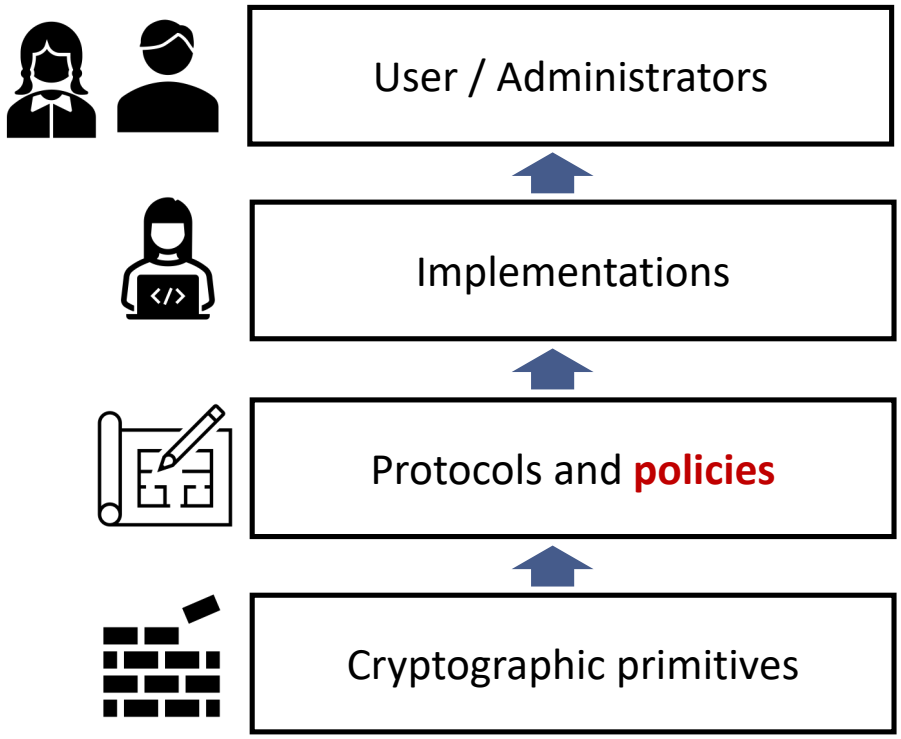
IT-Security

Chapter 9: Access Control, Firewalls, Intrusion Detection

Prof. Dr.-Ing. Ulrike Meyer

Overall Lecture Context

- **So far, we mainly looked at cryptographic protection of data in transit or storage**
  - IPSec, TLS, SSH, PGP, S/MIME, DNSSec
- **Now we look at**
  - **Access Control**: Blocking unauthorized access
    - Specifying a **policy** of who should be allowed to access what and how
  - **Firewalls:** Blocking unwanted network traffic
    - Specifying a **policy** of which traffic to allow and which to deny
  - How to at least **detect intrusions in general** if other security mechanisms fail



Overview

- **Access Control**
  - Access control matrices and lists
  - Discretionary access control
  - Access control on UNIX-based systems
  - Role-based access control
  - Attribute-based access control

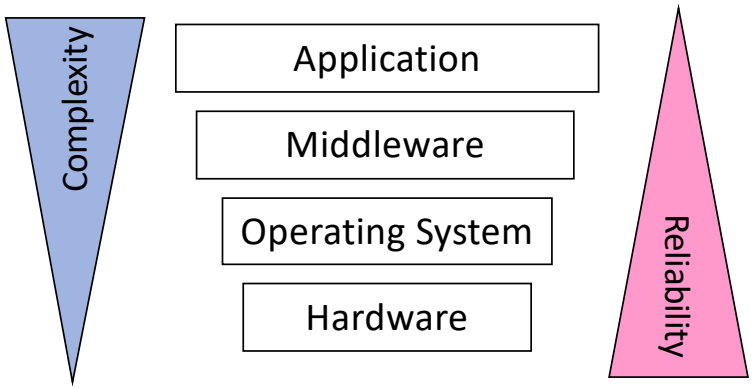
- **Firewalls**
  - Firewalls policy
  - Firewall types
  - Placement of firewalls

