Security

CS439: Principles of Computer Systems

April 29, 2015

Last Time

Deadlock Revisited
- Conditions for deadlock in the OS
- Preventing deadlock
- Avoiding deadlock
  - Banker's Algorithm
- Detecting deadlock
  - Resource Allocation Graph
- Real-life deadlock handling
  - Ostrich Algorithm

Today’s Agenda

Security

- Design Space
- Mechanisms: Authentication
- Typical Security Attacks: Rootkit, Buffer Overflow, Trojan Horses, …

Introduction

- Security is an engineering problem
- Always a tradeoff between safety, cost, and inconvenience
- Hard to provide any real guarantees
  - Because making mistakes is easy
  - And the nature of the problem implies that mistakes are always exploited

Security and Protection

- *Security* is a policy
  - E.g., “no unauthorized user may access this file”
- *Protection* is a mechanism
  - E.g., “the system checks user identity against access permissions”
- Protection mechanisms implement security policies

History of Security Problem

- Originally, there was no security problem
- Later, there was a problem, but nobody cared
- Now, there are increasing problems, and people are beginning to care

Security Threats

- Extremely wide range of threats
- From a wide variety of sources
- Requiring a wide variety of countermeasures
- Generally, countering any threat costs something
- So people frequently try to counter as few as they can afford

Physical Security

- Some threats involve access to the equipment itself
- Such as theft,

  destruction

  tampering
- Physical threats usually require physical prevention methods

Social Engineering and Security

- Computer security easily subverted by bad human practices
  - E.g., giving key out over the phone to anyone who asks
- Social engineering attacks tend to be cheap, easy, effective
- So all our work may be for naught

Security in Software

Fundamental Constraints of Practical Computer Security

- Security costs
  - If too much, it won’t be used
- If it isn’t easy, it won’t be used
- Misuse often makes security measures useless
- Fit the stringency of the measure to the threat being countered

Design Principle: Economy

- Economical to use (and develop)
- Should add little or no overhead
- Should do only what needs to be done
- Generally, try to keep it simple and small

Design Principle: Complete Mediation

- Apply security on every access to an object that a mechanism is meant to protect
  - E.g., each read of a file, not just the open
- Does not necessarily require actual checking on each access

Design Principle: Open Design

- Don’t rely on “security through obscurity”
- Assume all potential intruders know everything about the design
  - And completely understand it

Design Principle: Separation of Privileges

- Provide mechanisms that separate the privileges used for one purpose from those used for another
- To allow flexibility in the security system
- E.g., separate access control on each file

Design Principle: Least Privilege

- Give bare minimum access rights required to complete a task
- Require another request to perform another type of access
- E.g., don’t give write permission if user only asked for read

Design Principle: Least Common Mechanism

- Avoid sharing parts of the security mechanism among different users
- Coupling users leads to possibilities for them to breach the system

Design Principle: Acceptability

- Mechanism must be simple to use
- Simple enough that people will use it automatically
- Must rarely or never prevent permissible accesses

Design Principle: Fail-Safe

- Default to lack of access
- So if something goes wrong/is forgotten/ isn't done, no security is lost
- If important mistakes are made, you'll find out about them
  - Without loss of security

Security is as Strong as the Weakest Link

- Those breaking security will attack the weakest point
- Putting an expensive lock on a cheap door doesn’t help much
- Must look on security problems as part of an integrated system, not just a single component

Security Mechanisms

Authentication

Encryption

Access control mechanisms

Authentication

- If a system supports more than one user, it must be able to tell who's doing what
  – i.e., all requests to the system must be tagged with user identity
- *Authentication* is required to assure tags are valid
  – Passwords are a fundamental authentication mechanism

