SWE 781
Secure Software Design and Programming

Introduction

Lecture 1

IATAC

Ron Ritchey, Ph.D.
Chief Scientist
703/377.6704
Ritchey_ronald@bah.com

Introduction

- This course will discuss how to engineer secure software. The approach taken will be to highlight common security pitfalls that application developers frequently fall into so that these mistakes may be avoided in the future. We will also introduce modern topics in computer security that shall prepare students to do research in computer security
- The course will be somewhat Unix specific.
  - Unix provides a good platform for demonstrating many secure programming problems.
  - Both the operating system and the tools required for program development are freely available

IATAC

Agenda

- Syllabus
- Grading
- Discussion of information security
- Overview of information security terms
  - The CIA triangle
  - AAA
- Overview of the Unix security model

IATAC

Reading List

Primary Texts

- Brian Chess and Jacob West, Secure Programming with Static Analysis
- D. Wheeler, Secure Programming for Linux and Unix HOWTO

Secondary Materials

- Goertzel et al, Software Security Assurance State of the Art Report, May 2007
- Aleph One, Smashing the Stack for Fun and Profit. Phrack Vol 7, Nr. 49
- Tim Newsham, Format String Attacks, Guardent tech report, Sept 2000
- Bugtraq - http://www.securityfocus.com/archive/1
- Phrack - http://www.phrack.org/
- Other materials may be identified as class progresses

IATAC

Schedule (tentative)

Date	Subject
Sep 1st	Introduction (today) ; Chess/West chapter 1, Wheeler chapters 1,2,3
Sep 8th	Computer attack overview
Sep 15th	Input Validation; Chess/West chapter 5, Wheeler chapter 5
Sep 22nd	Buffer Overflows; Chess/West chapters 6, 7; Wheeler chapter 6
Sep 29th	Error Handling; Chess/West chapter 8; Wheeler chapter 9 (9.1, 9.2, 9.3 only)
Oct 6th	Privacy, Secrets, and Cryptography; Chess/West chapter 11; Wheeler chapter 11 (11.3, 11.4, 11.5 only)
Oct 13th	Columbus Recess
Oct 20th	Mid-Term exam
Oct 27th	Mid Term Review / Major Assignment Introduction
Nov 3rd	Implementing authentication and access control
Nov 10th	Web Application Vulnerabilities; Chess/West chapter 9,10
Nov 17th	Secure programming best practices / Major Assignment Stage Check ; Chess/West chapter 12; Wheeler chapters 7,8,9,10
Nov 24th	Static Code Analysis & Runtime Analysis
Dec 1st	The State of the Art (guest lecturer)
Dec 8th	TBD (Virtual Machines, Usability [phishing], E-Voting, Privilege Separation, Java Security, Network Security & Worms)

IATAC

Grading

- Minor assignments (20%)
  - At least four will be assigned
  - The first is a one page review of an application vulnerability from bugtraq.
- Mid Term Exam (30%)
- Major assignment / Final (50%)

Warning

- Widely-deployed vulnerabilities may be discuss and this is in no way intended as an invitation to go and exploit these vulnerabilities. You are to behave responsibly, but it is important that we candidly discuss real-world experience.
- You must abide by the University's computer usage policy

IATAC

1st minor assignment

- Task: Explore examples of insecure code using NIST's SAMATE database
- Detail
  - The NIST SAMATE project is collecting examples of vulnerable code. It is designed to be used to "provide users, researchers, and software security assurance tool developers with a set of known security flaws [1]". Your assignment is to browse through this collection, choose one of the code examples, determine why it is vulnerable and what could be done to produce a secure equivalent. You will be required to write up your findings in a one to two page report. Please make sure that you clearly identify which code example you have chosen, explain the vulnerability and your approach to solving it. If the code example is small enough, please include it as an attachment to your report. As in the real world, good grammar and spelling is required!
  - This is an individual assignment. No group work is allowed.
  - This assignment, as with all assignments in this class must be performed in STRICT COMPLIANCE to the honor code!

  [1] NIST SAMATE Reference Dataset Project, http://samate.nist.gov/SRD/.
- Due Date: Sept 8th

IATAC

Agenda

- Syllabus
- Grading
- Discussion of information security
- Overview of information security terms
  - The CIA triangle
  - AAA
- Overview of the Unix security model

IATAC

Why do we need secure programs?

Monster says millions of users' data may be stolen

Wed Aug 29, 2007 6:19PM EDT

Email | Print | Digg | Reprints | Single Page | Recommend (0) [-] Text [+]

1 of 1 Full Size

NEW YORK/BOSTON (Reuters) - The theft of contact information for job seekers in the database of Monster Worldwide Inc may have been much greater than the 1.3 million individuals reported earlier this month, Chief Executive Sal Iannuzzi said on Wednesday.

While investigating the recent theft, the company learned that its Web site had previously been hacked.

"We're assuming it is a large number. It could easily be in the millions," Iannuzzi said in an interview with Reuters.