- **Intrusion Detection Systems**
  - Components of an IDS
  - Performance and Base-rate fallacy
  - Anomaly vs. Misuse-based detection
  - Host-based vs. network-based
  - Example: SNORT

Access Control

Access Control IETF RFC 4949

Process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities (users, programs, processes, or other systems) according to that policy



Authentication determines who a subject IS

Access Control determines what a subject is AUTHORIZED to DO

Access Control

Access Control IETF RFC 4949

Process by which use of system resources is regulated according to a security policy and is permitted only by authorized entities (users, programs, processes, or other systems) according to that policy

Policy expressed by Access Control Matrix

	Object 1	Object 2	Object 3
Subject 1	rights	rights	rights
Subject 2	rights	rights	rights

Examples for rights: read, write, execute, append,…

Authentication determines who a subject IS

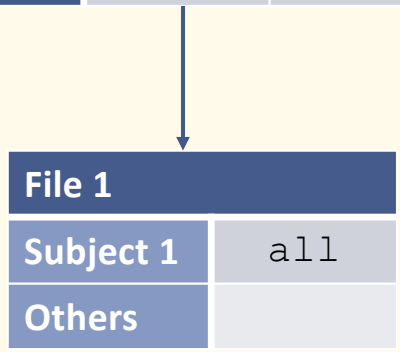
Access Control determines what a subject is AUTHORIZED to DO

Access Control Lists

- **Many cells in an access control matrix are empty**
  - E.g., private files of a subject
- **Access control lists abbreviate matrices by**
  - Storing rows of matrices alongside objects

Access Control Matrix

	File 1	File 2	File 3
Subject 1	all		all
Subject 2		all	read
Subject 3			
Subject 4			
Subject 5			



ACL of File 1

Basic Types of Access Controls

Discretionary Access Control

- Each object has an owner
- Owner decides who may access an object how
  - May include deciding how gets a special `grant access` right

Mandatory Access Control

- A system-wide security policy decrees access to objects
- Compares security labels of objects to security clearances of subjects

Often occur in combination in modern implementations

Discretionary Access Control Implementations Differ in

- **Which subjects can modify an objects ACL**
  - Creator of object
  - Specific right that allows changes (revocation difficult)
- **Privileged user and how ACLs apply to that user**
- **Support of groups or wildcards**
  - Allow to abbreviate ACLs
- **Handling of contradictory permissions**
  - Allow if any permission allows it
  - Deny if any permission denies it
  - Apply first matching entry
- **Application of default settings**

Classical Example for Discretionary Access Control: UNIX File System

- **Each object is associated with three classes of subjects**
  - owner group others
- **Three rights available**
  - `r:read w:write x:execute`
- **ACL for an object indicates**
  - If object is directory or not
  - Rights assigned to each subject class
- **Rights are initially set to default value**
- **Rights can be changed by owner with `chmod`**

Example ACLs

Directory	owner	group	others
d	rwx	r--	---
-	rw-	rw-	r--

Meaning of `chmod abc`

r = 4	w = 2	x = 1
7 = rwx	6 = rw-	5 = r-x
4 = r--	3 = -wx	2 = -w-
1 = --x		

`chmod 715 file` sets `rwx --x r-x`

Meaning of Rights for Directories in UNIX-based Systems

- **The Unix permissions have the following effect on directories**
  - `r` allows listing the content of a directory
  - `w` allows adding or deleting objects to/form a directory
  - `x` allows opening / executing files, cd to subdirectory if x is also set on it
- **Examples**
  - `d rwx r-- r--`
    - Allows group members to list the content of the directory but does not allow them to access any subdirectory
  - `d rwx --x r--`
    - Allows group members to change to subdirectories, open all files in the directory, execute all executables in there, but not to list them, delete them, or add files or subdirectories or executables,…