3 Pieces of Password Security

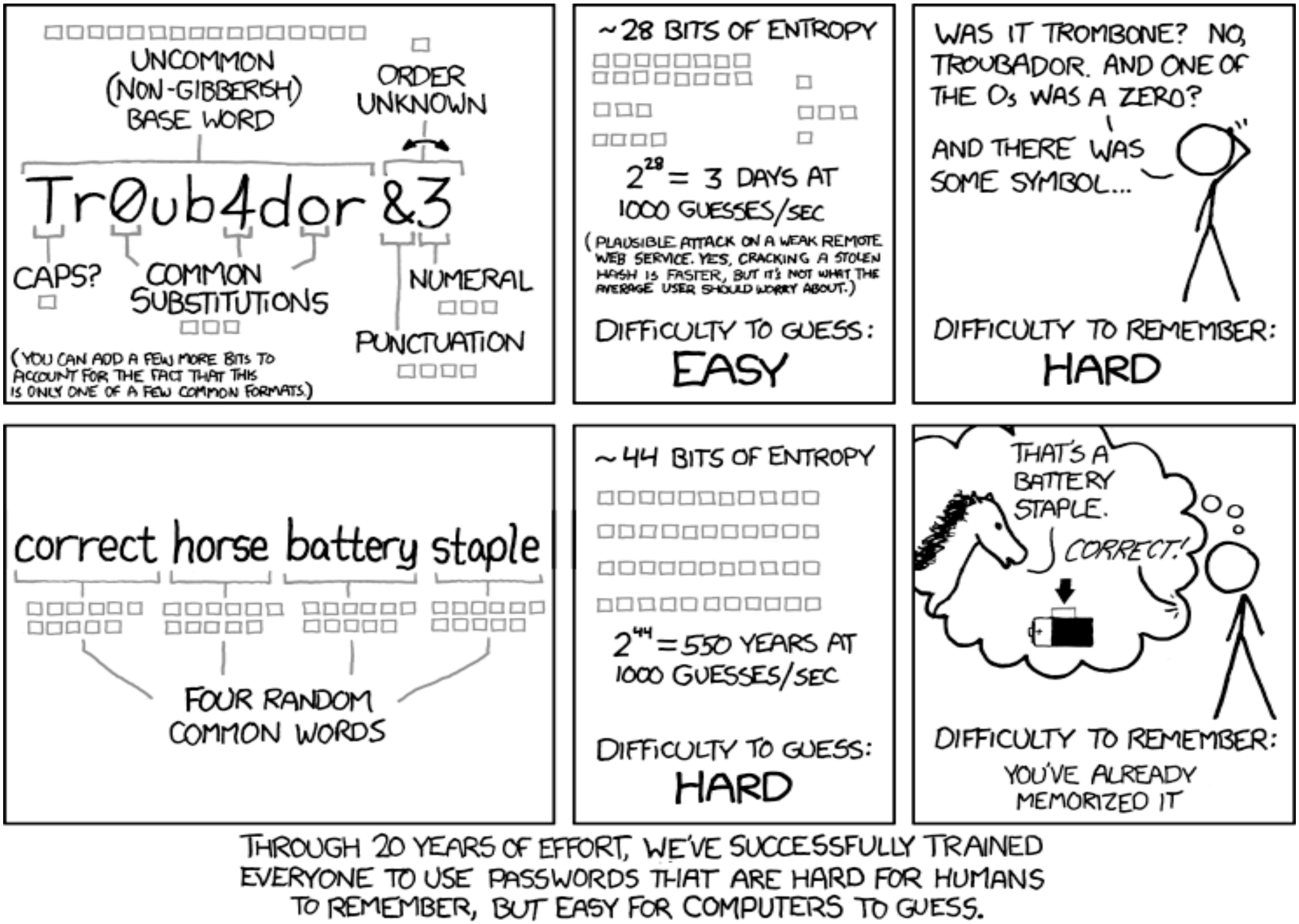
- Password selection
- Password storage and handling
- Password aging

Selecting a Password

Desirable characteristics include:

- Unguessable
- Easy to remember (and type)
- Not in a dictionary
- Too long to search exhaustively

XKCD


UNCOMMON (NON-GIBBERISH) BASE WORD
ORDER UNKNOWN
Tr0ub4dor &3
CAPS?
COMMON SUBSTITUTIONS
NUMERAL
PUNCTUATION
(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS.)
~28 BITS OF ENTROPY
2^28 = 3 DAYS AT 1000 GUESSES/SEC
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)
DIFFICULTY TO GUESS:
EASY
WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE Os WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER:
HARD
correct horse battery staple
FOUR RANDOM COMMON WORDS
~44 BITS OF ENTROPY
2^44 = 550 YEARS AT 1000 GUESSES/SEC
DIFFICULTY TO GUESS:
HARD
THAT'S A BATTERY STAPLE.
CORRECT!
DIFFICULTY TO REMEMBER:
YOU'VE ALREADY MEMORIZED IT
THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

