Google

Proprietary

Binary Exploitation

An Introduction to

Carl Svensson/2021-01-28

Proprietary

Biography

- MSc in Computer Science, KTH
- Security Engineer @ Google - Offensive Security
- CTF Player: HackingForSoju
- Email: zetatwo@google.com / Twitter: @zetatwo

Proprietary

Agenda

- Background
- Stack-based Exploitation
- Protections and bypasses
- Heap-based exploitations
- Next steps

Proprietary

Background

Proprietary

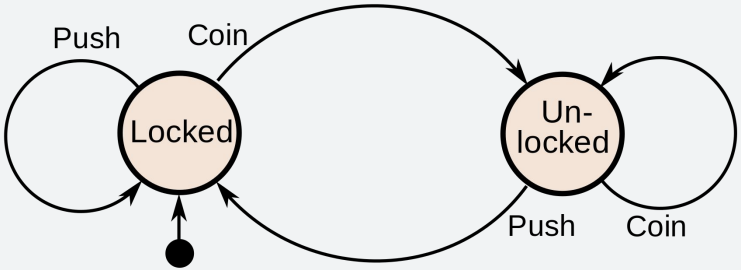
Who Are You?

- Programmer
- Security Interested
- Basic knowledge of some low-level language, e.g. C or C++
- Basic understanding of operating systems

Proprietary

What is an Exploit?

- Unintended behaviour
- State machine
  - Initial state
  - Reachable state
  - Invalid state
- Vulnerability
  - Unintended transition (bug)
  - Enabling an exploit
- Exploit
  - Transition to an Invalid state
  - "Dangerous" subset



Google

A Note on Data

Proprietary

- We organize bits into groups - nibble, byte, word, dword, qword
- Bits are **interpreted** as integers, text, code, addresses, etc.
- Same data, different interpretations - **Context** determines
- Remember endianness - Little vs big

65, 66, 67, 68
"ABCD"
inc ecx; inc edx; inc ebx; inc esp
0x44434241 = 1145258561

Little: 0x44332211 = 0x11 0x22 0x33 0x44
Big: 0x44332211 = 0x44 0x33 0x22 0x11

Proprietary

Where are We?

- Physics - Maxwell’s equations
- Circuits - Gates, flip-flops, wires
- Micro-architecture - Internals of CPU
- **Machine code - Assembly translated to bytes**
- Low-level code - C, Rust
- Mid-level code - Java, C#
- High-level code - Python, Javascript

