

An Introduction to Shellcode: Part Two

Playing By the Rules

Robert Peaslee

Quick Recap

- Program execution
- Memory management
- "Stack of Plates"
- Stack-based overflows
- Different types of shellcode

What can we do?

Where We Are Going

- x86 Memory
 - Stack
 - Stack Frame
- Vulnerable Code
- Anatomy of an Exploit
- Shellcode
- Gaining root

We have a lot to do!

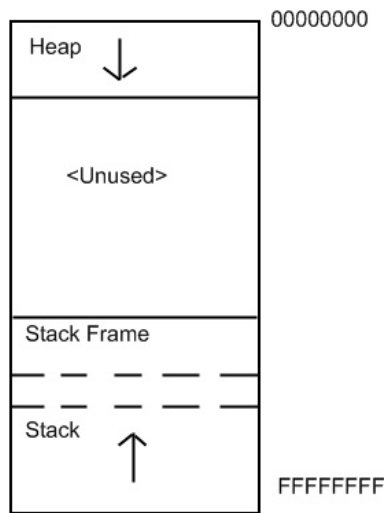
x86 Memory – What is up?

All architectures differ
x86 most common

Heap grows "up"

Stack grows "down"

I always order images
with this orientation



Stack Frame

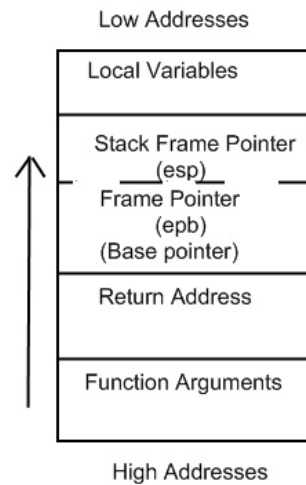
Layout is important

"Stack pointer" -> ESP

"Base pointer" -> EBP

Return address -> RET

Order of allocations
allows overflow



Analysis of Frame Elements

From bottom up (from high to low addresses):

- Function args
 - Stores parameters
- Return address
 - Stores instruction pointer (EIP) of caller
- Base Pointer
 - Stores base offset of program
- Stack Pointer
 - Stores base of frame
- Local Variables
 - Variables local to function

Vulnerable Code

Many forms of vulnerable code:
format string, stack and heap overflows...

Simple example, stack overflow:

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[]) {
    char buffer[128];
    strcpy(buffer, argv[1]);
    return 0;
}
```

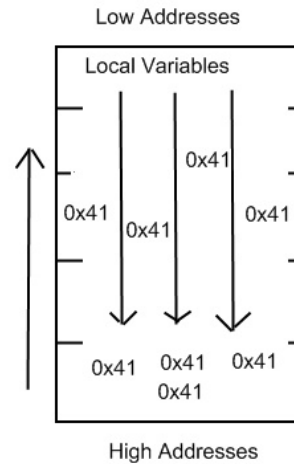
Stack Based Overflows

Too much in too little

Overwrite:

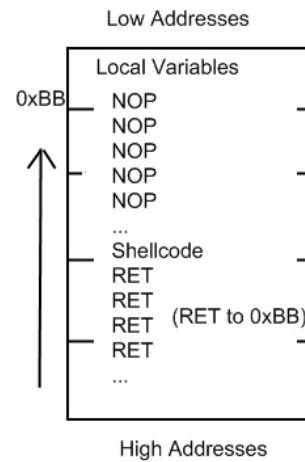
- Other local variables
- ESP
- EBP
- RET...

What happens during
normal execution?
Why?



Anatomy of an Exploit

- Create buffer
 - How big?
- Create NOP sled
 - How many?
- Insert shellcode into buffer
- Create RET Sled™
 - How many?
- Inject shellcode



Avoiding Graphics

Just as I hate plates,
I hate graphics.

Exploitation, live!

Querying the Hive-mind

Audience Participation!

What do we do if...

We cannot execute on the stack?

The buffer is too small to store the shellcode?

An intrusion detection system is looking for NOP sleds?

There are filters on what data can be input?

Practice Being a Hacker

Hackers are inquisitive by nature.

Practice being a hacker!

Special thanks: David Brenner

Questions, comments, requests?

peasleer@gmail.com