Review (II)

Week 10, Thursday

Some Words on Review

• It seems the more you learn, the more you’ll forget
  - We do have learnt a lot
• Re-thought the question people brought up last class on reviewing materials after midterm
  (Thanks for the question)
• Yes, at this stage it makes better sense to review than to learn something more
• So, let’s do it
  - I’ll lead you to go through lecture notes

What We’ve Covered After Midterm

• Assurance
• Malicious logic
• Intrusion detection
• Network security

Assurance

(was on) Week 7, Thursday

Outline

• Introduction
• TCSEC
• ITSEC
• CC

Absolute Security?

• There is no absolute security
  - Although vendors often claim their products are secure
• You address one attack, then you find you are facing another
• Infeasible to guarantee that a system is secure
  - Or remain secure over time
• So, how can we trust any single system?
Trust is a Belief

- Intuitively, one may trust a system to do what it should do in protecting itself
- Academically, trust is a measure of trustworthiness, relying on the evidence provided.
  - Where an entity is trustworthy if there is sufficient credible evidence leading one to believe that the system will meet a set of given requirements
  - The more evidence, the higher confidence (or belief) on the trustworthiness

This Confidence is Called Assurance

- Security assurance is confidence that an entity meets its security requirements
  - Based on specific evidence provided by applying assurance techniques
- Note there is a different term: information assurance
  - Which refers to the ability to access information in a high quality and security

Policy, Mechanism, & Assurance

<table>
<thead>
<tr>
<th>Policy</th>
<th>Statement of requirements that explicitly defines the security expectations of the mechanisms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assurance</td>
<td>Provides justification that the mechanism meets policy through assurance evidence and approvals based on evidence</td>
</tr>
<tr>
<td>Mechanisms</td>
<td>Executable entities that are designed and implemented to meet the requirements of the policy</td>
</tr>
</tbody>
</table>

Assurance Techniques

- The goal is to build trustworthy systems
- Examples:
  - A development methodology
  - Formal methods for design analysis
  - Testing
- Types: informal, formal, semiformal
  - Informal: natural languages for spec and justification
  - Semiformal: impose some rigor
  - Formal: mathematical proofs, for instance

Trusted System

- A trust system is a system that has been shown to meet well-defined requirements under an evaluation by a credible body of experts
  - who are certified to assign trust ratings to evaluated products and systems
- Does not guarantee perfect security, but does provide basis for confidence

Formal Evaluation Technology

- to provide measurements of trust of a particular system based on specific security (function) requirements and evidence of assurance of the system
  - Three elements here
Terms

- **Target of evaluation (TOE):** a product or system, and its associated administrator and user documentation, that is the subject of an evaluation
- **TOE Security Policy (TSP):** a set of rules that regulate how assets are managed, protected, and distributed within a system
- **TOE Security Functions (TSF):** a set of all hardware, software, and firmware of the system that must be relied on for the correct enforcement of the TSP

System Evaluation Standards

- **TCSEC:** Trusted Computer System Evaluation Criteria
- **ITSEC:** Information Technology Security Evaluation Criteria
- **CC:** Common Criteria
  - Replaces the previous two
- Military and government started this
  - So that they can choose “assured” commercial software

Evaluation Considerations

- Security evaluation is an investment
  - Considering tradeoff between security and cost
  - While not being able to guarantee the lack of vulnerabilities
- Can be used to compare products
- Should be done independently done by a credible third party

TCSEC (1983–1999)

- The first widely used formal evaluation methodology
- Known as *Orange Book*
- Presented a set of criteria for evaluating the security of commercial computer products

Ratings, or Levels of Trust

- Six different evaluation classes: C1, C2, B1, B2, B3, and A1
  - A1 is the highest
- They are in three divisions: C, B, A
  - There is also the *D division* for those that fail to meet all security requirements
  - Also called *D class*