XKCD: Text Description

- Panel 1: looking at the password someone has chosen according to the rules of a website
  - Password is: Tr0ub4dor&3
    - Has an uncommon (non-gibberish) base word (accounts for 16 bits of entropy)
    - Contains common substitutions (0 for o and 4 for a) (adds 3 bits of entropy)
    - Ends in punctuation (4 bits of entropy) and a number (3 bits of entropy) (could be switched in order, so 1 more bit of entropy)
    - Would have to guess if it started with a capital letter or not (so 1 more bit)
  - You could add a few more bits of entropy to account for the fact that this is only one of a few common formats to passwords
- Panel 2: showing how easy it would be a computer to guess this password
  - ~28 bits of entropy
  - 2^{28} = 3 days to compute at 1000 guesses/ sec
    - Plausible attack on a weak remote web service. Yes, cracking a stolen hash is faster, but it's not what the average user should worry about
  - Caption: difficulty to guess: EASY
- Panel 3: showing how hard the password is to remember
  - Man thinking to himself: "was it trombone? No, troubadour. And one of the Os was a zero? and there was some symbol…"
  - Caption: difficulty to remember: HARD
- Panel 4: looking at another kind of password
  - password is: correcthorsebatterystaple
    - Just 4 random common words
- Panel 5: showing how hard this would be to guess
  - ~44 bits of entropy
  - 2^{44} = 550 years at 1000 guesses/ sec
  - caption: Difficulty to guess: HARD
- Panel 6: showing how easy password is the remember
  - Man imagining a horse saying "that's a battery staple" and someone else replying "correct!"
  - caption: Difficulty to remember: you've already memorized it
- Caption for whole comic: through 20 years of effort, we've successfully trained everyone to use passwords that are hard for humans to remember, but easy for computers to guess.

Are Long Passwords Sufficient?

Example: Tenex system (1970s – BBN)

- Considered to be a very secure system
- Code for password check:

```
for (i=0, i<8, i++) {
      if (userPasswd[i] != realPasswd[i])
         Report Error;
}
```

- Looks innocuous – need to try 256^8 (= 1.8E +19) combinations to crack a password
- Is this good enough??

No!!!

Are Long Passwords Sufficient? Plain Text

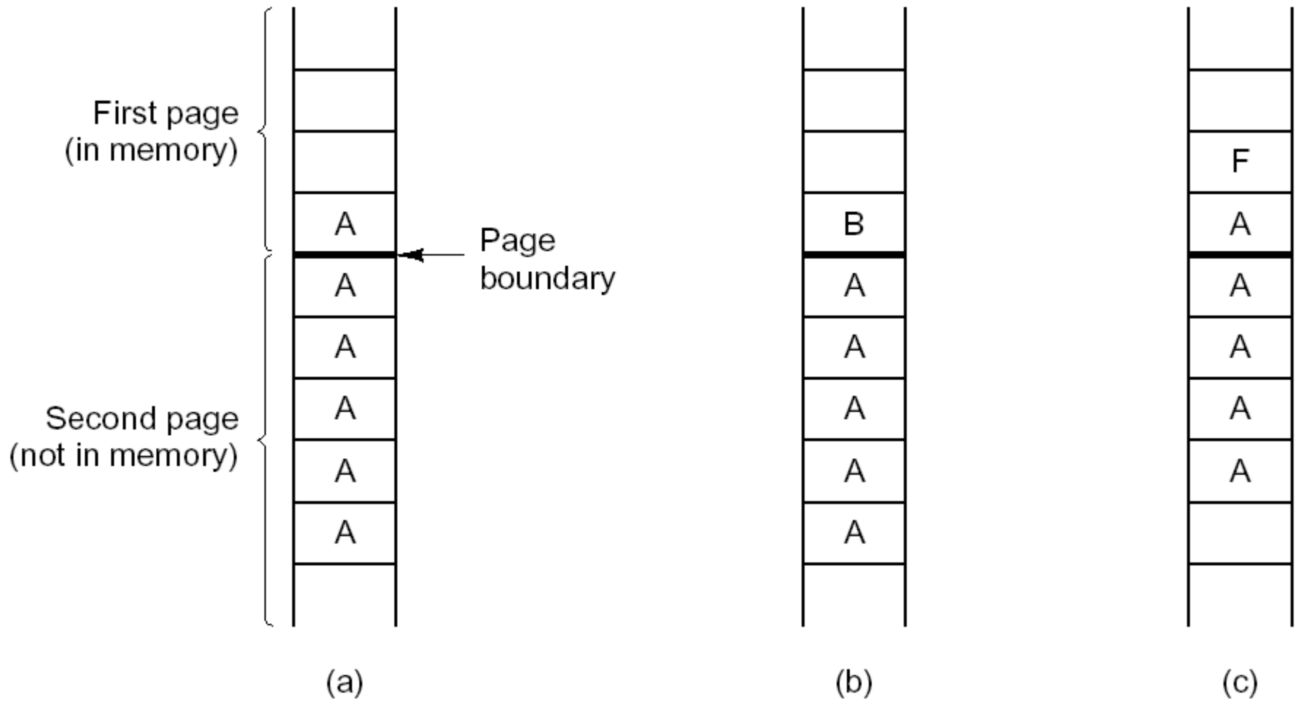
- Example: Tenex System (1970s - BBN)
  - Considered to be secure system
  - Code for password check:

```
for(i = 0; i < 8; i++)
    if(userPasswd[i] != realPasswd[i])
        Report Error
```

  - Looks innocuous---need to try 256^8 (= 1.8E+19) combinations to crack a password
  - Is this good enough??
    - NO!!

