flaw - fault or failure

failure when something actually goes wrong

- try and cause failures to find faults
- issue patches to fix faults

Bug vs Fault
- bug just mistake
- fault is a security concern

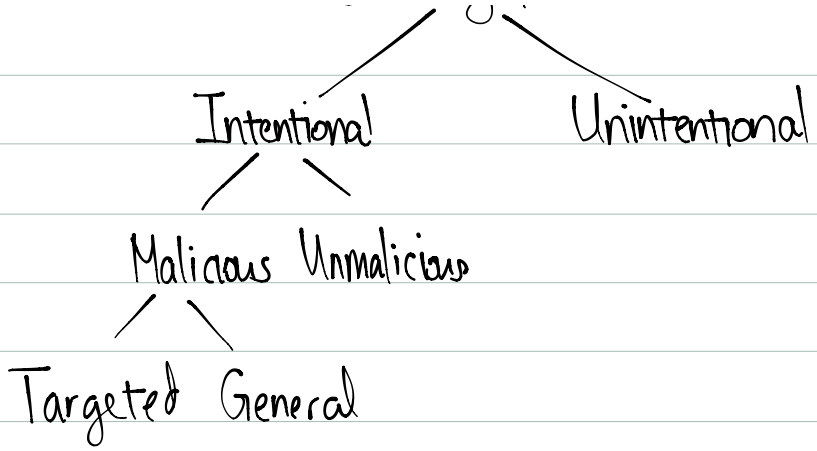
Patching sometimes makes things worse
- narrow focus - could lose vision of larger underlying cause

Unexpected Behaviour
- specified and additional behaviour
- what should and shouldn't it do

Testing -
- try and make program misbehave
- fuzz test
  ↳ random input

Types of Security Flaws

Security Flaws



Heartbleed Bug - TLS

- missing a bounds check
- reading buffer overflow

p - string sent to the sender
payload = integer (# of bytes requested)
padding = 16
buffer has length = payload + 19

fix - a bounds check

Apple SSL/TLS Bug (Feb 2014)

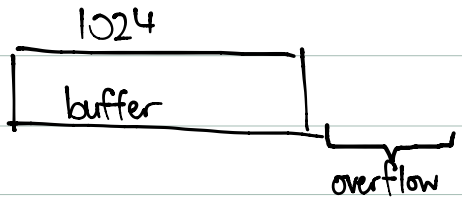
- OS X & iOS affected
- counterfeit key - man-in-the-middle attack

flaw -
Three verifications
- if a check fails the flag "err" set to a non-zero num

- used to indicate error
- two consecutive goto fails
- if two checks pass, we bypass the third check

Buffer Overflow

- most commonly exploited flaw
-



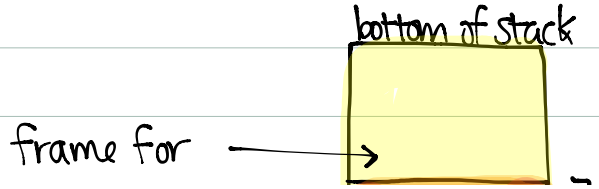
Smashing the Stack for Fun and for Profit

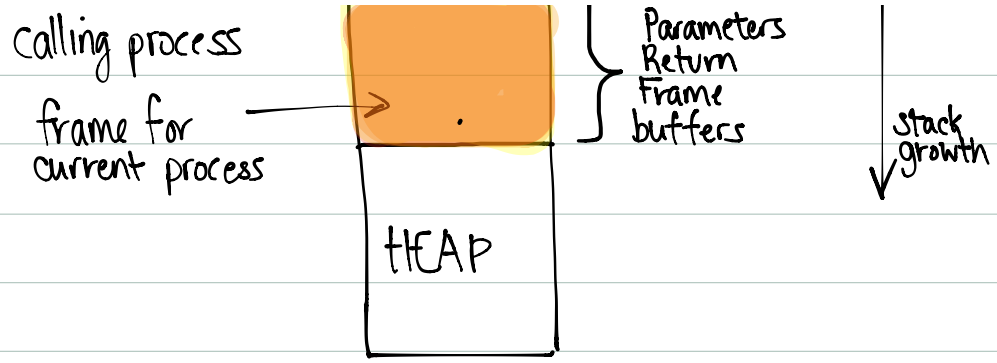
1. Place shell code into buffer
2. Overwrite return address to point to shell code

Attacker needs knowledge about the call stack
↳ usually done on programs with elevated privilledges

Modern Compilers and Operating Systems prevent the "classic" attack

Typical Structure of Call/Execution Stack





Defenses

- randomized virtual memory addresses
- a canary - block of data to see if modified

Integer Overflow

-

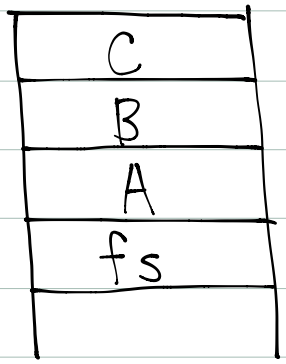
Format String Vulnerability

- "%n" - write to an address found on the stack
- most dangerous
- writes # of bytes that have been printed out so far
- could overwrite return address and then execute shell code
- printf("%x%x%x%x") dumps parts of the stack

If the address is invalid, then program crashes
%s - reads an address from stack
%x - prints a value from the stack

printf is compiled from right to left

call stack printf("fs", A, B, C)



fs = %s %s %s
A B C

If no arguments are given, then we take addresses immediately on top of fs

This could mean that the program could crash.

Incomplete Mediation
- application need to do mediation and make sure the data given is a meaningful request
- focus on catching certain types of errors
  ↳ reasonable values
  ↳ well-formed
  ↳ Inconsistent with other entries

Why do we care?
- buffer overflow
- SQL Injection

Client-side Mediation
- client-side - check
- state should be maintained on server side

Example:
- hidden fields with price

Server-side Mediation is the only secure means
-

Toctou errors
- Time-of-check to time-of-use
- aka "race conditions"

Occur when:
1. User requests an action
2. System verifies authorization
3. System performs an action

Delay between 2 & 3
↳ attacker can change state in-between.

Example:
- program needs sudo
- check if user has permisson to fileA → yes
- user changes logfile to fileB
- write to fileB instead of A

Defences to TOCTOU errors

- make sure the access control decision is constant
  ↳ keep a copy of request so it can't be altered

filehandle (file descriptor)
- process table is an array residing the kernel, contains details about all open files
- file handle is an integer index into this table
- cannot be changed by the user