

CSE 484 / CSE M 584: Computer Security and Privacy

Software Security (finish) & Cryptography (brief intro)

Spring 2016

Franziska (Franzi) Roesner
franzi@cs.washington.edu

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

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()`

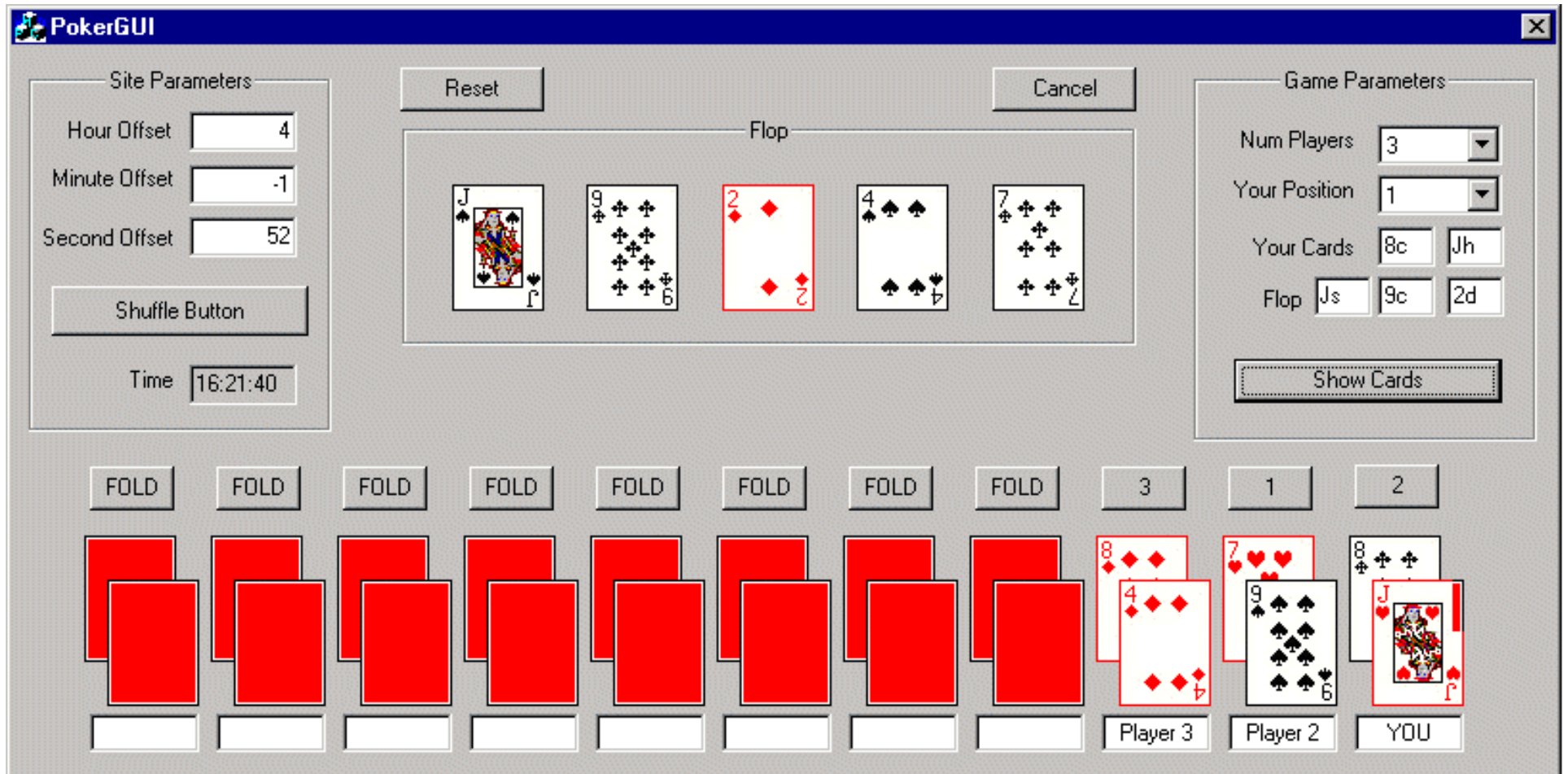
```
unsigned long int next = 1;
/* rand:  return pseudo-random integer on 0..32767 */
int rand(void) {
    next = next * 1103515245 + 12345;
    return (unsigned int)(next/65536) % 32768;
}
/* srand:  set seed for rand() */
void srand(unsigned int seed) {
    next = seed;
}
```

- 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?