To be safe, he said, each Monster.com user should assume that his or her contact information has been taken.

The company said earlier the theft of confidential information was not an isolated incident, and said the scope of illegal activity was impossible to pinpoint.

Monster is stepping up surveillance of site traffic, boosting its security staff, and is contacting users about ways of protecting their privacy.

Cost of data breach at TJX soars to $256m

The Boston Globe

Suits, computer fix add to expenses

By Ross Kerber, Globe Staff | August 15, 2007

TJX Cos. said its costs from the largest computer data breach in corporate history, in which thieves stole more than 45 million customer credit and debit card numbers, have ballooned to $256 million.

The figure is more than 10 times the roughly $25 million the Framingham retailer estimated just three months ago, though at the time it cautioned it didn't know the full extent of its exposure from the breach.

The costs include fixing the company's computer system and dealing with lawsuits, investigations, and other claims stemming from the breach, which lasted more than a year before the company discovered the problem in December.

IATAC

Annual cost of computer crime *

- $67 Billion
- Sixty-four percent of respondents (primarily large corporations and government agencies) detected computer security breaches within the last twelve months.
- $24,000 was the average cost per company with total reaching $32M
- Worms, Viruses and Trojan horses were the most costly, followed by computer theft, financial fraud and network intrusions
  - $12M to virus-type incidents
  - $3.2 M to theft
  - $2.8M to financial fraud
  - $2.7M to network intrusions

IATAC

* 2005 FBI Computer Crime Survey

Should you trust your users?

- Who are your users?
- How do you know?
- How do you prevent unauthorized users from accessing your application?
- Can your application support unexpected input gracefully?
- We frequently make assumptions about how our users will use our applications.
  - How often do we succeed?

IATAC

Why is secure software hard?

- Most of the focus during a development project is on functionality not security
- Developers not good at thinking of other ways their software may be used (or abused)
- Software has bugs!
  - A friendly user stumbles on bugs when presenting input the developers did not consider/test for
  - A malicious user seeks out bugs and attempts to exploit them
- Software programmers not sufficiently paranoid
  - Do not consider the malicious user

IATAC

Too much focus on security late lifecycle results in bolted-on, incomplete solutions

Focus purely on functionality guarantees insecure systems

Plan | Build | Test | Field

Firewalls
Intrusion Detection Systems
Penetration Testing

Designing security in from the beginning gives us a chance at long-term security

Plan | Build | Test | Field

Security Requirements
Architecture Risk Assessment
Static Code Analysis
Formal analysis

IATAC

Seven Pernicious Kingdoms *

- Input Validation and Representation
- API Abuse
- Security Features
- Time and State
- Error Handling
- Code Quality
- Encapsulation
- CWE/SANS TOP 25 Most Dangerous Programming Errors
  - http://www.sans.org/top25errors/

* From Tsipenyuk, Chess, Mcgraw, "Seven Pernicious Kingdoms: A Taxonomy of Software Security Errors". *Proceedings SSATTM, 2005*

Ok, why you should care!

Home | Search Jobs | My Monster | Career Center | Help | For Employers

monster Jobs

(security software) & (not sales)

All Jobs | Subsearch

[New Search]

Want to narrow your results? You can omit certain words, phrases or companies from your search by typing NOT, then the word you want to omit, in the box above. For more, see HELP.

Jobs 1 to 50 of 121

	Date	Location	Job Title	Company
1.	May 26	US-CA-Sunnyvale	Senior Java Security Software Architect	Bobbi Sanchez
2.	May 25	US-TX-Dallas	Systems Security Analyst	TMP Worldwide eResourcing
3.	May 25	US-MA-Framingham/Worcester	Network Security Software Developer	TMP Worldwide eResourcing
4.	May 25	US-OH-Cleveland	Network Planning Analyst	Kforce, Professional Staffing
5.	May 25	US-MA-Boston	Security Systems Architect Wanted!	Management Recruiters Intntl
6.	May 25	US-GA-Atlanta	AS400 Technician	Railcar Management
7.	May	US-IL-wood dale	Admin. Asst. Facilities	AAR Corporation