Unix: Access Decisions and User IDs and Group IDs

- **Each process or subject is associated with a user ID and at least one group ID**
  - Can be a member of more groups as well
- **Access decisions to objects are based on user IDs and group IDs**
- **When a file is created it is owned by a particular user and marked with that user's user ID as owner**
  - It also belongs to a specific group
    - Initially the primary group of its creator or the group of its parent directory if that has the setGID bit set
- **If the userid is 0 (root) then the access control decision is 'yes'**
  - I.e. root can do whatever it likes, some things can only be done by root

Unix: User IDs, Group IDs

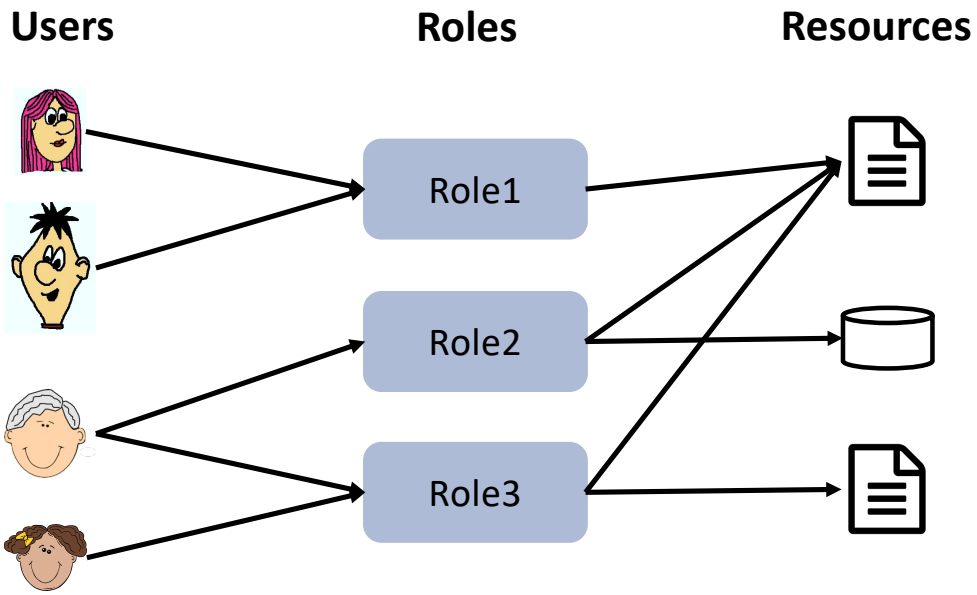
- **Each process has three user IDs and three group IDs**
  - real uid, effective uid, saved uid
  - real gid, effective gid, saved gid
- **Real user ID (ruid)**
  - identifies the owner of the process
- **Effective user ID (euid)**
  - used in most access control decisions
  - can be assigned to a process by a system call (e.g. setuid)
- **Saved user ID (suid)**
  - stores a previous user ID such that it can be restored later
- **Similar: group IDs**

Special Types of Permissions: setuid, setgid, sticky bit

- **When setuid permission is set on an executable file, then**
  - a process that runs this file is granted access based on the userID of the owner of the file
  - The effectiveUID on which the access decision depends is set to the UID of the owner of the file
  - This special permission allows allows the process running the file to access files and directories that are normally available only to the owner.
- **Similar setgid permission**
- **The setuid and setgid permissions are indicated as s instead of x**
  - chmod 2000 sets the setuid bit, chmod 4000 sets the setgid bit
- **The sticky bit protects the files within a directory**
  - If a directory has the sticky bit set, a file within it can be deleted only by file owner, directory owner, or root

Role-based Access Control Models (RBAC)

- Define roles of subjects, e.g. as job functions within an organization
- Assigns access rights to roles instead of individual users
- User are assigned roles according to their responsibilities



Access Matrix Representation of RBAC

Role

User	R_1	R_2	….	R_n
U_1	X			
U_2	X			
U_3		X		X
U_4				X
U_5				X
U_6				X
⋮				
U_m	X			