More details: “How We Learned to Cheat at Online Poker: A Study in Software Security”
http://www.cigital.com/papers/download/developer_gambling.php

PS3 and Randomness

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

<http://www.engadget.com/2010/12/29/hackers-obtain-ps3-private-cryptography-key-due-to-epic-programm/>

- 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

DILBERT By SCOTT ADAMS



```
int getRandomNumber()  
{  
    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

```
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**

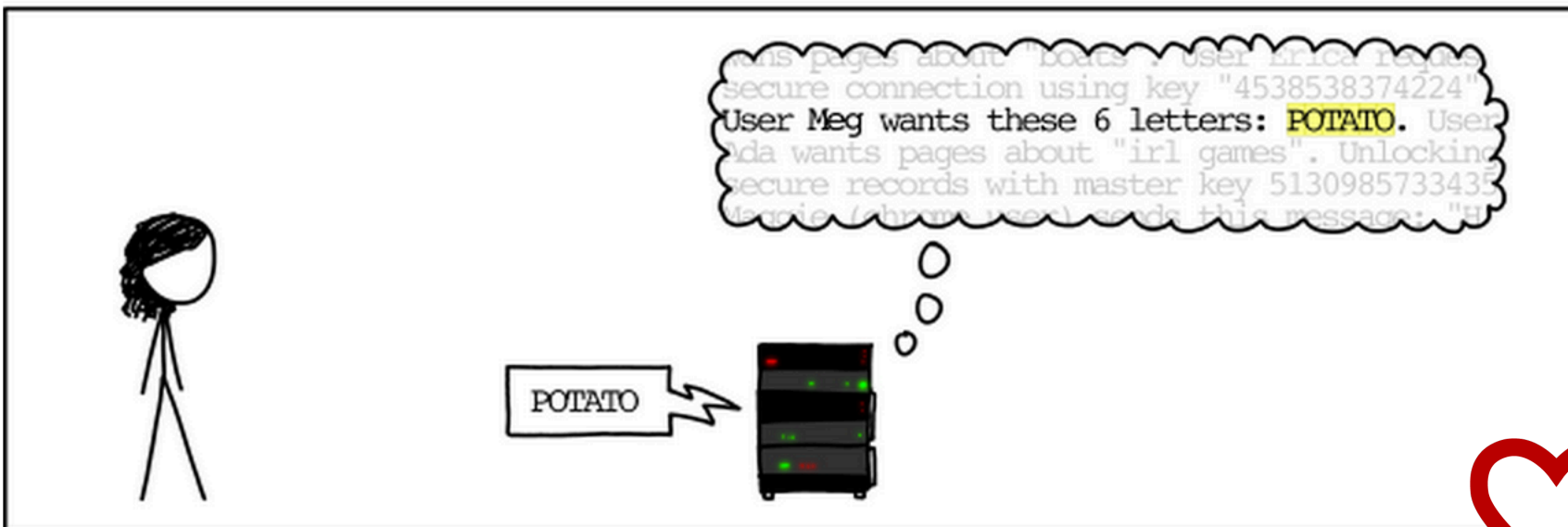
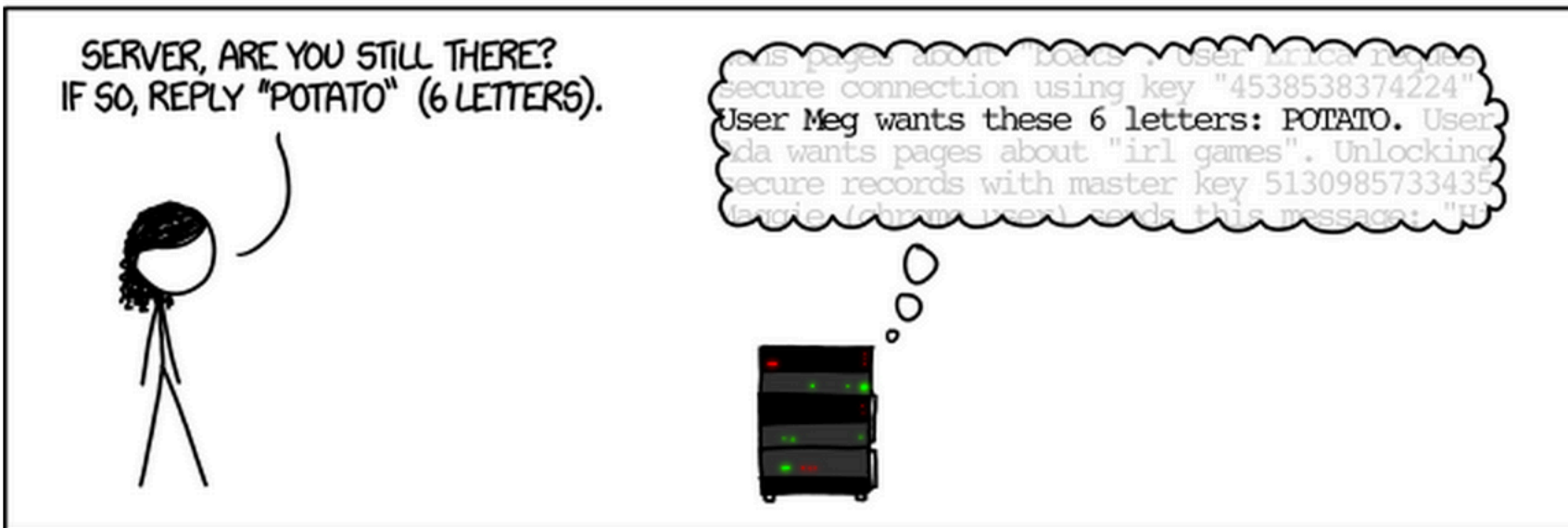
General Principles