Story from Security Legend

The TENEX password problem



- Program presents a password to gain access to a file.
- A user can observe page faults by specifying a handler function.
- Password checking is done one character at a time.
  - Wrong character terminates password checking.
- By placing characters to be checked at edge of page, a page fault would signal a correct guess!

Story from Security Legend: The TENEX Password Problem Plain Text

- How to crack the passwords
  - Program presents a password to gain access to a file
  - A user can observe page faults by specifying a handler function
  - Password checking is done one character at a time. wrong character terminates password checking
  - By placing characters to be checked at edge of page, a page fault would signal a correct guess!

Securely Storing Passwords

- Systems must store passwords (in some form) to compare when users log in BUT don’t want authentication compromised if system is broken
- Store encrypted passwords (only!)
  - Check passwords by encrypting them and comparing to stored (encrypted) version
  - In Linux/Unix, these are stored in /etc/passwd
- But there are tricky issues (always)

Tricky Issues in Storing Encrypted Passwords

- How do I encrypt them?
  - Encryption key must be stored in the system
  - If I use single key to encrypt them all, what if the key is compromised?
- What if two people choose the same password?
  - Then a file inspection can reveal the match

Authentication Using Passwords

- System encrypts password x by a one-way function *f(x)*
- System checks password presented by applying *f* to the password presented and comparing the result with password file.
- Can still be hacked easily by precomputing many *f(x)* for likely passwords *x*.
  - Use *salting* to increase cost

Password Salting

Bobbie, 4238, e(Dog,4238)
Tony, 2918, e(6%%TaeFF,2918)
Laura, 6902, e(Shakespeare,6902)
Mark, 1694, e(XaB@Bwcz,1694)
Deborah, 1092, e(LordByron,1092)

Salt

Password

The use of salt defeats attacks that rely on precomputation of encrypted passwords

Password Salting: Plain Text

- The use of a salt (a random number that is encrypted with the password) defeats attacks that rely on precomputation of encrypted passwords
- Examples:
  - Deborah, 1092, e(LordByron, 1092)
    - 1092 is the salt and LordByron is the password
  - Bobbie, 4238, e(Dog, 4238)
  - Tony, 2918, e(6%%TaeFF, 2918)
  - Laura, 6902, e(Shakespeare, 6902)
  - Mark, 1694, e(XaB@Bwcz, 1694)

Does this solve the problem?

- Not entirely
- Passwords exist in plaintext in process checking them
- Passwords may travel over communication lines in plaintext
  – Especially for remote logins
  – Or logins over modems

Other Problems with Passwords

- People choose bad ones
- People forget them
- People reuse them
- People rarely change them

How to Deal with Bad Passwords

- Educate users so they choose good ones
- Automatic password generation (ugh)
- Check when changed
- Periodically run automated cracker

Any solution must balance user needs, password security, and resources

Other Authentication Mechanisms

- Challenge/response
- Smartcards
- Other special hardware
- Detection of personal characteristics

iClicker Question

Can you trust code that you write?