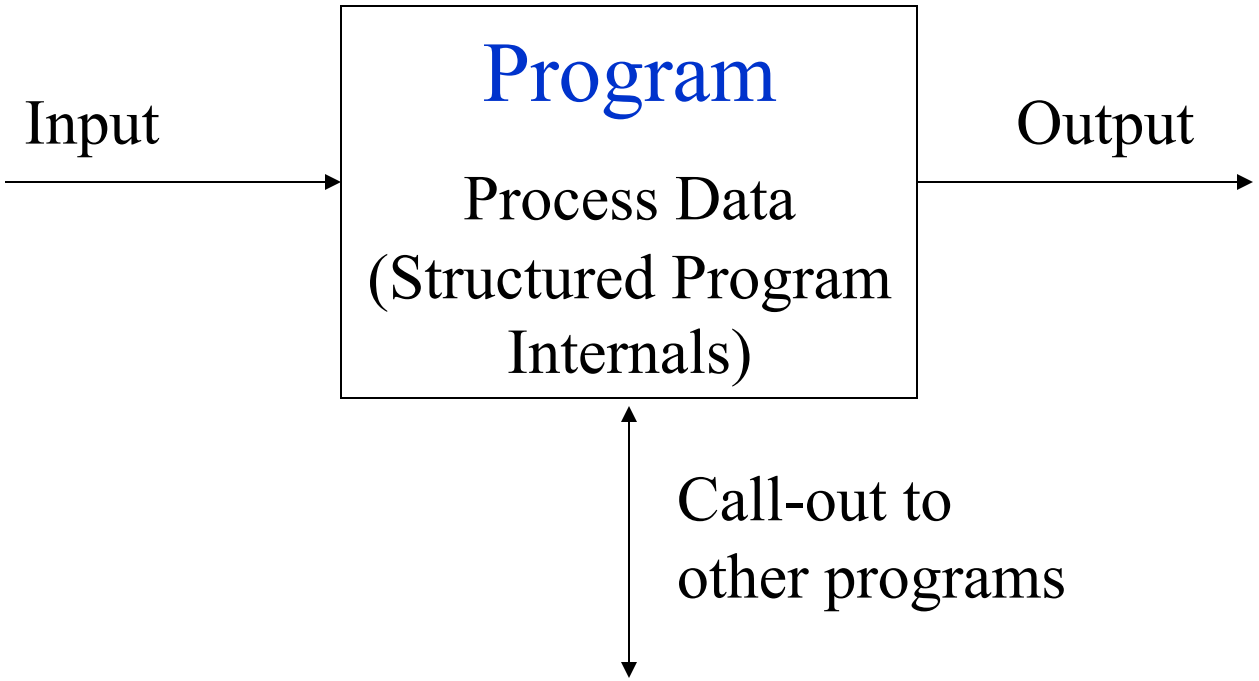
IATAC

Agenda

- Syllabus
- Grading
- Discussion of information security
- Overview of information security terms
  - The CIA triangle
  - AAA
- Overview of the Unix security model

IATAC

Diagram of a Program



Information Security Properties

- Information Security is frequently broken down into three main categories
  - **Integrity**
    - Must be accurate and complete
  - **Confidentiality**
    - Must only be revealed to authorized users
  - **Availability**
    - Must be reliably available when needed
- The programs your write should maintain these properties as appropriate for functionality

IATAC

Integrity

- It is essential to the operation of a information system that the information is not damaged, destroyed or falsified
  - System Failures
    - Should allow graceful recovery to a known stable state
  - Modification
    - Should only be allowed by trusted processes / users
  - Consistency
    - Should be maintained between the information system and the realities of the outside world

IATAC

Confidentiality and Availability

- Many types of information should not be accessible to unauthorized individuals
  - Logical access controls can be used to protect the information from unauthorized release
  - Encryption can be used to protect against eavesdropping and capture
- Information must be available on a timely basis
- Controls are application and system dependent

IATAC

Authentication, Authorization, and Auditing (AAA)

- Authentication
  - Proving identity
- Authorization
  - Granting access to resources based upon identity
  - Based upon User, User Group, Resource, and Resource Group
- Auditing
  - Recording the normal and abnormal operation of the system

IATAC

Authentication

- Asks the question, who are you?
- Critical for integrity and confidentiality
- You can use three types of data when attempting authentication
  - Something the user knows
  - Something the user has
  - Something the user is
- Most popular type is Username / Password
  - Type: Something the user knows
  - Default ability on most modern operating systems
- Is there anything wrong with password based authentication?

IATAC

Passwords = weak authentication

- Do users create good passwords? Why not?
  - Too short
  - Based upon user's environment (family, car, sports teams, etc.)
  - Based upon words found in dictionaries
  - Tend to contain only letters and numbers
  - Think that trivial substitutions make better passwords
    - Ralph becomes Ralph1, School becomes Sch00l
- Are system generated passwords better?
  - If passwords are too hard to remember, what will users do?
  - How many passwords do you have to remember?
- Passwords can be eavesdropped
  - Shoulder surfing
  - Networking sniffing

IATAC

Password quality criteria

- Length
  - Longer the better
    - Less than 7 characters can be brute forced in < 24 hours
- Complexity
  - Using more than two character types (letters, numbers, symbols) e.g. 3yaq!tun
- Not dictionary based
  - The password does not appear in a dictionary and is not a trivial substitution of a dictionary term

IATAC

Password guessing methods

- Brute Force
  - Attempts to break the password by attempting every possible character combination
  - Made difficult by length, complexity
- Dictionary Attack
  - Uses a password dictionary to guess passwords
  - Sophisticated dictionary based password crackers will create guess for permuted terms
  - Thousands of dictionaries available on the web
    - English, German, French, Swedish, etc.
    - Biology, chemistry, Shakespeare
    - Star Trek, The Matrix, sexual terms
  - Made difficult by complexity, non-dictionary based passwords

IATAC

What's better than passwords?