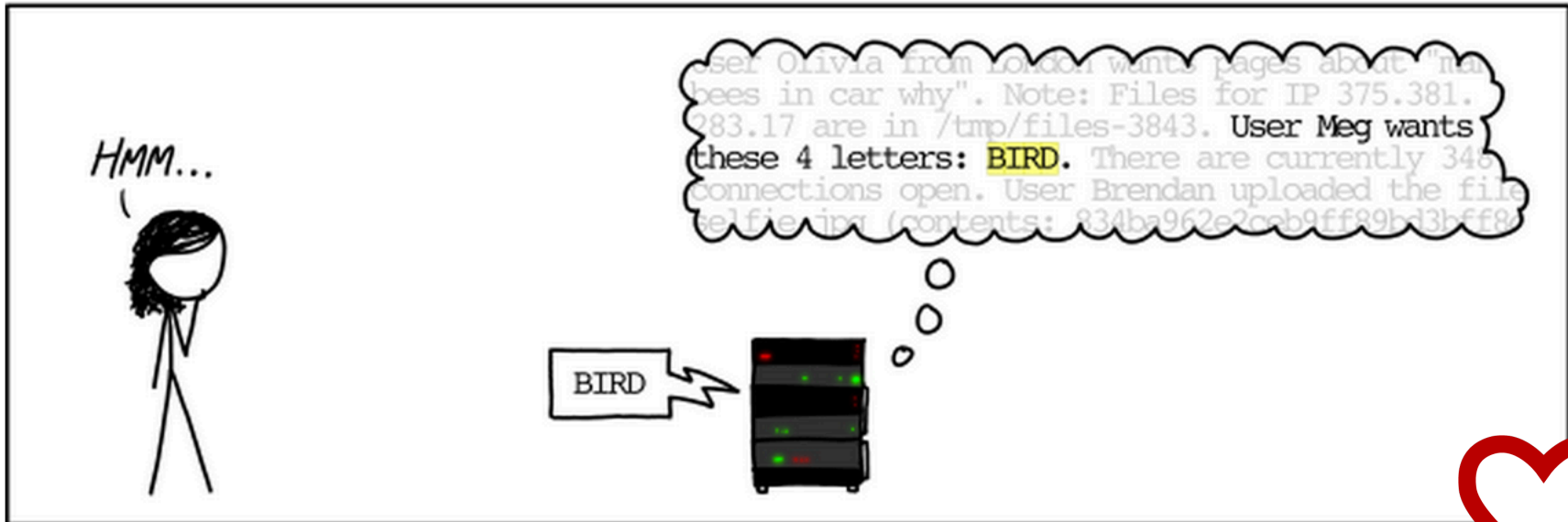
- Reduce size of trusted computing base (TCB)
- Simplicity, modularity
 - **But:** Be careful at interface boundaries!
- Minimize attack surface
- Use vetted component
- Security by design
 - **But:** tension between security and other goals
- Open design? Open source? Closed source?
 - **Different perspectives**

