.php?id=code_conduct

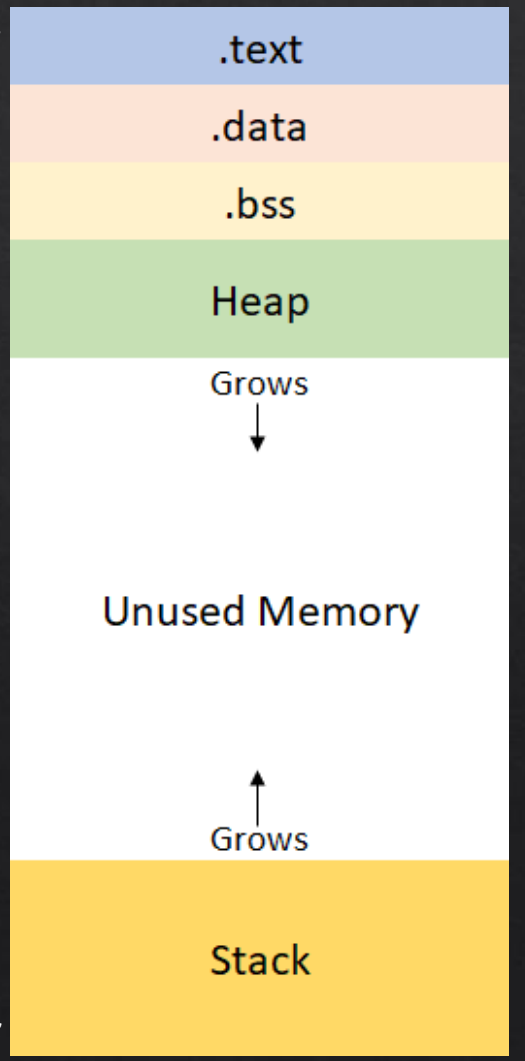
Stack Overflows

Stack Overflows - Remember The Stack?

- Data in program functions
- Last in First out (LIFO)
- Grows high to low
- Data is in stack frames
- And now we're going to exploit them!

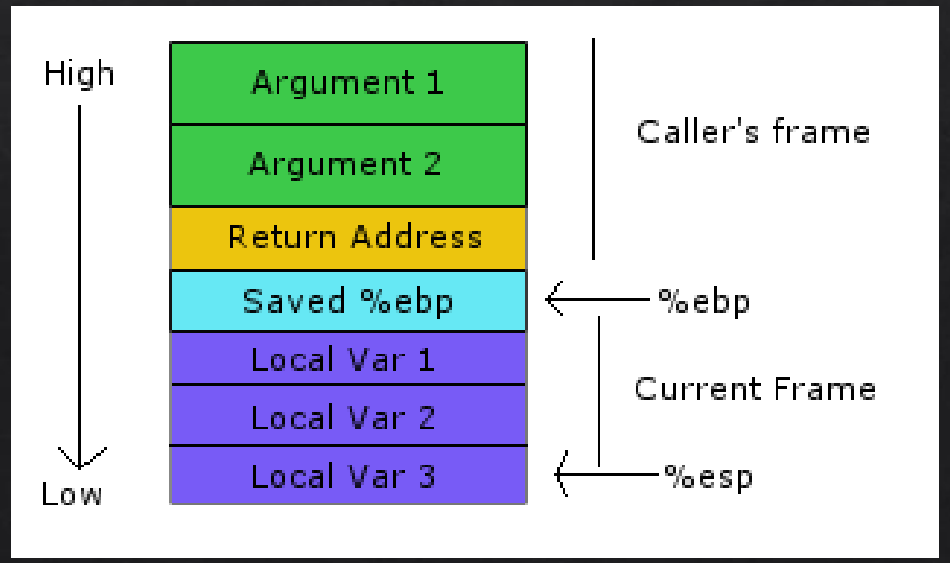


https://www.tutorialspoint.com/data_structures_algorithms/stack_algorithm.htm



Stack Overflows - Stack Frame Layout

- The local variables are on the stack frame
- Along with the return address
- Return address ends up in EIP register (The one that controls execution)
- Remember, data will be PUSHED onto the stack
- E.g arg1 went on first, local var 3 last
- Can anyone guess the potential problem here?