Google

Proprietary

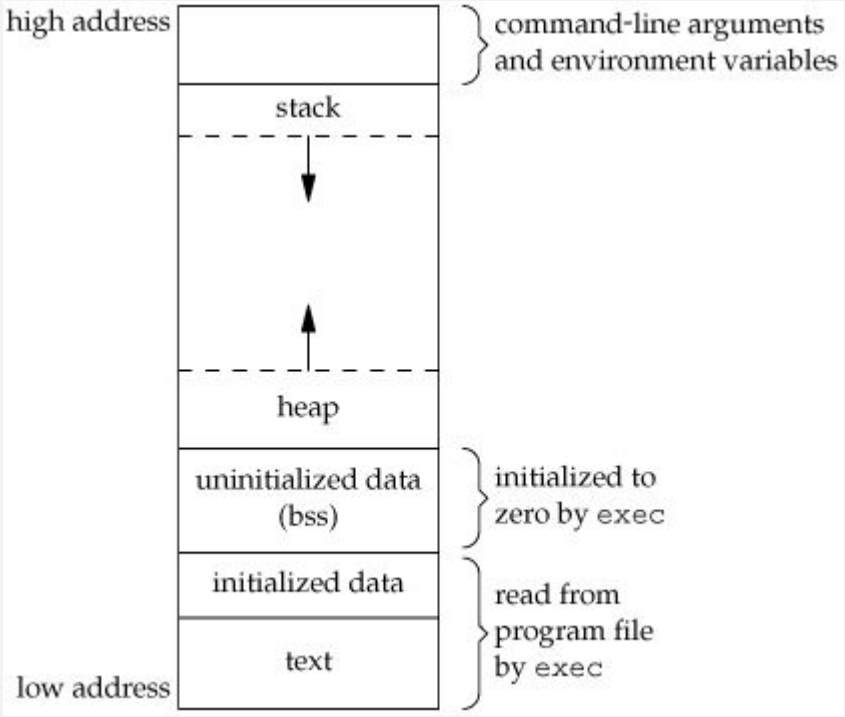
x86 Basics

Google

Proprietary

x86 Memory

- Virtual memory
- Stack
- Heap
- Code - Text

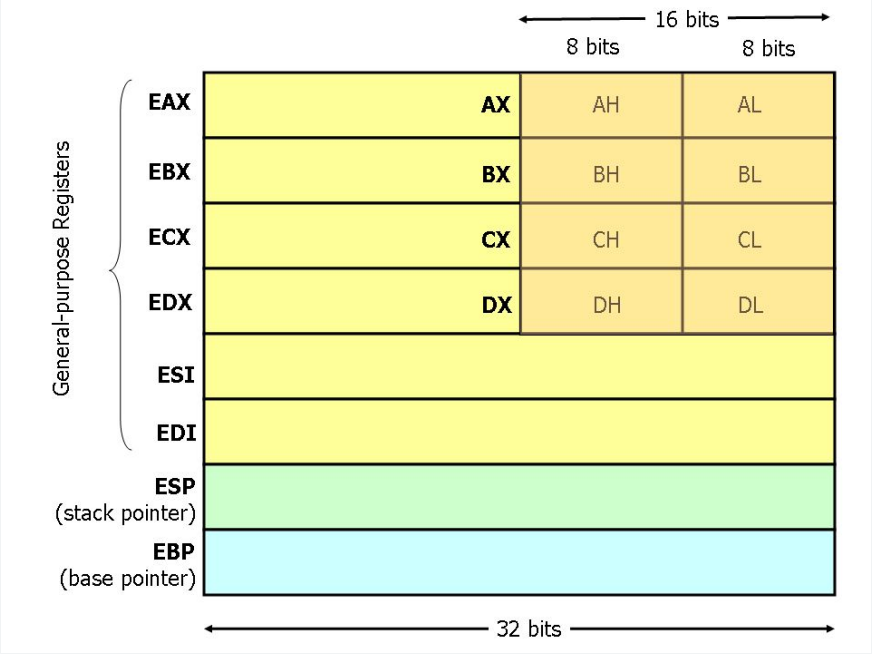


Google

Proprietary

x86 Registers

- General purpose
  - RAX, RBX, RCX, RDX
  - RDI, RSI, R8, R9
- Special purpose
  - RIP, RBP, RSP
- ...and a few hundred more



Google

Proprietary

x86 Calling convention

- Architecture specific
- x86, 32 bit, 64 bit
- Arguments
  - 32 bit: stack in reverse order
  - 64 bit: first few in registers
- Stack frame - base pointer

call **0xCAFEC0DE** ...	push eip+5 jmp **0xCAFEC0DE**
ret	pop eip
call **rip+0x1337** ...	push rip+5 jmp **rip+0x1337**
ret	pop rip
f(a, b)	push b; push a call f
f(a, b)	mov rdi a; mov rsi b; call f

Google

Proprietary

Stack-based Exploits

Proprietary

Stack buffer Overflow

- Unchecked write
- Overwrite adjacent memory
- Overwrite return address

```
void vuln() {
  long local1;
  char buf[16];
  fgets(buf);
}
```

[buf (16 bytes)]
[local1 (8 bytes)]
[saved bp (8 bytes)]
[return address (8 bytes)]

[AAAABBBBCCCCDDDD]
[EEEEFFFF]
[GGGGHHHH]
[**JJJJKKKK**]

Program received signal SIGSEGV, Segmentation fault.
0x4B4B4B4B4A4A4A4A in example1 ()

Google

Proprietary

Shellcode

- Code that launches a shell
- Can also do other things
- Mostly written in C or ASM
- Needs to be location independent

```
xor     rdx, rdx
mov     qword rbx, '//bin/sh'
shr     rbx, 0x8
push    rbx
mov     rdi, rsp
push    rax
push    rdi
mov     rsi, rsp
mov     al, 0x3b
syscall
```

```
0x48 0x31 0xd2 0x48 0xbb 0x2f 0x2f 0x62 0x69 0x6e 0x2f
0x73 0x68 0x48 0xc1 0xeb 0x08 0x53 0x48 0x89 0xe7
0x50 0x57 0x48 0x89 0xe6 0xb0 0x3b 0x0f 0x05
```

Google

Proprietary

Stack buffer overflow -96

- No protections present
- No longer viable
- A simple attack
  - Inject code
  - Overwrite return address with shellcode location

```
void vuln() {
  long local1;
  char buf[16];
  fgets(buf);
}
```

[buf (16 bytes)]
[local1 (8 bytes)]
[saved bp (8 bytes)]
[return address (8 bytes)]