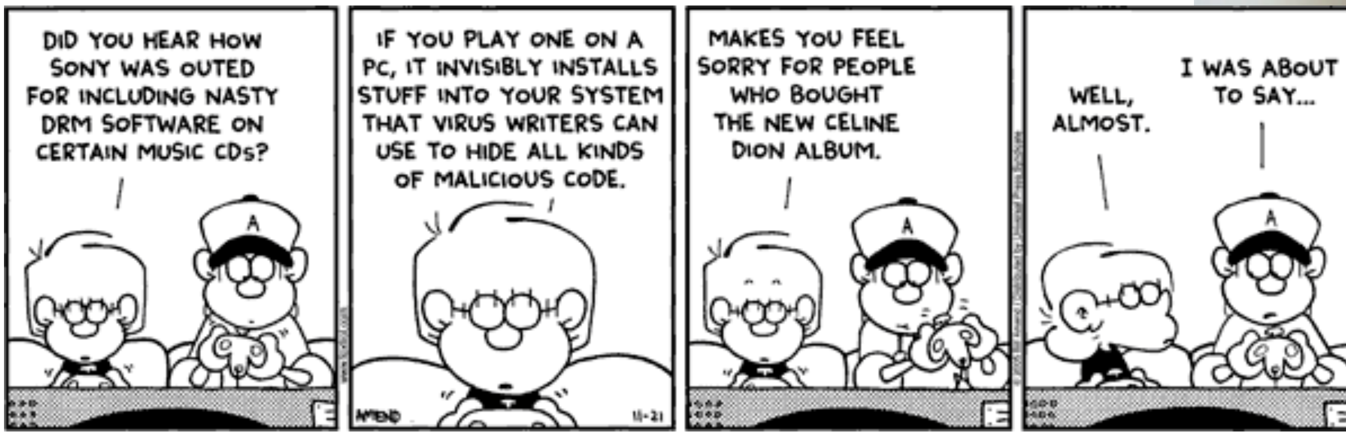
A. Yes
B. No

Sample Security Attacks

Typical Security Attack Methods

- Trojan Horse
  - Code segment that misuses its environment
  - Exploits mechanisms for allowing programs written by users to be executed by other users
  - Spyware, pop-up browser windows, covert channels
- Trap Door/Back door
  - Specific user identifier or password that circumvents normal security procedures
  - Could be included in a compiler
- Logic Bomb
  - Program that initiates a security incident under certain circumstances
- Stack and Buffer Overflow
  - Exploits a bug in a program (overflow either the stack or memory buffers)

The Sony Rootkit



- “Protected” albums included:
  - Billie Holiday
  - Louis Armstrong
  - Alicia Keys
  - Backstreet Boys
  - Kings of Leon
- Rootkits modify files to infiltrate and then hide
  - System configuration files
  - Drivers (executable files)

The Sony Rootkit

- Foxtrot Comic Strip
  - Panel 1:
    - Jason: “Did you hear how Sony was outed for including nasty DRM software on certain music CDs?”
  - Panel 2:
    - Jason: “If you play one on a PC it invisibly install stuff into your system that virus writers can use to hide all kinds of malicious code”
  - Panel 3:
    - Jason: “makes you feel sorry for people who bought the new Celine Dion album”
  - Panel 4:
    - Jason: “well almost”
    - Peter: “I was about to say…”
- “Protected” albums included:
  - Billie Holiday
  - Louis Armstrong
  - Alicia Keys
  - Backstreet Boys
  - Kings of Leon
- Rootkits modify files to infiltrate and then hide
  - System configuration files
  - Drivers (executable files)

The Sony Rootkit

- Sony’s rootkit enforced DRM but exposed computer
  - CDs recalled
  - Classified as spyware by anti-virus software
  - Rootkit removal software distributed
  - Removal software had exposure vulnerability
  - New removal software distributed
- Sony sued by 39 states

Ken Thompson

- Created the UNIX Operating System with Dennis Ritchie
- Created programming language “B”
  - What came next?
- 1983 Turing Award
  - Turing Award Lecture titled “Reflections on Trusting Trust”
- 1999 National Medal of Technology

Reflections on Trusting Trust

- “I am a programmer. On my 1040 form, that is what I put down as my occupation. As a programmer, I write programs.”
- “I would like to present you with the cutest program I ever wrote.”

Trusting Trust: Some Observations

- Stage I
  - A program can, when executed, output its own source code
- Stage II
  - A compiler can learn the meaning of a symbol
- Stage III
  - A compiler may (deliberately) output incorrect machine code

A Self-Reproducing Program

```
main()
{
    char*s="main(){ char*s=%c%s%c; printf(x, 34, s, 34);}"
    printf(s, 34, s, 24);
}
```

A Learning Compiler

```
compile(s)
char *s;
{
        if(match(s, "pattern")) {
                compile("bug");
                return;
        }
        ...
}
```

If the input program s is an Unix OS, compile in the trapdoor. The bug in this case is the code of the Unix “login” command altered to allow Ken Thompson to login.

See article in http://www.acm.org/classics/sep95/

A Learning Compiler: Plain Text

- Code:

```
compile(s)
char *s;
if(match(s, “pattern”))
{
   compile(“bug”);
   return;
}
```

- If the input program s is an Unix OS, compile in the trapdoor. The bug in this case is the code of the Unix “login” command altered to allow Ken Thompson to login
- See article in http://www.acm.org/classics/sep95/

Ken Thompson’s Unix Trap Door

- First we compile the modified source with the normal C compiler to produce a bugged binary.
- We install this binary as the official C compiler.
- We can now remove the bugs from the source of the compiler and the new binary will reinsert the bugs whenever it is compiled.
- Of course, the login command will remain bugged with no trace in source anywhere.

