Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Yoshi Kohno, Ada Lerner, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...
Announcements

• If you haven’t gotten access to lab 1, please do so ASAP!
  – Checkpoint due Friday (4/14)!

• Coming up
  – Today: finish software security
  – Wednesday: start cryptography
Beyond Buffer Overflows...
Another Type of Vulnerability

• Consider this code:

```c
int openfile(char *path) {
    struct stat s;
    if (stat(path, &s) < 0)
        return -1;
    if (!S_ISREG(s.st_mode)) {
        error("only allowed to regular files!");
        return -1;
    }
    return open(path, O_RDONLY);
}
```

• **Goal:** Open only regular files (not symlink, etc)
• **What can go wrong?**
TOCTOU (Race Condition)

- TOCTOU == Time of Check to Time of Use:

```c
int openfile(char *path) {
    struct stat s;
    if (stat(path, &s) < 0)
        return -1;
    if (!S_ISREG(s.st_mode)) {
        error("only allowed to regular files!");
        return -1;
    }
    return open(path, O_RDONLY);
}
```

- **Goal**: Open only regular files (not symlink, etc)
- Attacker can change meaning of `path` between `stat` and `open` (and access files he or she shouldn’t)
Another Type of Vulnerability

- Consider this code:

```c
char buf[80];
void vulnerable() {
    int len = read_int_from_network();
    char *p = read_string_from_network();
    if (len > sizeof buf) {
        error("length too large, nice try!");
        return;
    }
    memcpy(buf, p, len);
}
```

```c
void *memcpy(void *dst, const void * src, size_t n);
typedef unsigned int size_t;
```
Integer Overflow and Implicit Cast

• Consider this code:

```c
char buf[80];
void vulnerable() {
    int len = read_int_from_network();
    char *p = read_string_from_network();
    if (len > sizeof buf) {
        error("length too large, nice try!");
        return;
    }
    memcpy(buf, p, len);
}
void *memcpy(void *dst, const void *src, size_t n);
typedef unsigned int size_t;
```

If `len` is negative, may copy huge amounts of input into `buf`. 
Another Example

```c
size_t len = read_int_from_network();
char *buf;
buf = malloc(len+5);
read(fd, buf, len);
```

(from [www-inst.eecs.berkeley.edu—implflaws.pdf](http://www-inst.eecs.berkeley.edu—implflaws.pdf))
Integer Overflow and Implicit Cast

What if \( \text{len} \) is large (e.g., \( \text{len} = 0xFFFFFFFF \))?

Then \( \text{len} + 5 = 4 \) (on many platforms)

Result: Allocate a 4-byte buffer, then read a lot of data into that buffer.

```
size_t len = read_int_from_network();
char *buf;
buf = malloc(len+5);
read(fd, buf, len);
```

(from [www-inst.eecs.berkeley.edu—implflaws.pdf](https://www-inst.eecs.berkeley.edu—implflaws.pdf))
Password Checker

• Functional requirements
  – PwdCheck(RealPwd, CandidatePwd) should:
    • Return TRUE if RealPwd matches CandidatePwd
    • Return FALSE otherwise
  – RealPwd and CandidatePwd are both 8 characters long
• Implementation (like TENEX system)

```
PwdCheck(RealPwd, CandidatePwd) // both 8 chars
  for i = 1 to 8 do
    if (RealPwd[i] != CandidatePwd[i]) then
      return FALSE
  return TRUE
```

• Clearly meets functional description
Attacker Model

- Attacker can guess CandidatePwds through some standard interface
- Naive: Try all $256^8 = 18,446,744,073,709,551,616$ possibilities
- Better: Time how long it takes to reject a CandidatePasswd. Then try all possibilities for first character, then second, then third, ....
  - Total tries: $256 \times 8 = 2048$

```
PwdCheck(RealPwd, CandidatePwd)  // both 8 chars
    for i = 1 to 8 do
        if (RealPwd[i] != CandidatePwd[i]) then
            return FALSE
    return TRUE
```
Timing Attacks

• Assume there are no “typical” bugs in the software
  – No buffer overflow bugs
  – No format string vulnerabilities
  – Good choice of randomness
  – Good design

• The software may still be vulnerable to timing attacks
  – Software exhibits input-dependent timings