Theory Behind TCSEC

- **Reference monitor**
  - A resource validation mechanism (RVM) that requires a resource monitor to validate all accesses to ensure security
  - Generalized as **TCB** (Trusted Computing Base)
  - always invoked, tamperproof, and simple
- **Bell-LaPadula model**
  - not fully address the integrity and no consideration of availability
**TCSEC Got Expanded**

- A “Rainbow Series” by NCSC (national computer security center)
- Cover security in specific context
  - Networks, databases, audit systems

**Functional Requirements**

- Discretionary access control (DAC) requirements
  - Individuals should be able to do DAC
- Object reuse requirements
  - When an object is reused, the previous access rights must be revoked
- Mandatory access control
  - Enforce the simple security condition and *-property from Bell-Lapadula model
- Label requirements
  - Both subject and object must be properly labeled

**Assurance Requirements**

- Configuration management (B2+)
  - TCB generation, etc.
- Trusted distribution (A1)
  - Mapping between a master version and installation versions, etc.
- TCSEC system architecture (C1-B3)
  - Modularity, simplicity, etc.
- Design specification and verification
  - Each class is different
- Testing
- Product documentation

**TCSEC Evaluation Process**

- On requirements, C1 ≤ C2 ≤ B1 ≤ B2 ≤ B3 ≤ A1
  - Below each class only lists extra requirements compared to the next lower class
- C1: discretionary protection
  - I&A, DAC (for functional requirements)
  - Testing, doc (for assurance requirements)
- C2: controlled access protection
  - Object reuse, auditing
  - More stringent testing than C1
- B1: Labeled security protection
  - MAC for a specified set of objects
  - More stringent testing than C2

**(cont’d)**

- Identification and authentication (I&A) requirements
  - To prevent malicious users from using a system
- Trusted path requirements
  - A trusted communication path must be guaranteed between the user and the TCB
- Audit requirements
  - Appropriate auditing must be enforced

**(cont’d)**

- B2: structured protection
  - MAC for all, trusted path for login, least privilege
  - Covert channel analysis, more stringent doc, a proved formal security policy model
- B3: security domains
  - Full reference validation mechanism, more on trusted path
  - More stringent testing
- A1: verified protection
  - [B3’s]
  - Formal methods for assurance is required
ITSEC (1991-2001)

- An European standard
- Six levels of trust: E1, E2, E3, E4, E5, E6
  - E0 is for “no security” products
  - Certified product: a product evaluated under ITSEC
- Requires product vendors to define the security functional criteria in a security target

(TCSEC-ITSEC) Requirements

- Tamperproof reference validation mechanisms
- Process isolation
- Principle of least privilege
- Well-defined user interfaces
- System integrity

(ITSEC-TCSEC) Requirements

- Security measures for developer environment
- All levels of code representation of a product is needed
  - From various source level to object code
  - Every level must be verified
- Software distribution process
- Vulnerability assessment

ITSEC Evaluation Levels

- E1: a security target is needed (in order to evaluate a product) and an informal product description
- E2: Also detailed design description, configuration and distribution process, and testing evidence
- E3: Also the correspondence between source code and security requirements
- E4: a formal model needed
- E5: also the correspondence between the detailed design and source code
- E6: extensive use of formal methods

ITSEC Evaluation Process

- Let’s omit the bureaucracy for this time!

More Criteria

- “pass-or-fail” commercial certification (late 80’s)
- Federal Criteria (1992)
- FIPS 140 (1994-present)

- They are all not satisfactory to some degree . . . .
Common Criteria

- Introduction
  - History, components, terms, etc.
- Evaluation methodology
- Evaluation requirements
- Evaluation assurance levels
- Evaluation process

Introduction

- Draws from the strength of TCSEC, ITSEC, etc.
  - An International effort
  - Literally called Common Criteria Recognition Arrangement (CCRA), or Mutual Recognition Arrangement (MRA)
  - Also the standard 15408 of ISO (International Standards Organization)
- Became de facto security evaluation standard in US in 1998
  - TCSEC finally retired in 2000

