Software Security

CMSC 433
Bill Pugh

Software Security

• Making sure that if your software is misused, it doesn’t do any of the vast number of things you didn’t intend for the software to do
On trusting trust

- You can hide a trojan horse in a compiler
  - or in the operating system

Compiler

- Code generateCode(AST method) {
  if (method.getName())
  .equals("authenticateLogin)) {
  return ... code with trap door ...;
  .. generate code normally
Slightly cool, but not very interesting

- Get spotted in a code audit

Compiler

- Code generateCode(AST method) {
  if (method.getName().
equals("authenticateLogin")) {
    return .. code with trap door.. }
  if (method.getName().
equals("generateCode")) {
    return ... code with special code gen ...;
    .. generate code normally
  }
}
Trusted code base

- Trusted code base is the code that, if compromised, causes all of your security to fail
- Typically, includes all your software, your compiler, your operating system, ...

- Feeling comfy?

Software defects

- Traditional approach to correctness
  - define precondition
  - show that if precondition satisfied, output satisfied postcondition

- Didn’t examine what happened if input didn’t satisfy precondition
#1 source of security defects

- Untrusted, unverified and unexpected input leading to a program doing something completed unexpected
  - unexpected by developer
  - intended by attacker

- of all the untrusted input problems, #1 is buffer overruns in C/C++.

Buffer overflows

- In C, arrays are just locations in memory
- if you write past the allocated end of the array, you write into something else
- possibly other variables, return address
- can both rewrite return address and deliver payload
gets() is evil

- Impossible to use gets() correctly

```c
char buf[20];
gets(buf);
```

C String functions

```c
char buf[20];
char * prefix = "http://";
strcpy(buf,prefix);
strncat(buf, path, sizeof(buf));
```
sprintf

• char buf[80];
  sprintf(buf, “%s - %d\n”, path, errno);

safe copy

#define MAX_BUF 256

void doStuff(char * in) {
  short len;
  char buf[MAX_BUF];
  len = strlen(in);
  if (len > MAX_BUF) return;
  strcpy(buf, in);
  .. do stuff with buf ...
}

Some Sins

- Buffer Overflows
- Format String problems
- Integer overflows
- SQL injection
- Command injection
- Failure to handle errors
- Cross-site scripting
- Failing to protect network traffic
- Use of "magic" URLs and hidden forms

More sins

- Improper use of SSL
- Use of weak password-based systems
- Failing to store and protect data
- Information leakage
- Improper file access
- Trusting network address information
- Race conditions
- Unauthenticated key exchange
- Failing to use cryptographically strong random numbers
- Poor usability