0x00007FFFDEADC0DE:
[0x48 0x31 0xd2 0x48 ...]
[...]
[... 0x3b 0x0f 0x05]
[**0x00007FFFDEADC0DE**]

```
$ uname -a
Linux pwnbox 5.4.0-12.15-generic...
```

Proprietary

Shellcode placement

- Shellcode can be moved around
- For example further down the stack
- If exact location is unknown
  - NOP sled

```
void vuln() {
  long local1;
  char buf[12];
  fgets(buf);
}
```

[buf (12 bytes)]
[local1 (8 bytes)]
[saved bp (8 bytes)]
[return address (8 bytes)]
[prev frame (? bytes)]

0x00007FFFDEADC0DE:
[...] [...]
[...]
[0x00007FFFDEADC102]
[0x48 0x31 0xd2 ...]

```
$ uname -a
Linux pwnbox 5.4.0-12.15-generic...
```

Proprietary

Protection: ASLR -01

- Address Space Layout Randomization
- Randomize location of stack and heap
  - 32 bit: 12 bit entropy
  - 64 bit: 28 bit entropy
- So far code location still known
- Location of buffer now unknown
- Code reuse
  - Gadgets

0x00007FFFCAFECAFE:
jmp rsp

[buf (16 bytes)]
[local1 (8 bytes)]
[saved bp (8 bytes)]
[return address (8 bytes)]
[prev frame (? bytes)]

0x????????:
[...] [...]
[...]
[0x00007FFFCAFECAFE]
[0x48 0x31 0xd2 ...]

$ uname -a
Linux pwnbox 5.4.0-12.15-generic...

Google

Proprietary

Protection: NX/DEP -97

- Adds permission bits to memory
  - Code: RX
  - Heap+Stack: RW
- Shellcode on stack not possible
- Code location know
- Gadgets
  - Return-oriented programming

```
0x4000104A:
...
pop eax
ret
```

```
0x4000106A:
...
pop ebx
pop ecx
ret
```

```
0x????????:
[AAAA...DDDD]
[EEEE]
[FFFF]
[0x4000104A]
[0xDEADBEEF]
[0x4000106A]
[0xCAFEBABE]
[0xFEEDF00D]
```

```
eax = 0xDEADBEEF
ebx = 0xCAFEBABE
ecx = 0xFEEDF00D
```

Google

Proprietary

Protection: StackGuard -98

- Catch the overflow before damage
- Canary - random secret value
- The crash becomes controlled
- Relies on canary being secret
  - Memory leak
  - Forking servers

```
void vuln() {
  long local1;
  char buf[12];
  fgets(buf);
}
```

```
void vuln() {
  push_cookie();
  long local1;
  char buf[12];
  fgets(buf);
  check_cookie();
}
```

```
SECRET = ???
[...] [...]
[SECRET]
[saved bp (8 bytes)]
[return address (8 bytes)]
```

```
[...] [...]
[0x4141414141414141]
[0x4141414141414141]
[0x00007FFFDEADC0DE]
```

```
*** stack smashing detected ***: ./a.out terminated
======= Backtrace: =========
```

Google

Proprietary

GOT/PLT Overwrite

- Program Linkage Table, PLT
- Global Offset Table, GOT
- PLT contains stubs with jumps
- GOT contains addresses to libraries
- Overwrite GOT entry and call function

```
...
call printf@plt
...
```

```
printf@plt:
  jmp [printf@got]
```

```
printf@got: 0x7FFFC0DECAFE
```

```
printf@got: 0x7FFFDEADDEAD


...
call printf@plt -> 0x7FFFDEADDEAD
...
```

Google

Proprietary

Protection: RELRO

- RELocation Read Only, RELRO
- “Partial RELRO”
  - GOT before BSS
- Full RELRO
  - Actually Read Only
  - Handled by loader

```
...
call printf@plt
...
```

```
printf@plt:
  jmp [printf@got]
```

```
printf@got: 0x7FFFC0DECAFE
```