Components

- CC documents
  - Overview of the methodology, functional requirements, assurance requirements, and Evaluation Assurance Levels (EALs)
- CC evaluation methodology (CEM)
  - Detailed evaluation at each EAL
  - The lower four are completed
  - The higher three are not done yet
- A national scheme
  - Country-specific part on evaluation methodology
  - Selection of evaluators, certification process, legal issues, etc.
  - US uses CCEVS (CC Evaluation and Validation Scheme)

Evaluation Methodology

- Two kinds of evaluations
  - Evaluation of protection profiles (PP)
  - Evaluation against security targets (ST)
- Every system is evaluated to be at one of the seven predefined EALs
  - or at a user-defined EAL

Protection Profile (PP)

- CC protection file: an implementation-independent set of security requirements for a category of products or systems that meet specific consumer needs

PP Describes a Family of Products

- Introduction
  - The id of a PP
  - The abstract overview of the PP
- Product family description
- Security environment
  - Assumptions, threats, security policies
- Security objectives

- Security requirements
  - Functional requirements
    - Drawn from CC
  - Assurance requirements
    - Based on an EAL
    - Can be augmented
- Rationale
  - Security objective rationale
  - Security requirements rationale
Security Target

- **Security target**: a set of security requirements and specifications to be used as the basis for evaluation of an identified product or system
- Evaluation against a ST has two parts:
  - An ST must be developed and evaluated first
  - Then evaluate a product or system against the ST

ST Addresses a Specific Product

- Introduction
  - The id of an ST
  - The abstract overview of the ST
  - CC conformance claim
- Product description
  - Security environment
    - Assumptions, threats, security policies
  - Security objectives
- Security requirements
  - Functional requirements (from CC)
  - Assurance requirements
    - Based on an EAL but can be augmented
- Product summary specification

CC Requirements

- The heart of CC
- CC defines both functional requirements and assurance requirements
  - Then decide EAL of a product
- Requirements are divided into classes
  - Classes are broken into families

Functional Requirements

- 11 classes
  - FAU: security audit
    - Six families on audit automatic response, audit data generation, audit analysis, audit review, audit event selection, & audit event storage
  - FCO: communication
    - Two families on nonrepudiation of origin and nonrepudiation of receipt
- FCS: cryptographic support
  - Two families on key management and cryptographic operation
- FDP: user data protection
  - 13 families on access control and info flow policies and functions, data authentication, rollback, data integrity, etc.

Assurance Requirements

- 10 classes
  - One on PP, one on ST, one on the maintenance of assurance, and seven directly on assurance for a product
  - APE: PP evaluation
  - ASE: ST evaluation
  - AMA: maintenance of assurance
- ACM: configuration management (CM)
- ADO: delivery & op
- ADV: development
- AGD: guidance doc
- ALC: life cycle
- ATE: tests
- AVA: vulnerability assessment

(cont’d)

- FIA: identification and authentication
- FMT: security management
- FPR: privacy
- FPT: protection of security functions
- FRU: Resource utilization
- FTA: TOE access
- FTP: Trusted Path
EAL: Evaluation Assurance Levels

• EAL 1: functionally tested
• EAL 2: Structurally tested
• EAL 3: Methodically tested and checked
• EAL 4: Methodically designed, tested and reviewed
• EAL 5: semi-formally designed and tested
• EAL 6: semi-formally verified, designed and tested
• EAL 7: formally verified, designed and tested

Evaluation Process in US

• Controlled by the CC CEM and NIST
• Performed by NIST-accredited lab
• Different things can be evaluated
  – A product
  – A product’s security target
  – A product family’s protection profile

Future of Common Criteria

• Continue to evolve
  – Term interpretations, new assurance class and family, . . . .
  – EAL5’s evaluation methodology to be added to CEM

Malicious Logic

(was on) Week 8, Tuesday

Introduction

• Malicious logic: a set of instructions that cause a site’s security policy to be violated
• They make most traditional access control useless
  – Typically by assuming an authorized user’s identity
• We will talk Trojan horses, viruses, worms, etc.