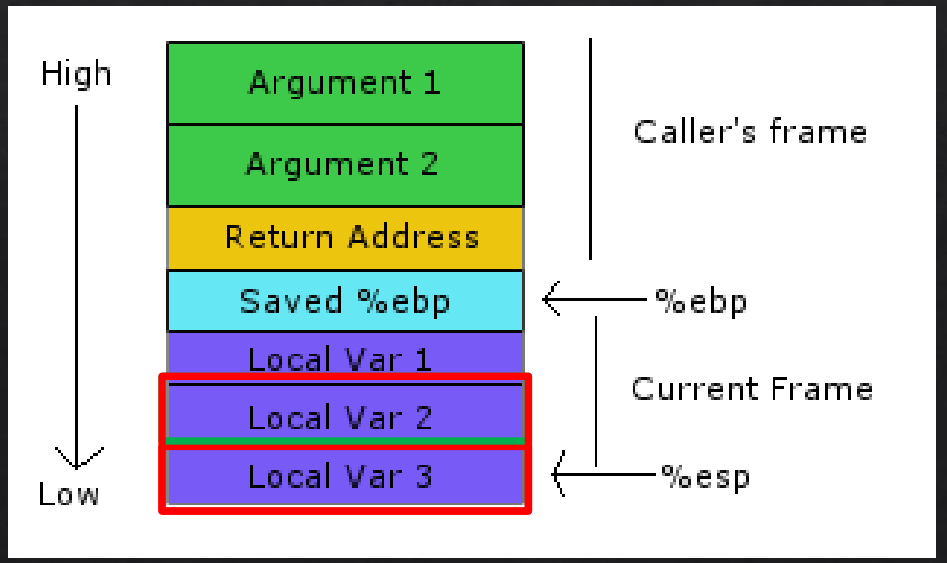


https://i.stack.imgur.com/XDoh3.png

Stack Overflows - Remember Buffers?

- This first pushes space (4 bytes) for the int to the frame
- Then 16 bytes for the char buffer
- What if we give it more than 16 bytes?
- If we imagine that "number" is "local var 2"
- And "buf" is "local var 3"
- What if we give "buf" 20 bytes of data?
- It will ***OVERFLOW*** into the space for "number"
- Anyone see where this is going?

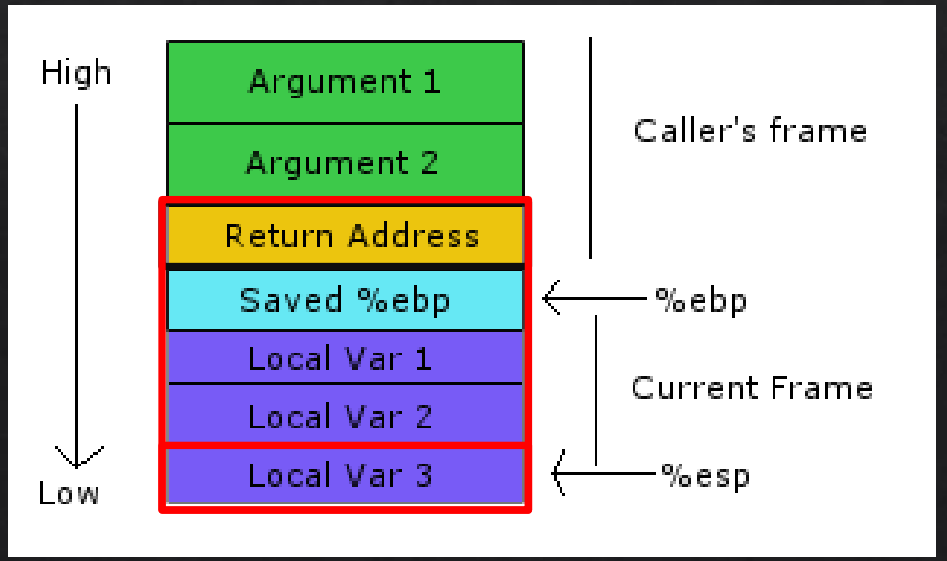
```
int number;
char buf[16];
```



https://i.stack.imgur.com/XDoh3.png

Stack Overflows -Owning The Program

- If we can control EIP
- We can control the next instruction
- And redirect execution
- EIPs value comes from the stack
- The "Return Address"
- If we overflow a variable enough
- We can overwrite EIP
- And then we own the program
- That is a stack overflow…. But what do we do with it?



https://i.stack.imgur.com/XDoh3.png

Stack Overflows - Controlling The Program

- Now we can redirect execution
- We need something to redirect it too
- We may want to call a function already in the program?
- But more often than not, we pair with shellcode
- Which we will look at now!

Shellcode

Shellcode - What is it?