~~printf@got: **0x7FFFDEADDEAD**~~

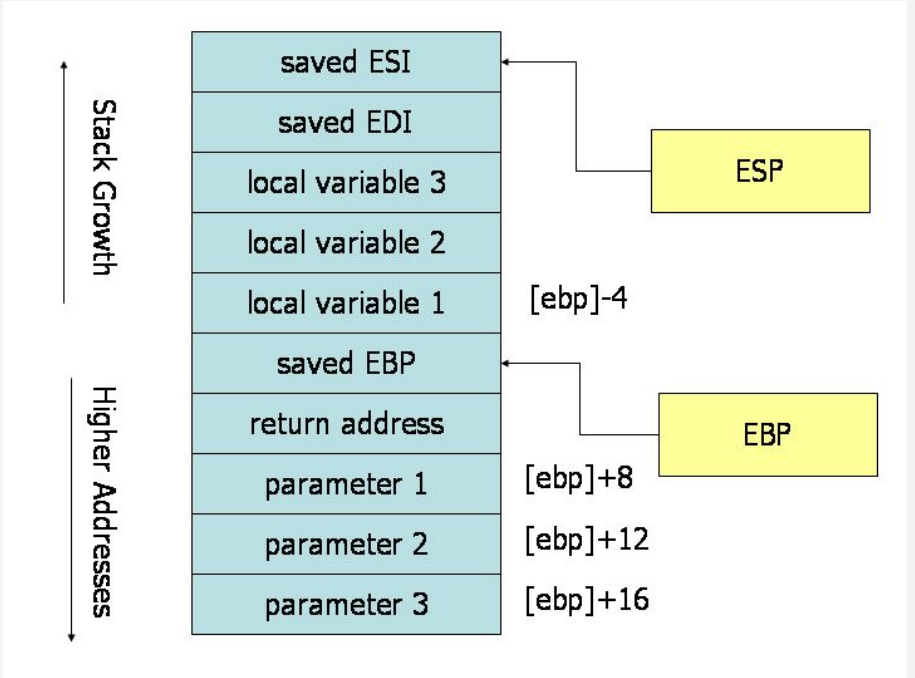
```
...
call printf@plt -> 0x7FFFC0DECAFE
...
```

Google

Proprietary

Base Pointer Overwrite

- Stack frames - linked list
- Misalign stack frame
  - Modify local variables
  - Modify stack pointer
- Partial overwrite
  - Shift stack frame



Google

Proprietary

Protection: CFG (-14)

- Control Flow Guard
- Control Flow Integrity
- Intended to prevent code-reuse attacks
- Bypass example: JIT

Proprietary

Protection: PAC (-17)

- Pointer Authentication Code
- Reuse unused bits for MAC
- Hardware support
- ARM64, Apple iOS
- Bypass: signing oracle
    - Project Zero blog

Google

Proprietary

Format String Vulnerability

- Calls to printf-like functions
- Control over first argument
- Variable number of arguments
- Read direct: %x/%d
- Read indirect: %s
- Write: %n
- Copy: %0*x, %n
- Skip: %4$x

```
int printf ( const char * format, ... );

printf("Name: %s, age: %d", name, age); // Ok
printf(name); // Vulnerable
```