Trojan Horse

• An example:
  A Unix script called “ls” in directory /usr, created by EVE:
  ```
  cp /bin/sh /tmp/.xxsh
  chmod u=r,s,a+x /tmp/.xxsh
  rm -/ls
  ls $*
  ```
  If user VICTOR is cheated to run the above “ls” instead of the original “ls” he will create a /tmp/.xxsh file. EVE then can run /tmp/.xxsh, a shell program that allows EVE to enjoy a shell running environment for VICTOR!

• This “ls” by EVE does do what the original “ls” does
  – But it also does something else: creating a shell program that can give EVE the access rights of VICTOR!
(cont’d)

• **Trojan Horse**: a program with an overt effect and a covert effect
  – Overt effect: documented or known effect
  – Covert effect: undocumented or unknown effect

• **Propagating Trojan horse**: a Trojan horse that creates a copy of itself
  – Also called **replicating Trojan horse**

**Computer Viruses**

• **Computer virus**: a program that inserts itself into one or more programs and then performs some action
  – A Trojan horse only propagates itself
  – A virus infects others

  • Note: some experts regard virus as a type of Trojan horse, where the infected program gives the overt effect and the virus code leads to covert effect.

**Virus Mechanism**

```plaintext
beginvirus:
  if spread-condition then begin
    for some set of target files do begin
      if target is not infected then begin
        1. determine where to place virus instructions
        2. copy virus instructions from beginvirus to endvirus into target
        3. alter target to execute added instructions
      end
    end
  end
perform some action(s)
go to beginning of infected program
endvirus
```

**There are many many viruses . . .**

• And there will be many many many of them!

  • Types:
    – Boot sector infectors
    – Executable infectors
    – Multipartite viruses
    – TSR viruses
    – Stealth viruses
    – Encrypted viruses
    – Polymorphic viruses
    – Macro viruses

**Boot Sector Infectors**

• Boot sector: the part of a disk used to bootstrap the system
  – Or mount a disk
• Boot sector is executed when the system “sees” the disk for the first time
• **Boot sector infectors**: a virus that inserts itself into the boot sector of a disk
  – Loaded into memory when invoked

**Executable Infectors**

• **Executable infectors**: a virus that infects executable programs (or applications)
  – Also called COM or EXE viruses
  – The virus prepends or appends itself
Multipartite Viruses

- **Multipartite virus**: a virus that can infect either boot sectors or applications
  - Often has two parts, one for each type
    - Depending on what to infect

TSR Viruses

- **Terminate and stay resident (TSR) virus**: a virus that stays active (resident) in memory after the application (or bootstrapping, or disk mounting) has terminated
  - Can be boot sector infectors or executable infectors

Stealth Viruses

- **Stealth virus**: a virus that conceal the infection of files.
- For example:
  - If you try to read an infected program to check the integrity of the program, the virus will return you the original program
  - But, the virus will still get executed when the program is executed

EncryptedViruses

- **Encrypted virus**: a virus that uses enciphered code
  - In order to prevent virus detection

Macro Viruses

- **Macro virus**: a virus that composed of a sequence of instructions that is interpreted, rather than executed directly
- Can infect either executable applications, or data files
  - The latter called **data virus**

Polymorphic Viruses

- **Polymorphic virus**: a virus that changes its form each time it inserts itself into another program
  - For example, using different machine instructions with the same effects
  - Can be automated
  - An encrypted virus can evolve into a polymorphic virus by varying the decipherment routine
### Computer Worms
- **Computer worm**: a program that copies from one computer to another
  - A variant of virus
- Many worms have occurred
  - Father Christmas worm, CodeRed worm, Slammer worm, etc.
  - The Slammer worm spread worldwide in ~5 minutes