Moral

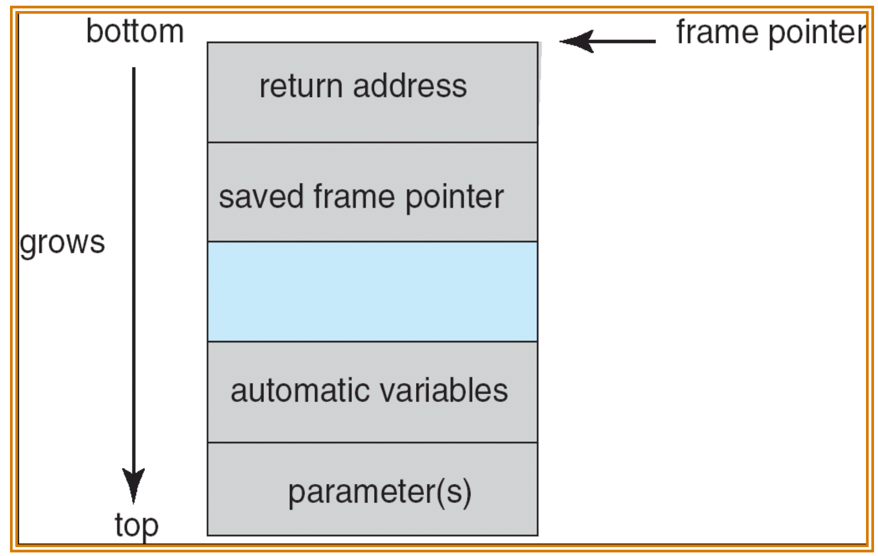
“The moral is obvious. You can't trust code that you did not totally create yourself. (Especially code from companies that employ people like me.) No amount of source-level verification or scrutiny will protect you from using untrusted code.”

Where is Ken Thompson now?

Typical Security Attacks: Buffer Overflow

```
#include <stdio.h>
#define BUFFER SIZE 256
int main(int argc, char *argv[])
{
   char buffer[BUFFER SIZE];
   int  other_data;

   if (argc < 2)
    return -1;
   else {
    strcpy(buffer,argv[1]);
    return 0;
   }
}
```



Typical Security Attacks: Buffer Overflow Plain Text

- Example of Buffer Overflow Waiting to Happen (code):

```
#include <stdio.h>
#define BUFFER_SIZE 256
int main(int argc, char *argv[]) {
        char buffer[BUFFER_SIZE];
        int other_data;
        if(argc < 2)
                return -1;
        else {
                strcpy(buffer, argv[1]);
                return 0; }
}
```

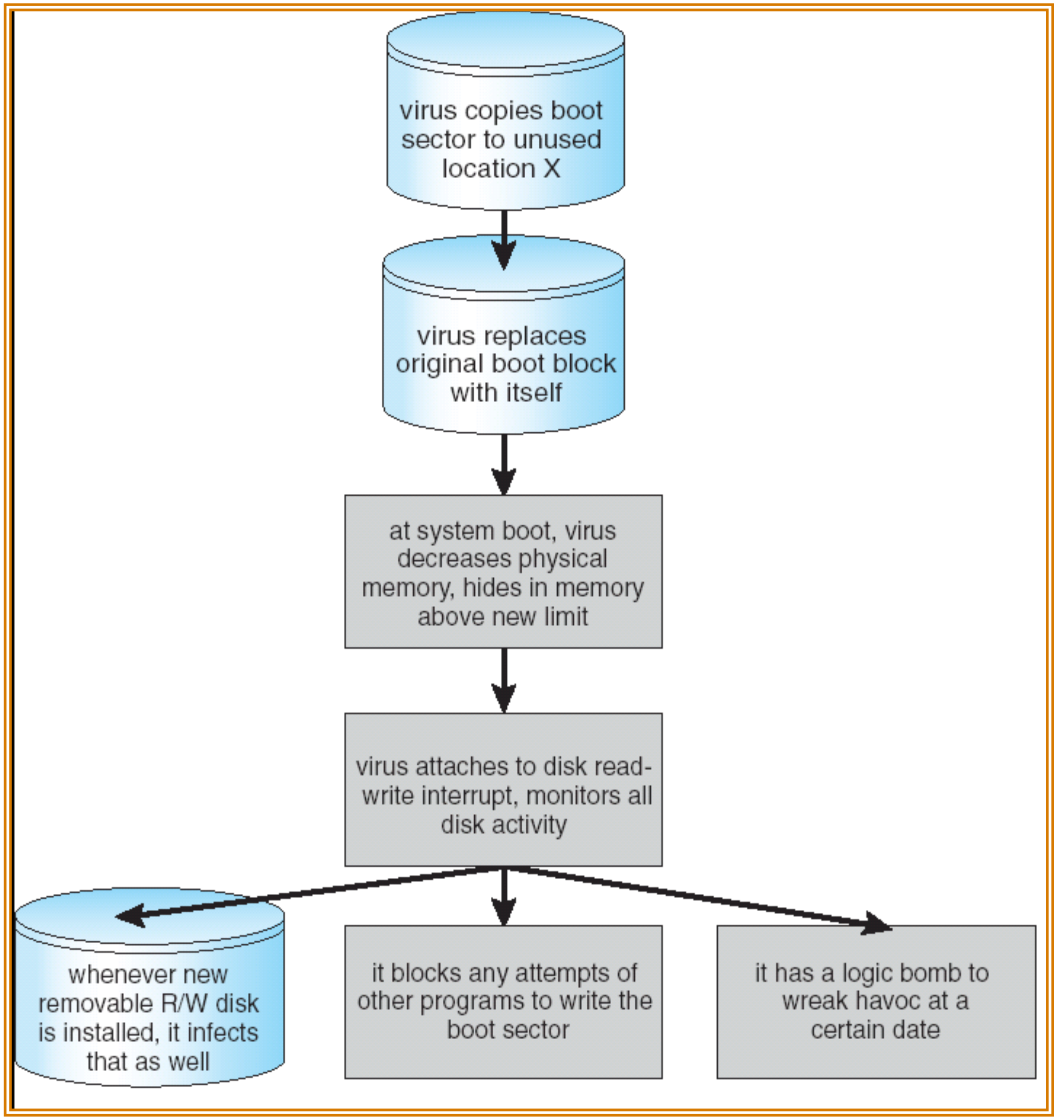
- Setup of Stack:
  - Parts listed from bottom to top, and stack is growing towards the top
  - Parts:
    - Return address
    - Saved frame pointer
    - OPEN SPACE
    - Automatic variables
    - Parameters
  - The frame pointer is pointing to the bottom of the return address