Basically everything but here are some specifics

- One-time passwords
- Shared secret
- Public Key
- Tokens and Smart cards

IATAC

One-time passwords

- Issue the user a bunch of passwords, then only accept each one once.
- Eliminates the problem of shoulder surfing and network eavesdropping but, …
- Introduces the problem of distributing the password lists
- If the list is compromised the user can still be impersonated
- Users hate one-time passwords

IATAC

Shared Secret

- User and server have a shared secret
- To authenticate
  - Server generates a random number and sends it to the client.
  - Client encrypts the number and returns it to the server
  - Server decrypts the number using the shared secret key.
  - If numbers match, the user is authenticated
- Prevents eavesdropping
- Does not provide non-repudiation
- If secret is compromised then user can be impersonated

IATAC

Public Key

- Uses public key cryptography to authentication
- User is issued a public/private key pair
- Server only knows the users public key
- To authentication
  - Server generates random number and sends to client
  - Client encrypts random with private key and sends to server.
  - Server decrypts with public key.
  - If numbers match then user is authenticated
- If users private key is compromised the user can be impersonated

IATAC

Tokens and Smartcards

- Hardware assisted authentication
- Many types of varying sophistication
  - Challenge Response: Servers sends challenge which is entered into token. Token computes password to return to server
  - Time-based challenge response: The current time is the challenge value
  - Smartcards: Contain the users credentials
    - Can be dumb, I.e. gives up credentials to the requesting device
      or
      Can be smart, I.e. never reveals authentication data

Strong Authentication

- Something you know, something you have, something you are
- Uses more than one method to determine users identity
  - ATM card and PIN code
- Examples:
  - Private keys when stored on diskette are normally protected by a passphrase
  - SecureID authentication requires the current value and a PIN code for authentication

IATAC

Examples: S/Key one-time pw

- Server knows user's secret pass phrase
- Server generates challenge, which includes a seed value
- User enters pass phrase at the client, which is combined with the seed to generate the response
- The user’s pass phrase never crosses the network and is not stored in either the client or server machines

Examples: SecureID

- Token-based with PIN
- Token generates a new passcode every minute (custom times available)
- Authentication server uses the current time to determine acceptable entries
- Multiple Form Factors
  - Keyfob
  - Credit Card
  - Smartcard

IATAC

Authorization

- Once you have determined who a user is, you then must decide what access to grant the user
- Can be simple or complex depending upon the needs of the application
- Role Based Access Control (RBAC)
  - Assigns users into roles
    - This can be done statically or dynamically
  - Access granted to the role, not directly to the user
  - Some systems add additional intelligence such as membership restrictions (i.e. a receiving clerk can not also be a purchasing agent)
- Coalition problem largely unsolved
  - How to authorize access based upon memberships that change over time and may vary based upon the context of the request

IATAC

Auditing

- Keeping a record of system activities is an essential management activity
- Audit logs can be used for …
  - Debugging. When the system fails, the audit logs can show what the system was doing. This is invaluable when trying to determine a problems cause.
  - Accounting. Determine who used what resources when
  - Discover unusual activity
    - Unexpected activity may be a sign that your system is under attack
  - Damage limitation and recovery

IATAC

Syslog

- The syslog service provides configurable event logging
  - kernel, daemons, and applications send logs to syslogd
  - syslogd decides where to deliver messages depending on configuration
- /etc/syslog.conf used to configure. Entries have the following format
  - facility.level destination
    - facility = subsystem sending the message
    - level = severity level of the message
    - destination = file, device, computer, or user-name to send message to

IATAC

Syslog samples

- Sample `syslog.conf`

```
*.err                                   /dev/console
*.err;daemon;auth.notice;mail.crit      /var/adm/messages
lpr.debug                               /var/adm/lpd-errs
```

- Sample /var/adm/messages

```
Mar 21 10:36:04 host8 su: 'su root' failed for user1 on /dev/ttyp2
Mar 21 10:36:08 host8 su: 'su aaa' succeeded for user1 on /dev/ttyp2
Mar 24 15:01:44 host8 login: REPEATED LOGIN FAILURES ON console, user3
Mar 24 15:12:02 host8 shutdown: reboot by user1
```

Keystroke Logs

- Can be a very useful method for tracking system activity.
- C (csh), Korn (ksh) and Bourne-Again (bash) shells support a command history mechanism
- Retains footprints of what commands the user has executed
- Commands are usually saved in /.history
- Use the history command to list recently executed commands, e.g.,
  - # history
  - 1 cd /etc/X11/
  - 2 ls -l
  - 3 cp XF86Config XF86Config.orig

IATAC

What can you discover in logs?