### Rabbits and Bacteria
- **Rabbit or bacterium**: a program that absorbs all or some class of resource
  - Creating denial-of-service attack
  - For example
    ```c
    while (true)
    do
        mkdir x
        chmod x
    done
    ```

### Logic Bomb
- **Logic bomb**: a program that performs a malicious action when some external event occurs

### Theory of Malicious Logic
- **Theorem 22-1**: It is undecidable whether an arbitrary program contains a computer virus
  - Proof: omitted
- **Theorem 22-1**: It is undecidable whether an arbitrary program contains a malicious logic
  - Proof: omitted

### Defenses
- Malicious logic acting as both data & instructions
- Malicious logic assuming the identity of a user
- Malicious logic crossing protection domain boundaries by sharing
- Malicious logic altering files
- Malicious logic performing actions beyond specification
- Malicious logic altering statistical characteristics

### Malicious Logic Acting as Both Data and Instructions
- When malicious code is inserted, it is data
- When malicious code is executed, it is instructions
- Solution
  - Treat all programs as “data”
  - Must be certified to become “executables”
  - Certification process must be secure
  - Modifying an “executable” will make it become “data” again
Malicious Logic Assuming the Identity of a User

- Limit the distance a virus can spread
  - Flow distance metric $fd(x)$: info $x$ can only travel for distance $fd(x)$.
- Reduce the rights
  - Principle of least privilege
  - Authorization
  - “watchdog” or “guardian” to check each access
- Use sandbox or virtual machines
  - To restrict process rights in a confined environment

Malicious Logic Crossing Protection Domain Boundaries by Sharing

- Inhibit users from different domains from sharing programs or data
  - But then the sharing is also disabled
  - Thus many service will be disabled as well
- Hard!

Malicious Logic Altering Files

- Use signature block to sign a program
  - We have learned this!
- Must ensure no malicious code before the signing operation

Malicious Logic Performing Actions Beyond Specification

- Treat infection and execution of viruses as **error**!
- Example 1:
  - Break a program into sequences of nonbranching instructions
  - Sign each of them (and store the signatures)
- Example 2: N-version programming
  - N versions of a program is made
  - Run concurrently
  - Hopefully the majority of them will be correct

(cont’d)

- **Proof-carrying code (PCC):**
  - Code user specifies a safety requirement
  - Code author generates a proof that the code meets such requirement
  - And integrate the proof with the code!
  - The code user then verifies the proof

Malicious Logic Altering Statistical Characteristics

- Malicious code may change the specific statistical characteristics that a program originally had
  - So the detection of such changes can help detect the attack
- Example:
  - The printing command “lpr” is used 1000 times by Joe in just one day
Intrusion Detection

Motivation

- We have learned quite a lot of security, can’t computer intrusions be simply blocked?
  - Unfortunately, the answer is NO!
- Intrusions could still happen no matter how desperate you are
  - There is no absolute security
  - Human being always make mistakes
  - Nothing is perfect
- So, let’s be able to detect an intrusion once it happens or happened

Principles

- The actions of users and processes of a computer system that is not under attack generally:
  - Conform to a statistical pattern
  - Do not include malicious sequence of commands to subvert the system
  - Conform to specification on privileges etc. (namely, security policies)
- A system under attack will not meet one of the three properties

Denning’s Hypothesis

- **Denning’s hypothesis**: any intrusion, when exploiting vulnerabilities of a system, will have to require an *abnormal* use of system facilities
- So, security violations can be detected by looking at abnormalities

Denning’s Intrusion Detection Model

- Very general
- Include abnormalities (recall the “Principles” slide here):
  - Deviation from usual actions (which corresponds to anomaly detection)
  - Execution of actions that lead to break-ins (which corresponds to misuse detection)
  - Actions inconsistent with the specifications of privileged programs (which corresponds to specification-based detection)

Intrusion Detection System (IDS)