• Complex and hard to fully protect against
Other Examples

• Plenty of other examples of timings attacks
  – AES cache misses
    • AES is the “Advanced Encryption Standard”
    • It is used in SSH, SSL, IPsec, PGP, ...
  – RSA exponentiation time
    • RSA is a famous public-key encryption scheme
    • It’s also used in many cryptographic protocols and products
Randomness Issues

• Many applications (especially security ones) require randomness
• Explicit uses:
  – Generate secret cryptographic keys
  – Generate random initialization vectors for encryption
• Other “non-obvious” uses:
  – Generate passwords for new users
  – Shuffle the order of votes (in an electronic voting machine)
  – Shuffle cards (for an online gambling site)
C’s rand() Function

- C has a built-in random function: `rand()`

  ```c
  unsigned long int next = 1;
  /* rand: return pseudo-random integer on 0..32767 */
  int rand(void) {
    next = next * 1103515245 + 12345;
    return (unsigned int)(next/65536) % 32768;
  }
  /* srand: set seed for rand() */
  void srand(unsigned int seed) {
    next = seed;
  }
  ```

- Problem: don’t use `rand()` for security-critical applications!
  - Given a few sample outputs, you can predict subsequent ones
Problems in Practice

• One institution used (something like) rand() to generate passwords for new users
  – Given your password, you could predict the passwords of other users

• Kerberos (1988 - 1996)
  – Random number generator improperly seeded
  – Possible to trivially break into machines that rely upon Kerberos for authentication

• Online gambling websites
  – Random numbers to shuffle cards
  – Real money at stake
  – But what if poor choice of random numbers?
mamajoe: Hey guys, Big B is in!
More details: “How We Learned to Cheat at Online Poker: A Study in Software Security”
PS3 and Randomness

Hackers obtain PS3 private cryptography key due to epic programming fail? (update)


• 2010/2011: Hackers found/released private root key for Sony’s PS3
• Key used to sign software – now can load any software on PS3 and it will execute as “trusted”
• Due to bad random number: same “random” value used to sign all system updates
Other Problems

• Key generation
  – Ubuntu removed the randomness from SSL, creating vulnerable keys for thousands of users/servers
  – Undetected for 2 years (2006-2008)

• Live CDs, diskless clients
  – May boot up in same state every time

• Virtual Machines
  – Save state: Opportunity for attacker to inspect the pseudorandom number generator’s state
  – Restart: May use same “psuedorandom” value more than once
int get_random_number()
{
    return 4; // chosen by fair dice roll.
    // guaranteed to be random.
}

https://xkcd.com/221/
Obtaining Pseudorandom Numbers

• For security applications, want “cryptographically secure pseudorandom numbers”
• Libraries include cryptographically secure pseudorandom number generators
• Linux:
  – /dev/random
  – /dev/urandom - nonblocking, possibly less entropy
• Internally:
  – Entropy pool gathered from multiple sources
Where do (good) random numbers come from?

- **Humans:** keyboard, mouse input
- **Timing:** interrupt firing, arrival of packets on the network interface
- **Physical processes:** unpredictable physical phenomena
Software Security:
So what do we do?
Fuzz Testing

• Generate “random” inputs to program
  – Sometimes conforming to input structures (file formats, etc.)
• See if program crashes
  – If crashes, found a bug
  – Bug may be exploitable
• Surprisingly effective

• Now standard part of development lifecycle
General Principles

• Check inputs
Shellshock

• Check inputs: not just to prevent buffer overflows
• Example: Shellshock (September 2014)
  – Vulnerable servers processed input from web requests
  – Passed (user-provided) environment variables (like user agent, cookies...) to CGI scripts
  – Maliciously crafted environment variables exploited a bug in bash to execute arbitrary code

```bash
env x='() { }; echo OOPS' bash -c :
```
General Principles

• Check inputs
• Check all return values
• Least privilege
• Securely clear memory (passwords, keys, etc.)
• Failsafe defaults
• Defense in depth
  – Also: prevent, detect, respond

• NOT: security through obscurity
General Principles

• Reduce size of trusted computing base (TCB)
• Simplicity, modularity
  – But: Be careful at interface boundaries!
• Minimize attack surface
• Use vetted component
• Security by design
  – But: tension between security and other goals
• Open design? Open source? Closed source?
  – Different perspectives
Does Open Source Help?

• Different perspectives...