- Accounting discrepancies (e.g., an 18-minute gap in a log file that normally has several entries per minute)
- Excessive logon attempts
- Unexplained, new user accounts
- Unexplained, new files or unfamiliar file names
- Unexplained modifications to file lengths or/or dates, especially in system executable files
- Unexplained attempts to write to system files or changes in system files

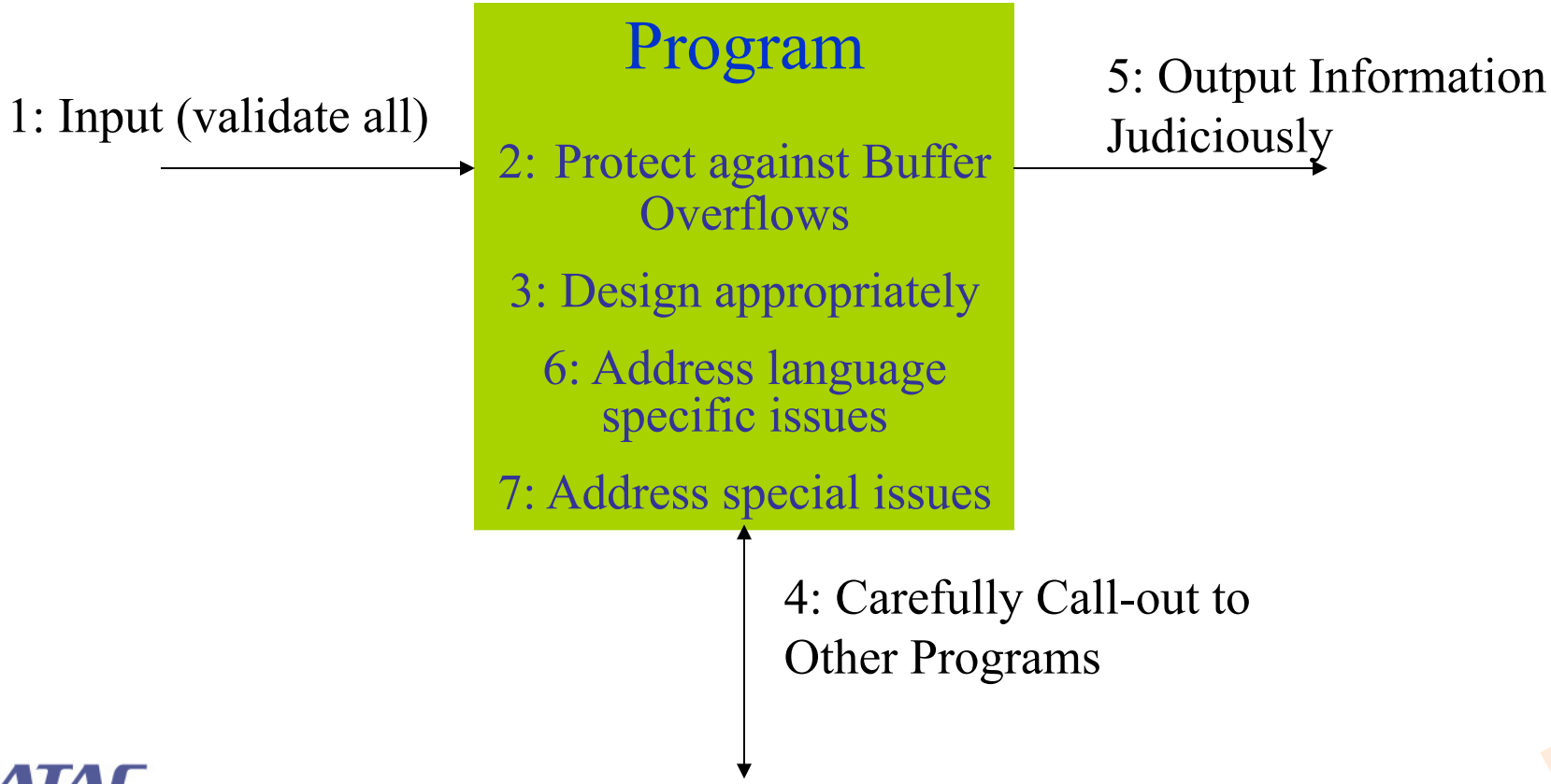
IATAC

Audit log protection

- A hacker's first target after gaining access to a system is an attempt to erase the evidence of the attack from system logs.
- Logs can be protected by
  - Having appropriate permissions set (though root usually can override this)
  - Writing logs to write-once media (WORM drives)
  - Sending audit data to an external protected server
  - Printing log entries as they occur

IATAC

Goal: Program in a Secure Context



IATAC

Programming Securely

- Validate all input
  - From untrusted sources
  - Strings (special characters) and numbers (min and max)
  - All other data types (email, filenames, command, environment variables, cookies, html form data, file content and file descriptors)
- Protect against Buffer (stack) Overflow
  - Avoid using risky functions (gets(), strcat(), etc)
- Design appropriately
  - Use safe defaults
  - Load configuration/initialization values safely
  - Fail safe, avoid race conditions, temp files in shared directories, etc.
  - Follow good security principles, prevent cross-site, etc.
- Carefully Call-out to other programs
  - Call only safe libraries, limit call parameters, encrypt sensitive info, etc