- Goals are:
  - Detect a wide variety of intrusions
  - Detect intrusions in a timely fashion
  - Present the analysis in a simple, easy-to-understand format
  - Be accurate

Denning’s Intrusion Detection Model
False Positives & False Negatives

- **False positive**: reports an attack, but no attack in fact
- **False negative**: does not report an attack, but there is an attack

Models of IDS

- Anomaly models
  - *The art of looking for unusual states*
- Misuse models
  - *The art of looking for states known to be bad*
- Specification-based models
  - *The art of looking for states known not to be good*

Anomaly Modeling

- Assumption: unexpected = bad
- **Anomaly detection**: compares the statistical behavior of a system with what’s expected, and reports if not matching
- Three statistical models
  - Threshold metric
  - Statistical moments
  - Markov model

Threshold Metric

- An event can only happen $x$ times, where $m \leq x \leq n$
  - Sometimes hard to determine $m$ and $n$
- Example:
  - At most 3 times of login are allowed
  - At most 200 http connections at a single time
  - Other examples?

Statistical Moments

- Compare values with the mean and standard deviation (the first two moments), or even higher moments
  - An “outlier” is an anomaly
- Flexible, but also complex
- Example: IDES (Intrusion Detection Expert System)
  - Every subject has a statistical profile that is updated daily, favoring most recent behavior

Markov Model

- Use History
- From every certain state, the system has high probabilities to enter normal states, while low probabilities to abnormal states
- Need good training data
- Example:
  - Event prediction using TIM system
    - If `c {} d(0.5) c(0.5)` then `c` is an anomaly!
    - system call trace analysis
      - If (read() or write()) always follows open() during training, then many open() calls w/o read() or write() calls is an anomaly!
Misuse Modeling

- **Misuse detection**: determines whether a sequence of instructions being executed *is known* to violate the site security policy
  - If so, it reports a potential intrusion
- The system uses a *rule set* for misuse detection
  - Thus can probably only detect *already known* intrusions
- Several misuse-based IDS:
  - IDIOT, STAT, NFR

IDIOT

- By Kumar and Spafford
- Stands for “Intrusion Detection in Our Time”
- Tries to detect unknown attacks as well as known attacks
  - Use Colored Petri nets, on which every intrusion maps to a compromised state
  - Info on new intrusions, also called *intrusion signature* can be dynamically added

STAT

- STAT views a computer always in a particular state
  - Commands move the computer from state to state
  - The goal of STAT is to find “bad” state transition
    - All compromises involves switching to a state that some “illegal” right is acquired

NFR (Network Flight Recorder)

- Allows new rules to be added
- Three components
  - **Packet sucker**: reads packets off the network
  - **Decision engine**: use a filter to extract info
  - **Backend**: write the data from filter to disk
    - A sysadmin can then query the disk
    - Also called *query backend*
  - The filter is pluggable and replaceable

Specification Modeling

- **Specification-based detection**: determines whether or not a sequence of instructions violates a specification of how the system *should* execute
  - Potentially this can help detect unknown attacks
    - Since it talks what *should* happen
  - Still at its infancy

IDS Architecture

- Three components:
  - **Agent**: collect data
  - **Director**: analyze data
  - **Notifier**: inform the intrusion
Host-Based Agents

- Often use logs of system and applications
- Examples:
  - Security-related logs
  - Account logs
  - Others?

Network-Based Agents

- Often monitors and collect info of network traffic
- Thus it may detect network attacks
- Features
  - Sniffing is often used
  - Can monitor all traffic passing the agent
  - So could monitor traffic for more than one hosts
  - Can monitor traffic contents as well
- Question: what if the traffic is encrypted?

