Writing Exploits

Nethemba s.r.o.

norbert.szetei@nethemba.com

Motivation

- Basic code injection
- W^X (DEP), ASLR, Canary (Armoring)
- Return Oriented Programming (ROP)
- Tools of the TradeTools
- **Metasploit**

A Brief History

- *08/11/1996 Phrack #49*
- *Smashing The Stack For Fun And Profit, Elias Levy*

“ … Code that does this is said to smash the stack, and can cause return from the routine to jump to a random address. This can produce some of the most insidious data-dependent bugs known to mankind.”

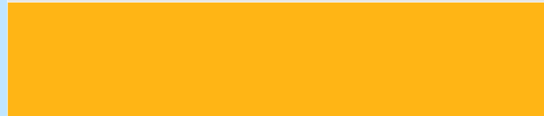
Stack Frame

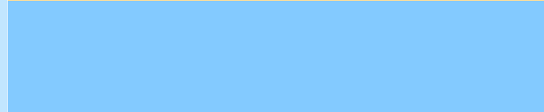
```
void func(int a, int b, int c) {
   char buffer1[BUFSIZE];
   char buffer2[BUFSIZE];
}

int main(int argc, char **argv) {
  func(10, 20, 30);
}
```

- prologue, epilogue

```
push ebp              mov esp, ebp
mov ebp, esp          pop ebp
sub esp, $const       ret
```





Buffer Overflow

```
void func(char *src) {
   char dest[64];
   strcpy(dest, src);
}

int main(int argc, char **argv) {
  func(argv[1]);
}
```

Low Memory Address
dest
EBP
EIP
args
High Memory Address

Buffer Overflow

```
void func(char *src) {
   char dest[64];
   strcpy(dest, src);
}

int main(int argc, char **argv) {
  func(argv[1]);
}
```

Low Memory Address
SHELLCODE
JUNK
JMP TO SC
JUNK
High Memory Address

Buffer Overflow – SEH

```
try {
  int a = 5;
  int b = 0;
  int c = a / b;
} catch (Exception e) {
  printf("ignore ..");
}
```

Next →	Next →	Next →	0xFFFFFFFF
Pointer to Exception Handler	Pointer to Exception Handler	Pointer to Exception Handler	Default Exception Handler

Buffer Overflow – SEH

```
try {
  int a = 5;
  int b = 0;
  int c = a / b;
} catch (Exception e) {
  printf("ignore ..");
}
```

Shellcode address	Next →	Next →	0xFFFFFFFF
POP POP RET	Pointer to Exception Handler	Pointer to Exception Handler	Default Exception Handler

Stack cookies – canaries

- Protection provided by the compiler (/gs, -fstack-protector, StackGuard, ProPolice)
- Can rearrange the stack layout, so string variables are on higher addresses and cannot overwrite other local variables
- Contain “bad” characters (0x00, 0xFF)

Stack cookies – canaries

- Usually a challenge
- Entropy weaknesses (24-bit entropy on Ubuntu, can by bypassed in reasonable time)
- Sometimes helps to overwrite SEH
- Cannot protect from buffer overflows in heap

Protection - DEP

- Stack is no longer executable
- W^X
- Both HW (NX bit) and software support
- Prevent basic buffer overflows
- Four policy levels on Windows platform: Optin, OptOut, AlwaysOn, AlwaysOff
- Can be bypassed by “return-to-libc”

Return to LIBC

- The most generic method to bypass NX
- No executable code in stack
- EIP is overwritten by library function (system())
- Parameters are passed via stack
- Chained “return to libc”
- No loops, conditional jumps, complicated things
- *28/12/2001 Phrack #58, Advanced return-into-lib(c) exploits*

Return to LIBC

←basic buffer overflow

return to libc →

ASLR

- Address Stack Layout Randomization
- Including Libraries, Heap, Stack
- But not necessary in all libraries
- You need at least one module without ASLR for bypassing in Windows
- Implementation weaknesses
- Can by bypassed by format string exploits

Format String Attacks

```
int main(int argc, char **argv) {
  printf("%s", argv[1]); // correct
  printf(argv[1]); // wrong
}
```

- Reading, writing from arbitrary memory
- Direct parameter access via %<num>$
- Writing via %n, %hn (2 bytes)
- *28/07/2002 Phrack #59, Advances in format string exploitation*

Return Oriented Programming

- The successor of “return to libc” technique
- Small number of instructions ending with “ret” (Gadgets) chained together
- **If we find them enough, we have the Turing Machine**
- Fixed Memory location for data interchange, usually in .data section
- 2 registers are usually efficient

Return Oriented Programming

- You can bypass character restrictions **(neg)**
- No injected code, just rewritten stack
- ESP determines which instructions you execute
- Automated by tools (ropeme, ROPGadget)

```
# execve /bin/sh bindport 8080 generated by RopGadget v3.3
p += pack("<I", 0x08050dda) # pop %edx | ret
p += pack("<I", 0x080cd6a0) # @ .data
p += pack("<I", 0x080a49f6) # pop %eax | ret
p += "//us"
p += pack("<I", 0x080796ed) # mov %eax,(%edx) | ret
...
```

Return Oriented Programming

- We can build the custom stack at fixed location (bypass ASLR)
- .data, .bss (readelf)
- Multi-stage exploit
- GOT entry overwriting

```
offset = execve() - printf()
execve() = printf() + offsef
```

- Countermeasure: Position Independent Executable (PIE)

Metasploit

- msfpescan, msfelfscan, msfmachscan
- irb, framework for exploits development
- tools/ (memdump, metasm_shell, pattern_create.rb, pattern_offset.rb, nasm_shell)
- mixins

Immunity Debugger

- 'mona' (successor of pvefindaddr)
- skeleton for metasploit exploit can by generated with Immunity Debugger (mona plugin)

Radare

- Reverse engineering framework, *nix-style, multiplatform
- *11/06/2009 Phrack #66, Manual Binary Mangling With Radare*

```
radare: the entrypoint for everything :)
rahash: block based hashing utility
radiff: multiple binary diffing algorithms
rabin:  extract information from binaries
rasc:   shellcode construction helper
rasm:   commandline assembler/disassembler
rax:    inline multiple base converter
xrefs:  blind search for relative code references
```

Wargames

- http://overthewire.org
- http://smashthestack.org

References

- http://www.radare.org/get/radare.pdf
- https://www.metasploit.com

Any questions?

Thank you for listening

Norbert Szetei, CEH