Google

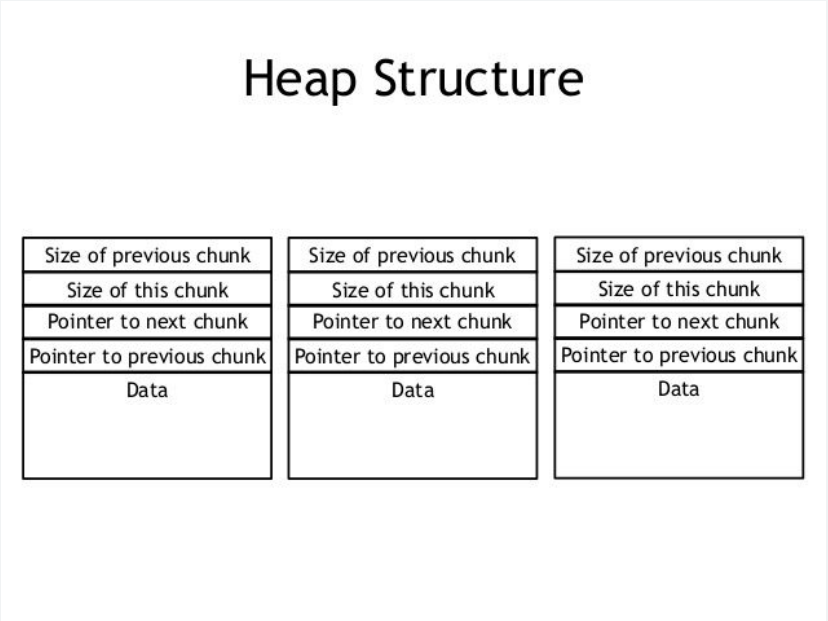
Proprietary

Heap-based Exploits

Proprietary

A Refresher on Memory

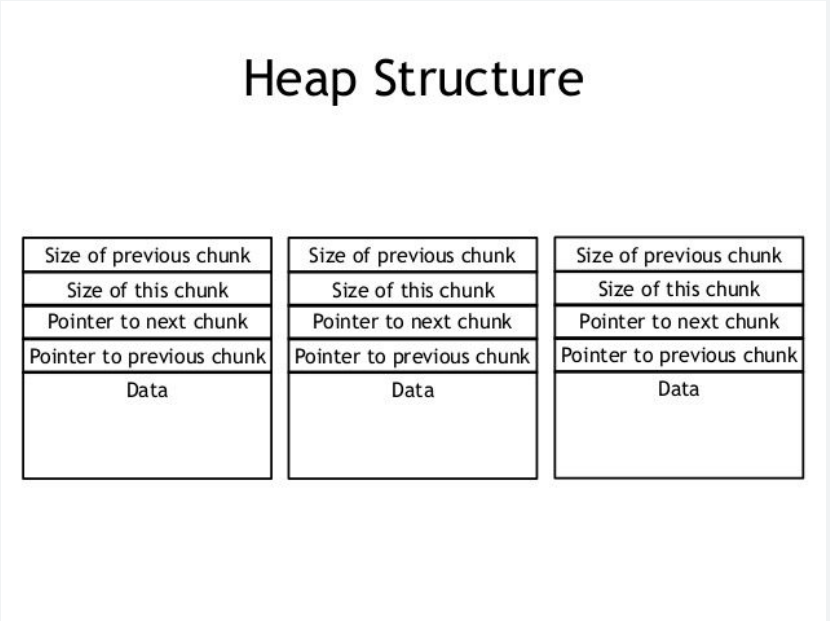
- Physical
- Virtual
- Pages
- Memory allocator
  - malloc/free
  - glibc
  - jemalloc



Proprietary

Heap corruption: app layer

- Heap overflow
- Use after free
- Type confusion
- Heap spraying

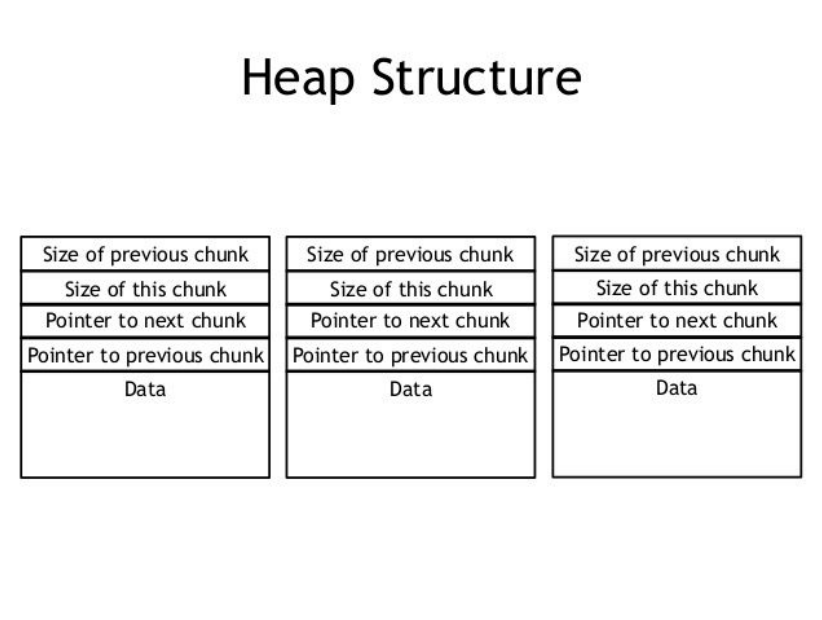


Google

Proprietary

Heap corruption: allocator

- Corrupt allocator metadata
- Linked lists
- Requires understanding of allocator
  - Slabs
  - Bins
  - Cache
- glibc - House of X



Google

Proprietary

Next Steps

Proprietary

Want to try it out?

Capture the Flag

https://capturetheflag.withgoogle.com
https://ctftime.org

Wargames

https://picoctf.com
https://github.com/zardus/wargame-nexus
https://pwnable.kr
https://overthewire.org

Community

CTF players Discord:
https://discord.gg/ArjWjvctft

Further Materials

Proprietary

Videos

https://securitycreators.video
https://www.youtube.com/GynvaelEN
https://www.youtube.com/ZetaTwo
https://www.youtube.com/LiveOverflow

Tools

Python + Pwntools
gdb + gef
IDA, Binary Ninja, Ghidra

Learning

https://pwn.college
https://github.com/RPISEC/MBE
https://github.com/shellphish/how2heap

Google

Interested in Google?

Internships and full-time positions:
https://careers.google.com/students

Questions about working at Google, specifically security:
Email zetatwo@google.com or
Twitter @zetatwo



Thank You