Director

- A director works in two phases
  - Filter data gathered and reported by agents
  - Analyze the data
- A director often relies on multiple analysis engines
- A director typically runs on a separate system
  - Separation may mean isolation from attacks
  - Preventing attackers from knowing too much info

Notifier

- Accepts info from director and notify
- In addition, can respond to attack!
  - Also called incident response
- Example:
  - GrIDS (Graphical IDS) uses GUI to alert intrusions
  - IDIP (Intrusion detection and isolation protocol) allows several IDSs to notify each other

Organization of IDS’s

- We look at three paradigms
  - Monitoring network traffic for intrusions: NSM
  - Combining host and network monitoring: DIDS
  - Autonomous Agents: AAFID

Network Security Monitor (NSM)

- NSM develops a profile of expected usage of network
- And compares current usage with the profile
- Also includes signatures for known attacks
- Monitors a whole local area network
- NSM has been quite widely used
  - Can deal with enciphered traffic since NSM does not do content monitoring
NSM is connection-based
- Each connection has an ID
- # of packets sent etc. for each connection is recorded
- And compared with what’s expected

Too much data
- So build a hierarchy
  - E.g. any traffic between Alice and Bob
  - Only go down the hierarchy if something abnormal
  - E.g. if A-B traffic looks abnormal, then look at FTP, HTTP, SMTP, TELNET, etc traffic separately

Autonomous Agents (AAFID)
- The director in IDS architecture is a single point of failure
  - So, why not partition an IDS into multiple correlated components
- AAFID does this
  - Then encourage the cooperation of agents
  - Disadvantage:

Containment Phase
- Limit the access of an attacker
  - Only a reduced set of resources can be accessed
- Passive traffic monitoring does not contain any attack
- Step-by-step access constraining is complicated
  - Especially when not knowing what attackers may do
- Honeypots (decoy servers)

Distributed Intrusion Detection System (DIDS)
- Combines NSM and host-based intrusion detection
  - Agents are on both the monitored hosts and monitored network
- Uses a centralized analysis engine (DIDS director)
  - A rule-based expert system
- GrIDS is an extension of DIDS
  - GrIDS can also contain data from network infrastructure (such as DNS)

Intrusion Response
- Incident prevention
  - Stop intrusions before they happen
- Intrusion handling (6 phases)
  - Preparation
  - Identification
  - Containment
  - Eradication
  - Recovery
  - Follow-up

Eradication Phase
- Stop the attack!
- Examples:
  - place wrappers around suspected targets
  - Boundary controller in IDIP
    - Block connections from entering a perimeter
Follow-up Phase

• Often legal actions, either criminal or civil
• Technically, people focus on trace back of network attackers

Network Security Overview

(was on) Week 10, Tuesday

Network Security

• Still, the security policy dictates the security mechanisms
• We analyze a particular case where network security must be applied
  – So you can get a feeling on network security
  – And do well in the final :)
• The network security class this coming fall will provide a comprehensive coverage

Case Intro: The Dribble Inc.

• The Dribble Corporation wants to provide certain services:
  – Web access (ads, online sale, etc.)
  – Email
  – And connection to the Internet from inside
• While still having a solid security

The Drib’s Security Goal

• Confidentiality
  – Sensitive company data must be kept secret
    • only accessible by those who need to know
    • Customer info (such as credit card numbers) must be kept secret
      • only available to those who fill the order
    – Releasing sensitive data requires the consent of company officials and lawyers
• Integrity
  – ???
• Availability
  – 99% time web service is up

Security Policy

<table>
<thead>
<tr>
<th></th>
<th>Outsiders</th>
<th>Developers</th>
<th>Executives</th>
<th>Employees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public Data</td>
<td>read</td>
<td>read</td>
<td>read</td>
<td>read</td>
</tr>
<tr>
<td>Data for existing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>products</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>data for future</td>
<td>read, write</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>products</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Corporate data</td>
<td></td>
<td></td>
<td>read, write</td>
<td></td>
</tr>
<tr>
<td>Customer data</td>
<td>write</td>
<td></td>
<td>read</td>
<td>read, write</td>
</tr>
</tbody>
</table>
**Terminologies**