• Happy example:
    (http://www.freedom-to-tinker.com/?p=472)

• Sad example:
  – Heartbleed (2014)
    • Vulnerability in OpenSSL that allowed attackers to read arbitrary memory from vulnerable servers (including private keys)
http://xkcd.com/1354/

SERVER, ARE YOU STILL THERE? IF SO, REPLY "POTATO" (6 LETTERS).

User Meg wants these 6 letters: POTATO. User Xanida wants pages about "irl games". Unlocking secure records with master key 5130985733435. User in (chrome user) sends this message: "HI"

User Meg wants these 6 letters: POTATO. User Xanida wants pages about "irl games". Unlocking secure records with master key 5130985733435. User in (chrome user) sends this message: "HI"

POTATO
http://xkcd.com/1354/

SERVER, ARE YOU STILL THERE? IF SO, REPLY "BIRD" (4 LETTERS).

User Olivia from London wants pages about "tuna bees in car why". Note: Files for IP 375.381.383.17 are in /tmp/files-3843. User Meg wants these 4 letters: BIRD. There are currently 345 connections open. User Brendan uploaded the file selfie.jpg (contents: 834ba962e2eb06f89b13b6f69).

HMM...

User Olivia from London wants pages about "tuna bees in car why". Note: Files for IP 375.381.383.17 are in /tmp/files-3843. User Meg wants these 4 letters: BIRD. There are currently 345 connections open. User Brendan uploaded the file selfie.jpg (contents: 834ba962e2eb06f89b13b6f69).

BIRD
http://xkcd.com/1354/

SERVER, ARE YOU STILL THERE? IF SO, REPLY "HAT" (500 LETTERS).

User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User

User Meg wants these 500 letters: HAT. Lucas requests the "missed connections" page. Eve (administrator) wants to set server's master key to "14835038534". Isabel wants pages about "snakes but not too long". User Karen wants to change account password to "CoHoBaSt". User
Vulnerability Analysis and Disclosure

• What do you do if you’ve found a security problem in a real system?

• Say
  – A commercial website?
  – UW grade database?
  – Boeing 787?
  – TSA procedures?
Now for some cryptography!
Cryptography and Security

• Art and science of protecting our information.
  – Keeping it private, if we want privacy.
  – Protecting its integrity, if we want to avoid forgeries.

Images from Wikipedia and Barnes & Noble
Some Thoughts About Cryptography

• Cryptography only one small piece of a larger system
• Must protect entire system
  – Physical security
  – Operating system security
  – Network security
  – Users
  – Cryptography (following slides)
• “Security only as strong as the weakest link”
  – Need to secure weak links
  – But not always clear what the weakest link is (different adversaries and resources, different adversarial goals)
  – Crypto failures may not be (immediately) detected
• Cryptography helps after you’ve identified your threat model and goals
  – Famous quote: “Those who think that cryptography can solve their problems doesn’t understand cryptography and doesn’t understand their problems.”
Improved Security, Increased Risk

- RFIDs in car keys:
  - RFIDs in car keys make it harder to hotwire a car
  - Result: Car jackings increased
Improved Security

- RFIDs in car keys:
  - RFIDs in car keys
  - Result: Car jackins

Biometric car lock defeated by cutting off owner's finger

POSTED BY CORY DOCTOROW, MARCH 31, 2005 7:53 AM |
PERMALINK

Andrei sez, "'Malaysia car thieves steal finger.' This is what security visionaries Bruce Schneier and Ross Anderson have been warning about for a long time. Protect your $75,000 Mercedes with biometrics and you risk losing whatever body part is required by the biometric mechanism."

"...[H]aving stripped the car, the thieves became frustrated when they wanted to restart it. They found they again could not bypass the immobiliser, which needs the owner's fingerprint to disarm it.

They stripped Mr Kumaran naked and left him by the side of the road - but not before cutting off the end of his index finger with a machete."
A CRYPTO NERD’S IMAGINATION:

His laptop’s encrypted.
Let’s build a million-dollar cluster to crack it.

NO GOOD! IT’S 4096-BIT RSA!

BLAST! OUR EVIL PLAN IS FOILED!

WHAT WOULD ACTUALLY HAPPEN:

His laptop’s encrypted.
Drug him and hit him with this $5 wrench until he tells us the password.

GOT IT.