Object

Role	R_1	R_2	F_1	F_2	P_1	D_2	…
R_1	control		Owner, r		wake	search	
R_2		c	w	x			
R_3							
⋮							
R_n				w	stop		

- **Assignment of users to roles according to their responsibilities**
- **Access control matrix maps Roles to Objects**
  - Allowing for roles as objects allows for hierarchy

Attribute-based Access Control

- **Defines authorizations expressed as conditions on properties of the subject, object, and the environment**
  - E.g. attribute of object: creator of the object
  - Then a single access rule can specify the ownership privilege for any creator of an object
- **Advantage of ABAC**
  - Flexibility and expressiveness
- **Main concern with ABAC**
  - Performance impact of evaluating predicates on objects and user properties for each access
- **Proposed uses include cooperating web services and cloud computing**

Entities and Attributes in the ABAC Model

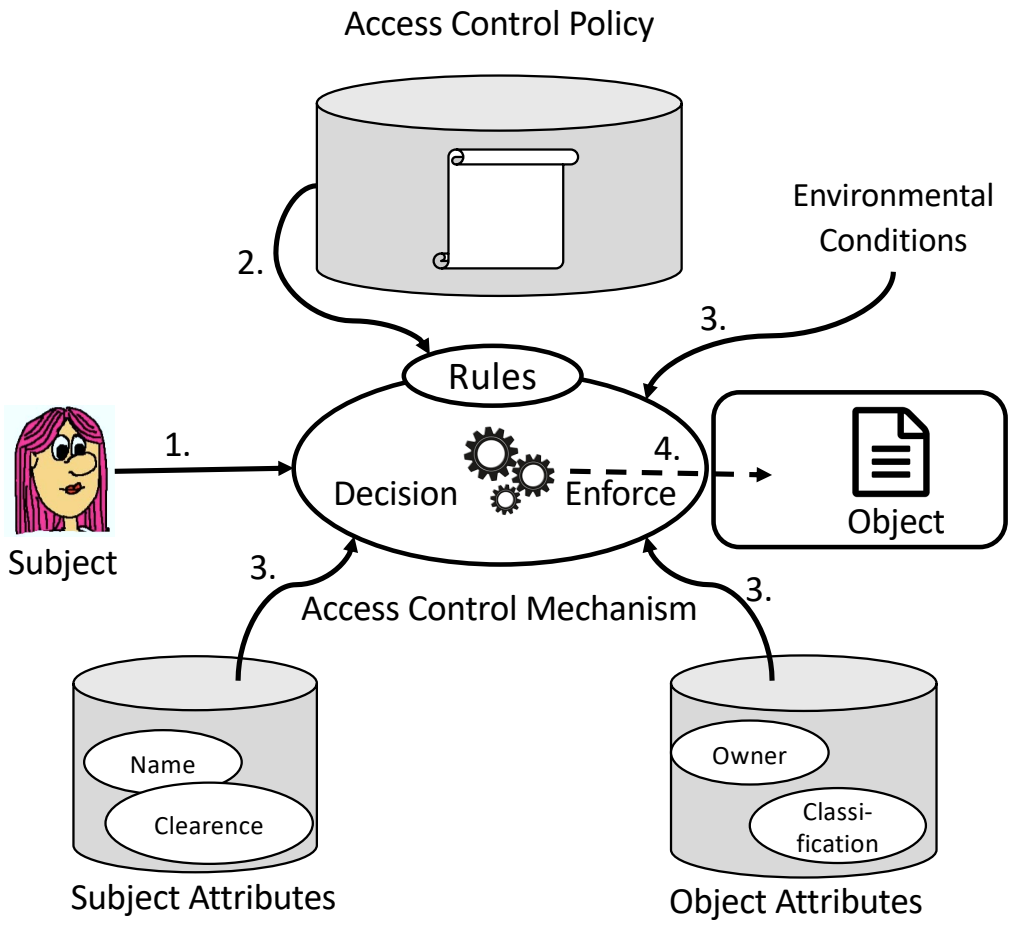
- **Subject**
  - An active entity that causes information to flow amount objects or changes system state
    - E.g. , user, application, process, device
- **Subject attributes**
  - Associated with a subject
  - Define the identity and characteristics of the subject
    - E.g., subject identifier, organization, job title,…
- **Object**
  - Passive system-related entity
    - in the context of a specific request information
    - E.g., device, file, record, table, process, program, network, domain,...
- **Object attributes**
  - Associated with an object
    - E.g., title, creator, date, author,…
- **Environment attributes**
  - Operational, technical, or situations environement or context in which the access occurs
    - E.g. Current date and time, current malware activities,,...