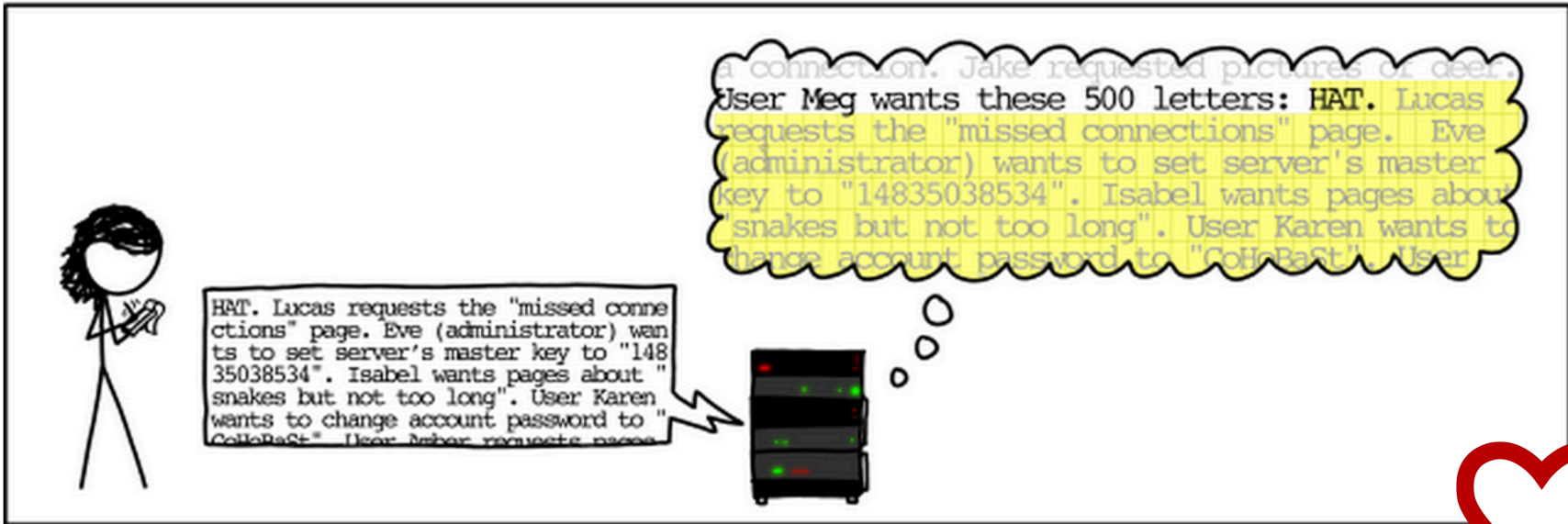
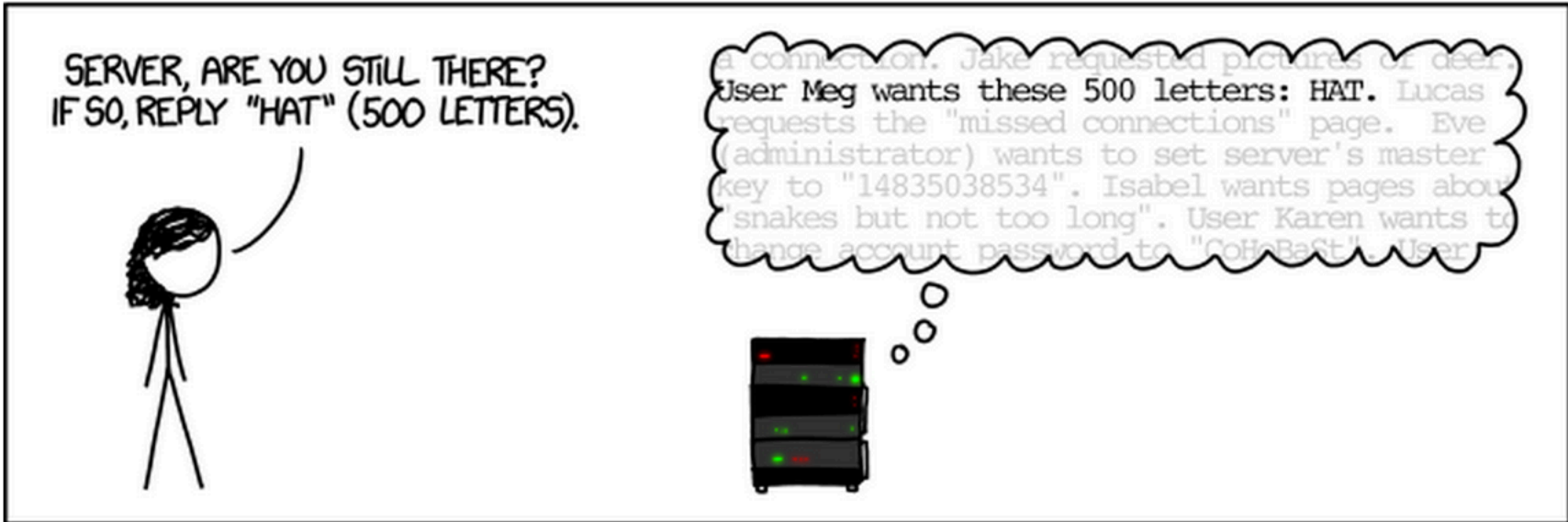
Does Open Source Help?