IATAC

Programming Securely (cont')

- Output Information Judiciously
  - Control data formatting
    - Printf ("whatever"); vs printf("%s", "whatever");
  - Handle disk full and/or unresponsive recipient
  - Minimize feedback
    - Log failuires
    - Avoid sending program version numbers
- Address Language Specific Issues
  - C/C++: type must be strict, turn on all warning, use gcc
  - Perl: enable –w (warn) –T (taint) options
  - Python: check use of exec, eval, execfile, etc.
  - Shell: never use them for setuid/setid
- Address Special Issues/Topics
  - Random number use, password in clear, user authentication, use of proprietary crypto modules

IATAC

Agenda

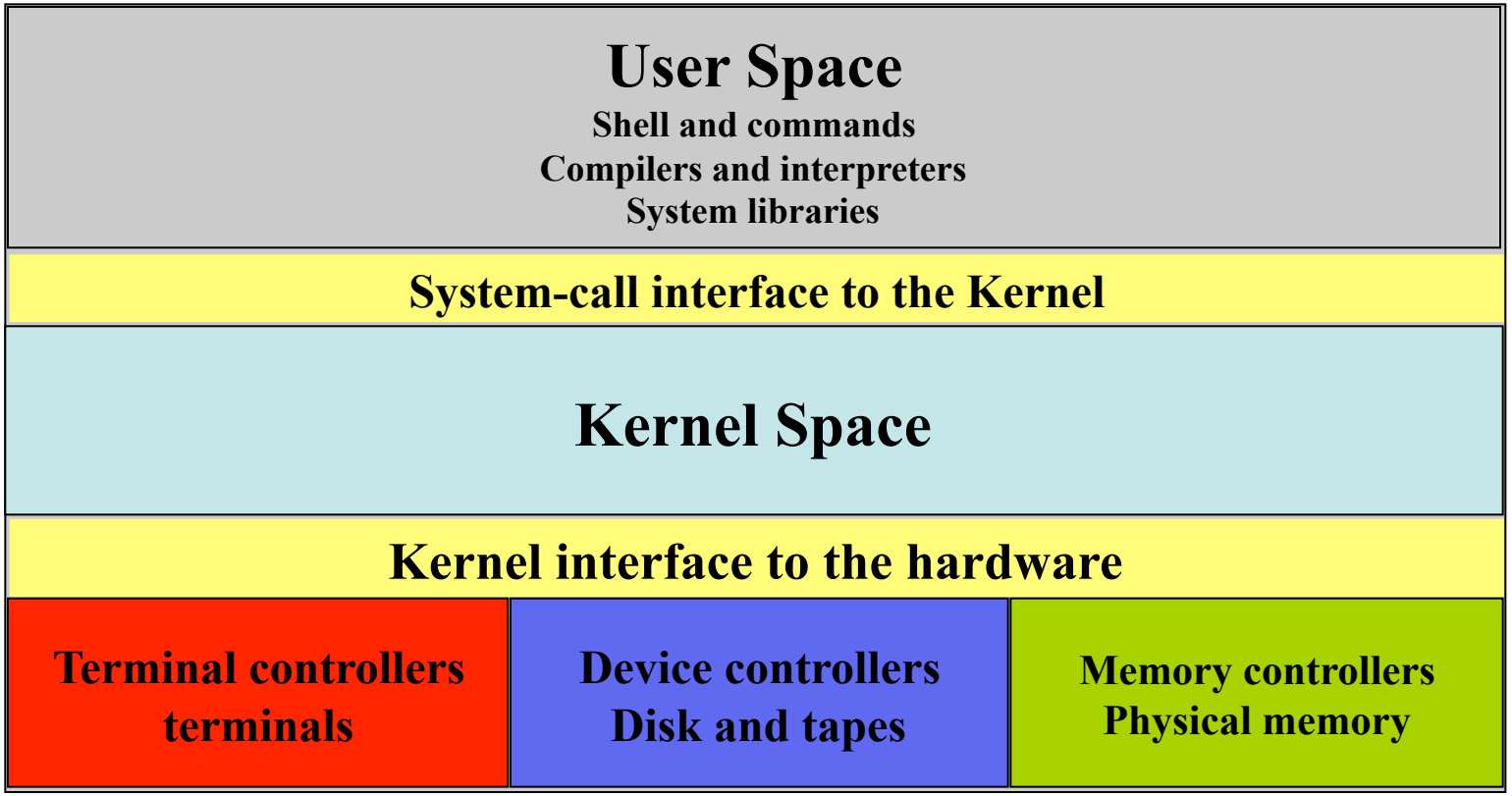
- Syllabus
- Grading
- Discussion of information security
- Overview of information security terms
  - The CIA triangle
  - AAA
- Overview of the Unix security model

UNIX System Structure

- Was limited by hardware functionality
- Consists of two separate parts
  - System programs
  - The Kernel
    - Everything below the system-call interface and above the physical hardware
    - Provides the file system, CPU scheduling, memory management, etc.