- Shellcode is the payload we want to execute
- We redirect execution into the shellcode to run it
- Often the hex representation of instructions
- This executes execve to spawn a shell

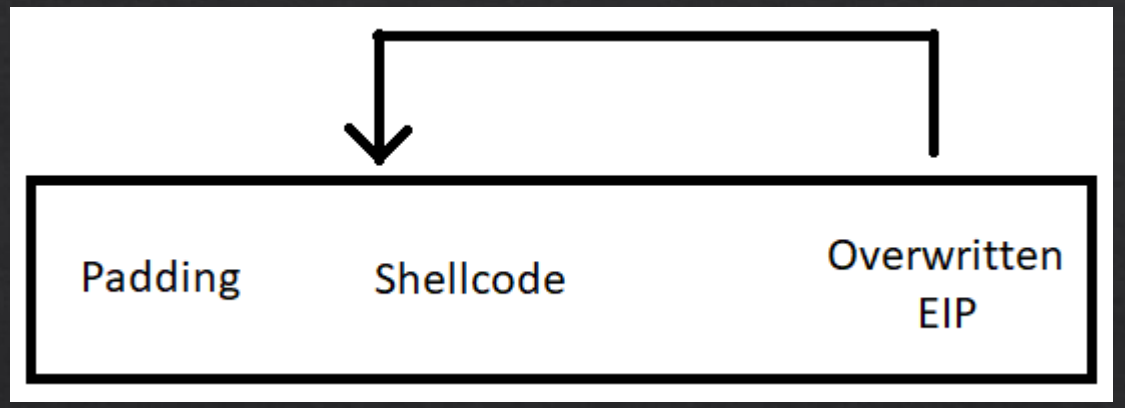
```
/** str
\x6a\x42\x58\xfe\xc4\x48\x99\x52\x48\xbf
\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x54
\x5e\x49\x89\xd0\x49\x89\xd2\x0f\x05
**/
```

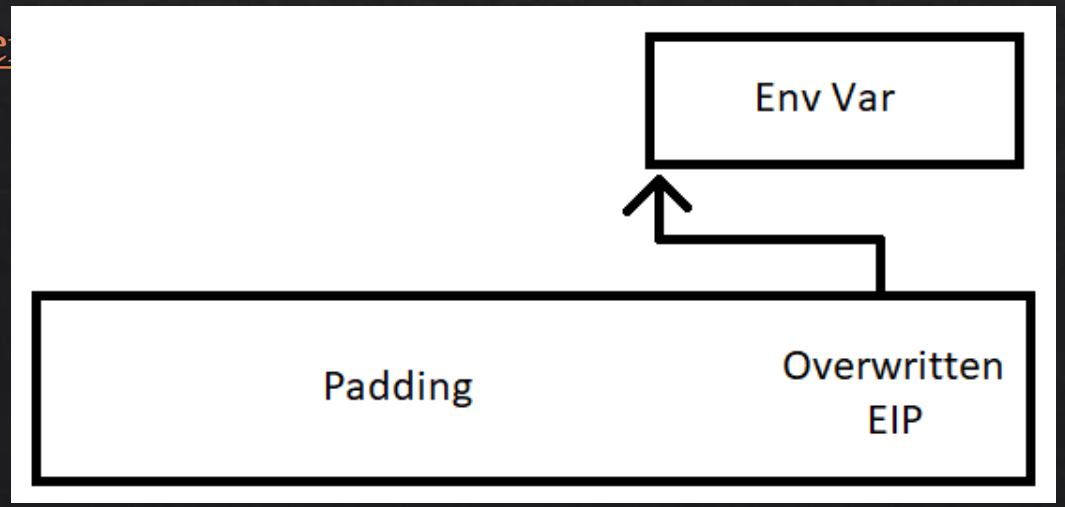
Shellcode - What Can it do?

- Pop a shell
- Spawn a reverse shell
- Elevate privileges
- Add a user
- Well….anything really
- http://shell-storm.org/shellcode/
- Msfvenom