- Different perspectives...
- Happy example:
 - Linux kernel backdoor attempt thwarted (2003)
(<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)









Vulnerability Analysis and Disclosure

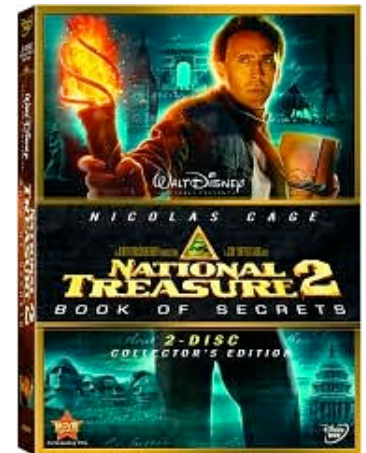
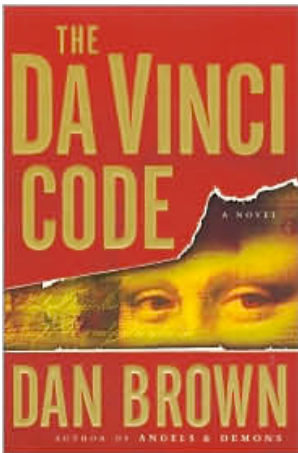
- What do you do if you've found a security problem in a real system?
- Say
 - A commercial website?
 - UW grade database?
 - Boeing 787?
 - TSA procedures?

Abj sbe fbzr pelcgbtencul!

Now for some cryptography!

Cryptography and Security

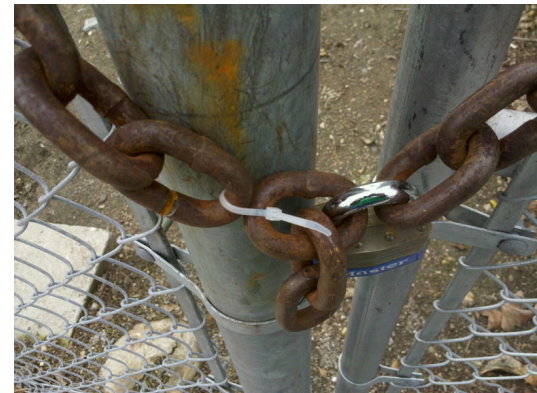
- Art and science of *protecting* our *information*.
 - Keeping it **private**, if we want privacy.
 - Protecting its **integrity**, if we want to avoid forgeries.



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 jackin
- 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.

XKCD: <http://xkcd.com/538/>