Typical Security Attacks: Viruses

- Code fragment embedded in legitimate program
- Very specific to CPU architecture, operating system, applications
- Usually borne via email or as a macro
  - Visual Basic Macro to reformat hard drive

```
Sub AutoOpen()
Dim oFS
    Set oFS = CreateObject(“Scripting.FileSystemObject”)
    vs = Shell(“c:command.com /k format  c:”,vbHide)
End Sub
```

Typical Security Attacks: Boot Sector Virus



Typical Security Attacks: Boot Sector Virus
Plain Text

- Virus copies boot sector to unused location X
- Virus replaces original book block with itself
- At system boot, virus decreases physical memory, hides in memory above new limit
- Virus attaches to disk read-write interrupt, monitors all disk activity
- Now executes 3 possible actions:
  - Whenever a new removable R/W disk is installed, it infects that as well
  - It blocks any attempts of other programs to write the boot sector
  - It has a logic bomb to wreak havoc at a certain date

Typical Security Attacks: Internet Worm

- Self-replicating program that exploits errors and replicates itself
- Spreads itself (unlike viruses)
- First one developed by Cornell grad student, Robert Morris Jr. in 1988
- Exploited errors in rsh (no authentication), finger (buffer overflow), sendmail (execute mailed bootstrap)
- Once established, began breaking passwords

Stuxnet

- A computer worm discovered in June 2010
  - Spreads via Microsoft Windows
  - Targets Siemens industrial software and equipment.
- Initially spread using infected removable drives (such as USB flash drives)
- Then uses other exploits and techniques to infect and update other computers
  - Allows it to spread inside private networks that are not directly connected to the Internet.
- The malware has both user-mode and kernel-mode rootkit capability under Windows
- Its device drivers have been digitally signed with the private keys of two certificates that were stolen from two separate companies.
  - Helped it install kernel mode rootkit drivers successfully and therefore remain undetected for a relatively long period of time.

Siemens Simatic S7-300 PLC CPU with three I/O modules

Stuxnet

- Once installed, Stuxnet:
  - Infects files belonging to Siemens' control software
    - This control software runs the machines in the Iranian nuclear program
  - Subverts a communication library
    - Intercepts communications between software running under Windows and the target Siemens devices
- Stuxnet malware periodically modifies a control frequency to and thus affects the operation of the connected centrifuge motors by changing their rotational speed.
  - This causes the centrifuges to be destroyed.

Covert Steganography Channel

- Pictures appear the same
- Picture on right has text of 5 Shakespeare plays
  - encrypted, inserted into low order bits of color values

Zebras

Hamlet, Macbeth, Julius Caesar
Merchant of Venice, King Lear

Covert Stenography Channel: Text Description

- Pictures appear the same (innocent zebras standing in African landscape)
- Picture on right has text of 5 Shakespeare plays embedded in it
  - Hamlet, Macbeth, Julius Caesar, Merchant of Venice, and King Lear
  - encrypted, inserted into low order bits of color values

Summary

- Determining a security policy is easy
- Protection is hard
- Difficult to re-secure after a breach
- Hard to detect a breach
- Any system with bugs has loopholes