Shellcode - Where Can We Put It

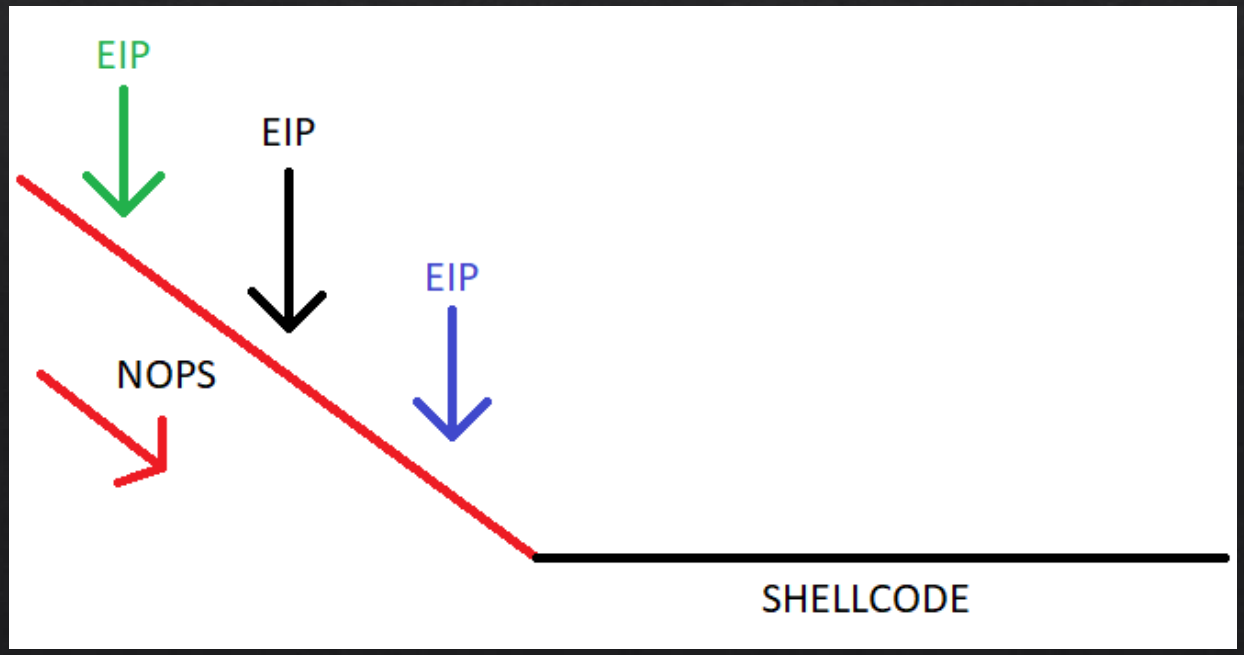
- There are 2 common places to store shellcode
- In the buffer, before the EIP overwrite
- Where we point EIP back into the buffer
- (Sometimes shellcode goes after EIP)
- Or in an environments variable
- It is possible to locate where an environment variable will be in memory!
- https://github.com/Partyschaum/haxe/blob/maste
- Then point EIP at the environment variable





Shellcode - The NOP Sled

- It can be hard to get exact memory locations
- GDB environment != real
- How can we account for this
- Enter the NOP
- Just moves onto the next instruction
- So use the dif between buf and shellcode len
- Point EIP in the middle
- And hopefully you hit the sled!



An Aside - Pitfalls

- Causing argv[0] to change
- Offsets due to GDB
- Python3 text encoding (REEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE)
- Setuid code that doesn't set its uid
- Forgetting your endianness
- Trying to use null bytes in string exploits (remember null termination?)

Example Time

https://pastebin.com/EB4EBhWU

Example Time - Our Binary

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <stdlib.h>
4  #include <unistd.h>
5
6
7  void overflow(char *source) {
8
9      char buf[128];
10
11     strcpy(buf, source);
12
13     printf("Buf: %s\n", buf);
14
15 }
16
17
18 int main(int argc, char **argv) {
19
20     setuid(0);
21
22     printf("Arg: %s\n", argv[1]);
23
24     overflow(argv[1]);
25
26     return 0;
27 }
```

Example Time - Compilation and Execution

- Need to turn off some modern protections
- ASLR needs to be off
- **echo 0 > /proc/sys/kernel/randomize_va_space**
- And compile with
- **gcc vuln.c -o vuln -m32 -fno-stack-protector -z execstack**
- Give it the setuid bit so it can run as root (the setuid function in C earlier)
- **chmod 4777 ./vuln**

Example Time - Demo

- Now lets use this to go from user "kek" to root!
- Demo time
- Shellcode used: http://shell-storm.org/shellcode/files/shellcode-811.php
- And there we go, a simple stack overflow

Next Week

Next Week

- Practical session!
- You'll need a laptop (can work in pairs)
- AND A WORKING VIRTUAL BOX (We'll provide the VM image)
- If you are unsure of virtual box, please ask today so we can help before next week
- Macs, I'm not sure how to fix them, one day I'll actually save the link
- There will be a provided VM with various challenges, each will let you access the next

Some Extras

DC44114

- A local defcon group
- Next event on the 28th
- Down in the city 6.30 - 8
- Register at https://tinyurl.com/y62lf8zr
- Its free!
- Let us know if you're going, we can try to meetup