Principles of Secure System Design (continued)

Separation of Privilege

- 2 keys to launch missile
- mix nets
- workflow

Open Design

"With many eyes, all bugs are shallow"

Fuzz Revisited

- Fed tools random input and tried to make them crash
  Open source tools failed 13% of the time
  Commercial tools failed 25% of the time

Psychological Acceptability

If you make your security measures too restrictive or too difficult to use people will get around them

Too Restrictive

Eg. Person behind corporate firewall will tether to their phone

Too Difficult

- Why Johnny Can't Encrypt
- Password Managers
  - Password Hash for Firefox
    - Generates site specific password
    - Hashes master password with site url to generate unique password per webpage
    - No feedback
- Human based challenge system
  - You remember some value X
  - Server gives user a challenge
  - User combines X with challenge to authenticate

Preventing Exploitation at Deployment Time

Sandboxing & (Host based) Intrusion Detection Systems

- Native Client
  - Part of Chrome Browser
  - Allows you to download x86 code and run it as a plugin to the browser on the fly
    - But doesn't allow the code to destroy your system

- You have a process and want to prevent it from destroying your system
  - Put process inside containment chamber
  - Either trust process (but it might be buggy)
    - Program comes with specifications which can be loaded into container (which enforces specs)
You can't trust program's developer to provide specifications

- You define your own specs or use default specs

**IDS specs**
- Only way you can do damage is through system calls
- Describe allowed "behaviors"
  - Allowed system calls
- Simple IDS model: spec = set of system calls application is suppose to make (app is treated as black box)

**Next step up: sequences of system calls**
- Strace, prints out every system call makes as it makes them
- How to build (specs) model?
  - 6-grams
    - Chop sequences of system calls into blocks of 6 (sliding window)
      - Even if we take a lot of sequences we may miss one
    - Keep track of last 5 calls and match 6th call to a table
    - Vulnerable to Mimicry Attack
      - Can be bypassed by making dummy system calls to bring the state machine to a state where the system call he wants to make is available

**Models without false positives**

```c
void foo(void)
{
    int fd = open(...);
    if (fd < 0) {
        write(...);
        return;
    }
    read(...);
    write(...);
}
```

```c
bar(...)
{
    getuid();
    foo();
    setuid();
}
```

```c
baz(...)
{
    ioctl(...);
    foo();
    setuid();
}
```

Problem is the impossible code path is allowed in this automata (blue path)
Can use pushdown automata to eliminate this but it slows down the system
  - "Efficient Pushdown Intrusion Detection"