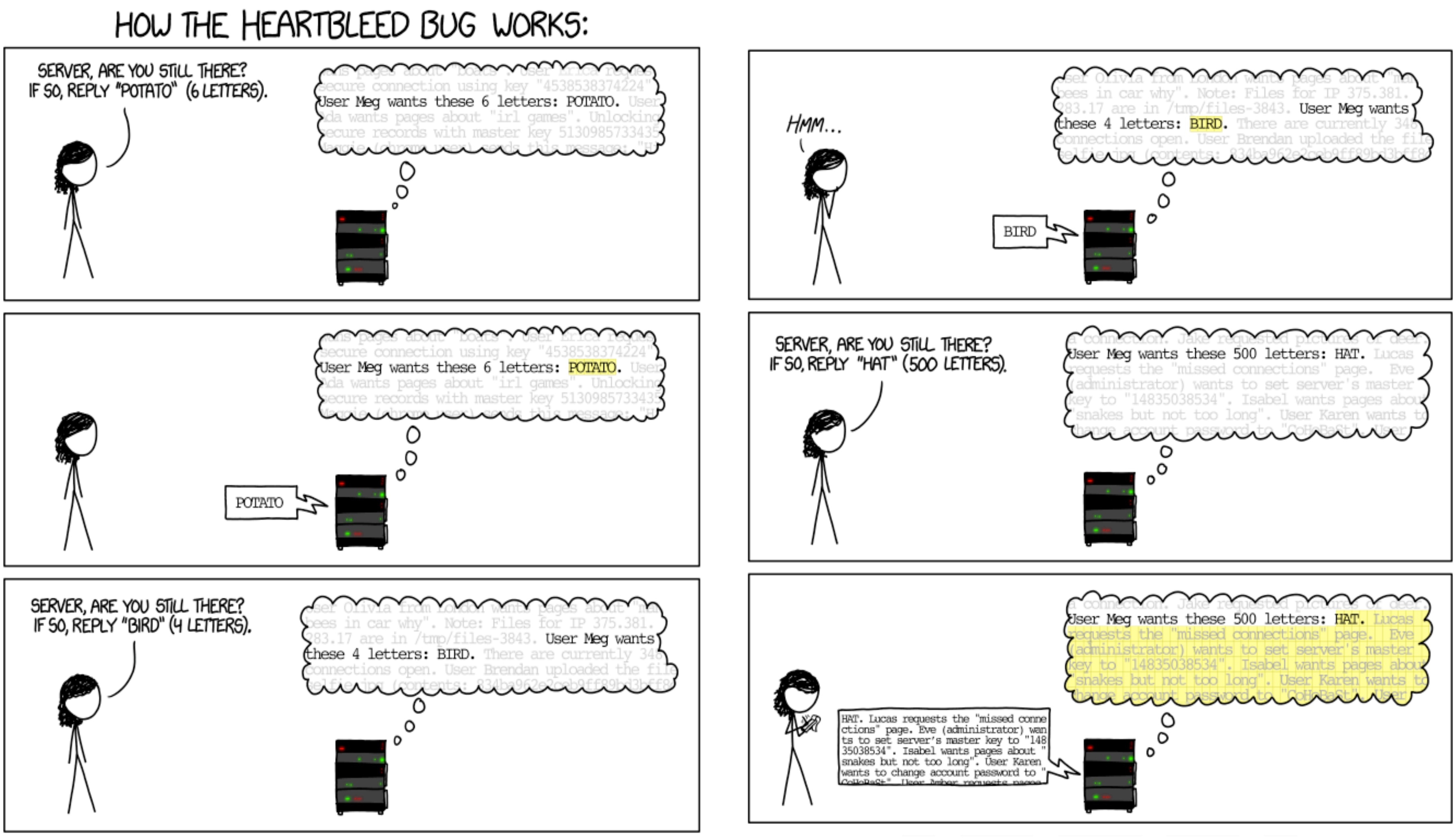
Announcements

- Project 4 (Last one!) due Friday, 5/8, 11:59p
  - No slip days!
- If you have a conflict for the final, you should have already contacted me (email, please!) AND gotten a response
  - Thursday, May 14th, 7p-10p in UTC 2.102A

Password Guessing

- Password guessing is often successful with clever social engineering, e.g., bad guy pretends to be computer repairman and asks unwary secretary to borrow staff password.
- Long and “random” passwords are hard to remember; guesser can narrow the range of passwords to guess by characterizing easy passwords with a grammar.

Heartbleed Explained



SECURITY

Typical Security Attacks

System And Network Threats

- Worms – use **spawn** mechanism; standalone program
- Internet worm
  - Exploited UNIX networking features (remote access) and bugs in *finger* and *sendmail* programs. (See next slide)
  - **Grappling hook** program uploaded main worm program
- Port scanning
  - Automated attempt to connect to a range of ports on one or a range of IP addresses
- Denial of Service
  - Overload the targeted computer preventing it from doing any useful work
  - Distributed denial-of-service (**DDOS**) come from multiple sites at once