Attribute Evaluation

- **ABAC relies on**
  - Evaluation of attributes of subjects and objects
  - A formal access control rule defining allowable operations for subject/object attribute combinations in a given environment
- **ABAC systems are able to enforce DAC, RBAC, and MAC concepts**
- **ABAC enables fine-grained access control**

ABAC Logical Architecture

1. **Subject request access to object**
2. **Access control mechanism governed by set of rules defined by access control policy**
3. **Based on these rules assesses attributes**
4. **Grants access to object if access is authorized, denies otherwise**



Example: RBAC vs ABAC (1)

- **Assume store must enforce access rule to movies based on user age and movie rating (no environment here)**
- **In RBAC**
  - Users would be assigned one of the three roles adult, juvenile, child
  - Three rights: can view R-rated movies, can view PG-13-rated movies, can view G-rated movies
  - The adult role obtains all three rights, the juvenile role only the last two, child role only the last one

Movie Rating	Users Allowed Access
R	Age 17 and older
PG-13	Age 13 and older
G	Everyone

Example: RBAC vs ABAC (2)

- **In ABAC there is no need for roles, instead whether a user `u` can access a movie `m` given environment `e` is determined by a rule**

```
R1: can_access(u,m,e) ←
        (Age(u)  ≥ 17 ∧ Rating(m) ∈ {R,PG-13,G}) ∨
        (Age(u)  ≥ 13 ∧ Rating(m) ∈ {PG-13,G}) ∨
        (Age(u)  < 13 ∧ Rating(m) ∈ {G}
```

Movie Rating	Users Allowed Access
R	Age 17 and older
PG-13	Age 13 and older
G	Everyone

- **No user to role assignment, no role to rights assignment necessary**

Example: RBAC vs ABAC (3)

- **Advantage of ABAC becomes clearer if we add more attributes**
- **Assume that objects have an additional release date**
  - Divides movies into `new release` or `old release`
- **Users have the attribute `premium user or regular user`**
  - Only `premium users` are allowed to access `new release` movies
- **To capture this new situation in RBAC we would have to**
  - Double the number of roles and double the number of rights
- **In ABAC we just need two new rules in addition to `R1` on last slide:**

```
R2: can_access(u,m,e) ←
        ((MembershipType(u) = Premium) ∨
        ((MembershipType(u) = Regular ∧ MovieType(m) = OldRelease)

R3: can_access(u,m,e) ← R1 ∧ R2
```

