Miscellaneous Stuff

Stuart Nevans Locke

Overview

- Background on Integers
  - Size
  - Signed vs Unsigned
- Overflow and Underflow
- Integer Size Mismatch
- Uninitialized Memory
- Out of bounds Accesses
- Tools
  - One_gadget
- Fastcall

Background on Integers

- Signed and Unsigned Integers
  - Unsigned have an extra bit to use and can store numbers up to twice the magnitude
- Normal Sizes
  - char - 8 bit
  - int - 32 bit
  - long long - 64 bit

Integer Overflow

```
unsigned char x = 255;

x+=1;
```

- X will now have a value of 0
- This is because 255 is **11111111** in binary

Integer Underflow

unsigned char x = 0;

x-=1;

- X now has a value of 255
- 0 is **00000000** in binary

Integer Size Mismatch

```
void boundsCheck(int x){
    return x <100;
}
size_t var = 1LL<<33+1
boundsCheck(var) // returns true
memcpy(dest, src, var) //going to be an overflow
```

- Compilers can give warnings about these (-Wconversion flag for gcc)
  - Nonetheless, these occur in real software
  - https://googleprojectzero.blogspot.com/2015/07/when-int-is-new-short.html

Uninitialized Variables

- Stack Variables
  - int uninitialized;
  - printf(“%i\n”,uninitialized);
- Heap Variables
  - int * x = malloc(32);
  - printf(“%i\n”,*x);
- We can sometimes control the value of these variables
  - If they are used before being given values, we probably break things
- Compiler also typically warns about these

Out of Bounds Accesses

- Sometimes used as a way to exploit overflow

Example:

```
int offset;
scanf(“%d”,&offset);

printf(“%llx”,array[offset]);
```

- This would give us an arbitrary leak
- If we can set using an OOB, we have arbitrary write

Tool - one_gadget

- Recall back to ROP
  - Having a function that called ***system(“/bin/sh”)*** was great
  - But we said that doesn’t happen in the real world
- A Magic Gadget is a piece of code that does exactly that
  - Found inside of libc
- No more full ROP chains
  - Just call the magic gadget

Tool - one_gadget

- There are multiple magic gadgets in libc
- one_gadget prints them and their prerequisites
- usage: one_gadget /path/to/libc

Note: Don’t become too reliant on this, it acts as a crutch. If /bin/sh doesn’t exist, this won’t work

Fastcall

- Most popular calling convention used on 64 bit
- Arguments passed in registers
  - 1st - rdi
  - 2nd - rsi
  - 3rd - rdx
  - 4th - r10
- Return value put in rax

Questions

What do you want to cover?

- Heap Stuff (There’s a lot of this)
- Fuzzers
  - Theoretical
  - Practical
- Cool Variants of ROP
  - Blind Return Oriented Programming
  - Signal Return Oriented Programming (Not much substance to this one)
- Browser Stuff
- Kernel Stuff

Print Format Vulnerability

- These could be their own presentation
  - Might be, if people want to do more with them
- Calling printf(userInput) is really bad
  - If the user passes “%d”, printf reads an integer from the stack
  - More generally, any format specifiers are read from the stack
- Leaks
  - %llx will leak a pointer off the stack in 64 bit
- %n
  - Format specifier seemingly made to allow print formats to be dangerous
  - Writes the amount of bytes written so far to the variable referred to by some address
  - printf(“Hi%n”,&int)
    - int would contain two