- **DMZ** (demilitarized zone): a portion of a network that separates a purely internal network from an external network
- **Firewall**: a host that mediates access to a network, allowing and disallowing certain types of access on the basis of a configured security policy

**Firewall Types**

- **Filtering firewall**: performs access control on the basis of attributes of the packet headers, such as destination addresses, source addresses, and options
  - Often filter on packets passing by
- **Proxy firewall**: uses proxies to perform access control. A proxy firewall can base access control on the contents of packets and messages, as well as on attributes of the packet headers
  - Often intercept packets first for analysis

**The Drib’s Firewalls**

- The public cannot communicate directly with internal network
- Nor the internal network can communicate directly with the public
- DMZ is the “bottleneck” for both
- Firewalls help conceal the addresses of the internal network

**Outer Firewall**

- Properties:
  - Restrict public access to the Drib’s network
    - Similar to the read access of Bell-Lapadula model
  - Restrict the Drib’s access to the Internet
    - Similar to the write access of Bell-Lapadula model
  - While allowing certain sanitized info exchanges
    - Based on source or destination address, port number, etc.
Inner Firewall

• Goal: protect the internal network where sensitive data resides
• Implementation: block all traffic except for that specifically authorized to enter
  – DMZ web server to the internal web server
  – DMZ email server to the internal mail server

DMZ

• DMZ mail server
• DMZ WWW server
• DMZ DNS server
• DMZ log server

(End of review)

General Class Issues

On Homework 5

• Grades of HW 5 is lost
  – Jin’s laptop had some problem
• So, please each email her your HW-5 grade
  – Unless you don’t have a grade for HW-5
  – Jinzhang@cs.uoregon.edu
• Be honest

Class Project (grads only)

• Review your progress!
• The final project report is due on 6/13, midnight, PDT
  – Of this year!
  – Email me in PDF format
  • Imagine you are publishing a paper
  • 5 pages on letter-sized paper, 1” margin on all sides, single space, 11 Times New Roman font
  • At most 5 graphs, each smaller than 3.5”x3.5”
  – Extension request will be considered
Project Report Requirements

• A final report MUST contain:
  – Title
  – Author name
  – Introduction
    • Problem statement
    • Motivation
    • Prior work (if none or little, then say so)
    • Overview of your approach
  – Design
  – Implementation
  – Evaluation
    • Analysis
    • Measurement or simulation results
  – Conclusions
  – Acknowledgment
  – References

How Would I Grade Projects?

• I’ll consider the following factors:
  – Is the idea original?
  – Are your hands dirty?
  – Are in-class presentation and progress report well done?
  – A good project web page also helps
  – A well-presented final report
  – Positive attitude: proactive, constant interaction with me, lots of discussion

Final Exam (undergrads only)

• Time: 6/10, Tuesday, 8:00-10:00 a.m.
• Place: here! 102 Deady Hall
• What may be tested: All materials in this course
• Format: similar to the midterm
• Difficulty level: won’t be harder than the midterm
  – Will have problems on practical issues
  – You may be asked to solve “real” problems
• Our common wish:
  – everybody excels in the final!
  – It is possible, isn’t it?

Course Evaluation

• The instructor likes to do well in his final too
  – This is the FIRST COMPUTER SECURITY class in UO
  – Feel free to say good words :)
  – And write down your comments
  – Only signed written comments will be counted
  – Use pencil for the bubble sheet
• Feel free to contact him even after this class is over

See You This Fall on Network Security Class

* Week 1: Overview of Network Security
* Week 2: Firewall techniques
* Week 3: PKI infrastructure and a case study on e-commerce
* Week 4: IPsec
* Week 5: Securing routing protocols
* Week 6: Wireless security
* Week 7: SSL: Secure socket layer
* Week 8: Distributed denial of service over the Internet
* Week 9: Internet worm study
* Week 10: Traffic analysis attack