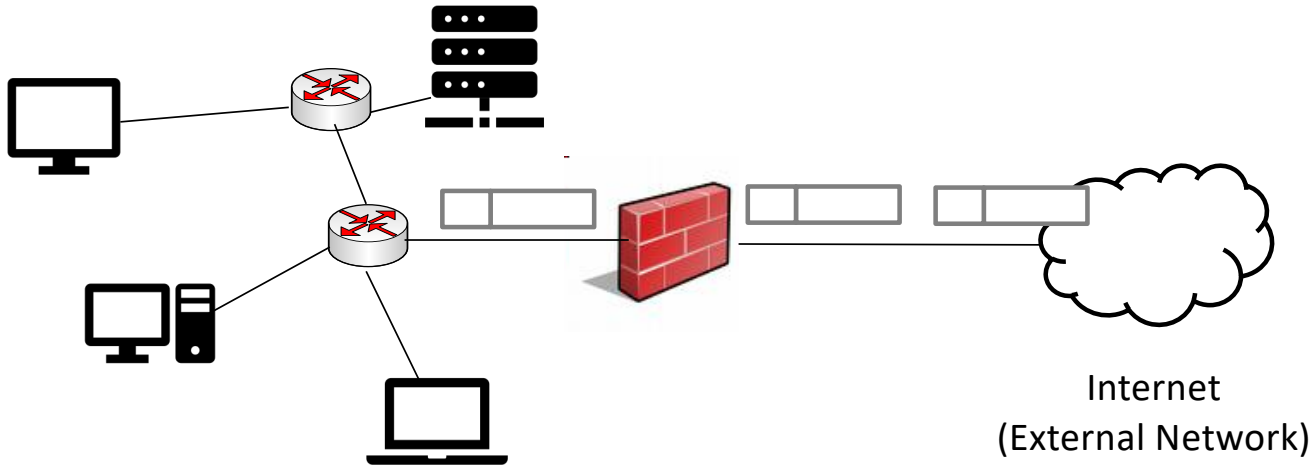
Overview

- **Access Control**
  - Access control matrices and lists
  - Discretionary access control
  - Access control on UNIX-based systems
  - Role-based access control
  - Attribute-based access control

- **Firewalls**
  - Firewalls policy
  - Firewall types
  - Placement of firewalls

- **Intrusion Detection Systems**
  - Components of an IDS
  - Performance and Base-rate fallacy
  - Anomaly vs. Misuse-based detection
  - Host-based vs. network-based
  - Example: SNORT

Principle of Firewalls



- **A network firewall**
  - Controls access between an internal network and an external network
  - Allowing or denying (IP) packets according to a **security policy**
- **The internal network is to be secured, the external network is not trusted**

Firewall Policy

- **When a packet arrives at a firewall, a security policy is applied to determine the appropriate action**
  - Accept / deny
  - If a packet is denied it is either silently dropped or bounced back
  - In addition a firewall often logs information about packets arriving

- **A firewall policy can be viewed as a list of rules**
  - Each rule consists of a set of tuples and actions
  - Each tuple corresponds to a field in the packet header
    - E.g for IP packets: the protocol type, source IP, destination IP, source port, destination port

Simple Example for a Firewall Policy

No.	Protocol	Src IP	Src Port	Dest IP	Dest Port	Action
1	UDP	190.1.1.*	*	*	80	deny
2	TCP	180.*	*	180.*	90	accept
3	UDP	210.1.*	*	*	90	accept
4	TCP	210.*	*	220.*	80	accept
5	UDP	190.*	*	*	80	accept
6	*	*	*	*	*	deny

- **Simple packet filter firewall policy**
  - Rules can be fully specified or contain wildcards
  - Header information of passing packet is compared to the fields of a rule
  - If the packet header information is a subset of the rule, the packet is said to match the rule

Rule Matching Policy: First Match Policy

Most firewalls use a first-match policy as rule matching policy

- The packet header information is matched sequentially with the rules **starting from the first rule**
- The action of the **first matching rule** is executed
- Any other rules further down in the policy that may also match the packet are ignored

- A **default rule** is often placed at the end of a policy **with action deny**
  - Makes the policy comprehensive

No.	Protocol	Src IP	Src Port	Dest IP	Dest Port	Action
1	UDP	190.1.1.*	*	*	80	deny
2	TCP	180.*	*	180.*	90	accept
3	UDP	210.1.*	*	*	90	accept
4	TCP	210.*	*	220.*	80	accept
5	UDP	190.*	*	*	80	accept
6	*	*	*	*	*	deny

Example

No.	Protocol	Src IP	Src Port	Dest IP	Dest Port	Action
1	UDP	190.1.1.*	*	*	80	deny
2	TCP	180.*	*	180.*	90	accept
3	UDP	210.1.*	*	*	90	accept
4	TCP	210.*	*	220.*	80	accept
5	UDP	190.*	*	*	80	accept
6	*	*	*	*	*	deny