IATAC

UNIX System Structure



IATAC

Kernel and User Space

- The kernel implements basic system functions and runs with no security restrictions.
- Most programs execute in user space (on top of the kernel)
- We will concentrate on user space programs

IATAC

UNIX Security Model

- User authenticated on logon
  - User ID associated with process
  - Default Group ID associated with process
  - Default Process listed in ***passwd*** file
- Group defined in /etc/groups
  - Set of users listed with each group definition
  - Users can be member of multiple groups

User privileges

- During login, the user is mapped to a user ID (UID) and group ID (GID).
- Files (including programs) are owned by an individual UID, and belong to a single GID.
- Each file has three groups of privileges, the owner, the group, and the world
- Privileges are read, write, and execute
- UID 0 has special privileges
  - Can overrule most security checks
  - Used to administer the system
  - Referred to as root

IATAC

Program privileges

- Programs normally execute at the privilege level of the user that started them
- It is possible to mark a program so that the program executes with the privilege level of the owner of the file
  - Set user ID (SUID)
  - Also possible to execute with the group assigned to the file. (SGID)
  - This is a major security concern

Interesting Process Attributes

- Real User ID (RUID) and real group ID (RGID)
- Effective User ID (EUID) and effective group ID (EGID)
- Saved UID and GID; used to support switching permissions on and off
- Supplemental Groups; a list of groups (GIDs) in which this user process has membership
- File system root; the location in the file system that the process thinks is the root of the file system

Program Startup

- Depends on type of program
- Executable programs started with execve call
  - Replaces current executing process with image from file
- Scripts handled differently
  - 1st line of script read to determine which interpreter should be used
  - Shell then executes interpreter and passes the name of the script file
  - Caused race condition in earlier versions of Unix

Processes

- For our purposes, programs and processes are identical
- Programs can create copies of themselves (using fork calls or variants)
  - These copies inherit the privileges of their parent process
- Threads
  - Sometimes referred to as lightweight processes
  - Security issues similar to processes

File System Objects (FSO)

- In unix almost everything is a file
  - Files
  - Directories
  - Symbolic Links
  - Named Pipes
  - Sockets
  - Character special device files
  - Block special device files

FSO permissions

- Owning UID and GID
  - Only the owner or root can change the ownership / group membership of a file
- Permission bits
  - Read, write, execute for each owner, group, and other
  - Files: as you would expect
  - Directories:
    - Read = ability to display
    - Write = add, remove, or rename files in dir
    - Execute = ability to enter directory
    - Sticky bit = limits write to add only

IATAC

Interpreting Permissions

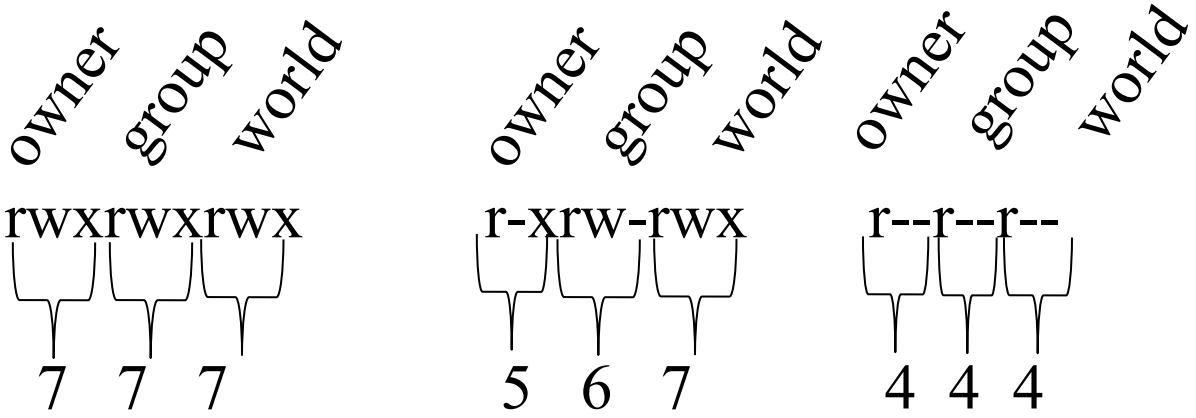
Symbolic notation		Octal notation
r	Read	4
w	Write	2
x	Execute	1
s	Set user id (suid)	4000
s	Set group id (sgid)	2000
t	Set save text (sticky bit)	1000

Interpreting Permissions

ls -l myfile

-rwxr-xr-- 1 jsmith user 6002 Sep 15 14:02 myfile

r = 4, w = 2, x = 1, - = 0 rwx = 4 +2+1 = 7



Interpreting Permissions

ls -l yourfile (suid/sgid)

-rwsr-sr-t 1 rjones group1 6002 Sep 15 14:02 yourfile

rws = file will execute with all the privileges of user rjones

r-s = file will execute with all the privileges of group1

r-t = indicates the sticky bit is set for file yourfile

`suid` and **`sgid`** must be closely managed: any user who is running a file with suid/sgid set acquires the privileges of the file owner (dangerous if the file is owned by root).

IATAC

Interpreting Permissions: Dirs

ls -l yourdirectory (execute/sgid/sticky)

drwxrwsrwt 1 rjones group3 6002 Sep 15 16:22 yourdirectory

d = this is a directory

rwx = owner rjones has full access to the directory -- has read and write privileges to the directory itself, and can access files within it

rws = members of group3 have full access to the directory; new files always belong to group group3 regardless of owner's default group

rwt = all users have full access to the directory, but can only delete files that they own

Posix Style ACLs add richer set of permissions

POSIX ACL Entry Name	Meaning	Short Form	Long Form
ACL_USER_OBJ	The rights of the owner	u::	user::
ACL_USER	The rights of some specific user, other than the owner	u:USERNAME:	user:USERNAME:
ACL_GROUP_OBJ	The rights of the group that owns the file	g::	group::
ACL_GROUP	The rights of some other group that doesn't own the file	g:GROUPNAME:	group::GROUPNAME:
ACL_OTHER	The rights of anyone not otherwise covered	o::	other::
ACL_MASK	The maximum possible rights for everyone, except for the owner and OTHER	m::	mask:GROUPNAME:

* Not ratified but is implemented on some modern systems

When are access control attributes enforced?

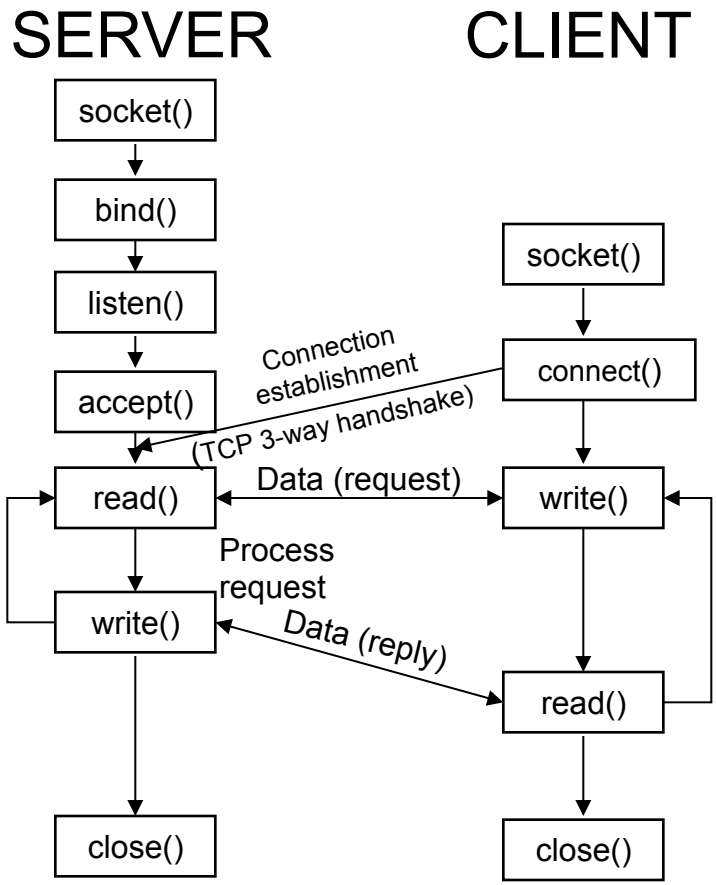
- When file is opened but not during reads / writes
- Calls that check include:
  - open – open file
  - create – create new file
  - link – create a file that points to another file
  - unlink – remove the link
  - rename – rename the file
  - mknod – make a special file (such as a named pipe)
  - symlink – create a symbolic link
  - socket – create an endpoint for communication

Interprocess Communication (IPC)

- Method to communicate between different processes
- Three types supported
  - Message Queues
  - Semaphores
  - Shared memory
- Permissions implemented in a similar fashion to FSOs

Sockets

- Primarily used to communicate across a network



Domain sockets

- Connect to sockets on the same machine
- Alternative to named pipes but has security advantages
- Each connection request results in a new communication channel
- Must use socket calls to access

Quotas and Limits

- Useful for preventing denial of service attacks
- File system quotas
  - Can limit the total blocks and the total number of files
  - Per user and/or per group
  - Soft limit (can be temporarily exceeded)
  - Hard limit (actual maximum)
- Process resource limits
  - File sizes
  - Number of child processes
  - Number of open files
  - Etc.

IATAC

Popular security extensions

- Pluggable Authentication Modules (PAM)
  - Permits run-time configuration of authentication methods
    - Passwords
    - Smartcards
  - Will be covered in detail later
- tcpwrappers
  - Used to control access to sockets
  - Configured in /etc/hosts.allow, /etc/host.deny

Next Thursday's Class

How hackers discover and take advantage of security flaws

IATAC

Questions?