- **Assume the following packet arrives:**
  - TCP, 210.1.1.1:3080, 220.2.33.8:80
- **What will be the rule to apply?**

Example

No.	Protocol	Src IP	Src Port	Dest IP	Dest Port	Action
1	UDP	190.1.1.*	*	*	80	deny
2	TCP	180.*	*	180.*	90	accept
3	UDP	210.1.*	*	*	90	accept
4	TCP	210.*	*	220.*	80	accept
5	UDP	190.*	*	*	80	accept
6	*	*	*	*	*	deny

- **Assume the following packet:**
  - TCP, 210.1.1.1:3080, 220.2.33.8:80
- **First matching rule: rule 4, action: accept**

Simple Mathematical Model of a Packet Filtering Firewall

- **Each tuple in a rule can be modeled as set of packets**
  - E.g ,the tuple 198.188.150.* corresponds to the set of IP addresses ranging from 198.188.150.0 to 198.188.150.255
- **The tuples of a rule collectively define a set of packets that match this rule**
  - E.g. the rule Proto = TCP, SIP = 190.150.140.38, SP = 188, DIP = 190.180.39.*, DP = 80, action = accept defines a set of 256 unique packet headers that match this rule
- **The overall set of possible packets is denoted by P**
- **Each firewall policy R can be described by three sets**
  - A(**R**) ⊆ **P** describes the set of **packets** that will be **accepted**
  - D(**R**) ⊆ **P** describes the set of **packets** that will be **denied**
  - U(**R**) ⊆ **P** describes the set of **packets** that **do not match any rule in the policy**

Simple Mathematical Model (2)

- **A firewall policy R is considered comprehensive if any packet from P will match at least one rule**
  - I.e. A(**R**) ∪ D(**R**) = **P** or equivalently U(**R**) = ∅
  - typically ensured by adding a default rule of “deny” at the end of the policy
- **This simple model also allows to compare two policies**
  - Assume two firewall policies **R**, **S**
  - The two policies are said to be **equivalent** if their accept, deny and non-match sets are the same
- **Note that being equivalent does not mean that the two policies have the same rules!!**
  - Just that given any packet the two policies will lead to the same actions always

Anomalies on First-Match Policies - Shadowing

- **In first-match policies more specific policy rules typically appear at the beginning of the policy and more general ones appear towards the end**
- **An anomaly is an unintended consequence of adding rules in a certain order**
  - Introducing anomalies into large firewalls is very easy
- **Example: shadowing**
  - Occurs if an earlier rule i matches every packet that another lower rule j (j > i) matches

No.	Protocol	Src IP	S - Port	Dest IP	D - Port	Action
i	TCP	190.150.140.38	188	190.180.39.*	80	accept
j	TCP	190.150.140.38	188	190.180.39.180	80	drop

- **If both rules have the same action, this is not a problem but if e.g. rule i is added after rule j the**

Anomalies on First-Match Policies – Half-Shadowing

- **Only a portion of an earlier rule i shadows a lower rule j (j > i)**
  - For example

No.	Protocol	Src IP	S - Port	Dest IP	D - Port	Action
i	TCP, SIP	190.150.140.38	188	190.180.39.*	80	accept
j	TCP, SIP	190.150.140.38	*	190.180.39.180	80	drop

  - The rule j is partially shadowed by the first rule i
  - By itself rule j will drop any TCP packet arriving from 190.150.140.38 and destined to 190.180.39.180 on port 80
  - When rule i is added before rule j, then any packet like this with source port 188 will be accepted
  - Only the system administrator will typically know whether or not this behavior was intended

Policy Optimization

- **The number of firewall rules will typically impact the firewall performance**
  - Every rule requires some processing time
  - More rules will require more processing time on average
- **Ways to enhance performance through optimizing the policy**
  - Policy reordering such that rules that match more packets are placed earlier in the policy
    - Must be done with care to avoid violating the integrity of a policy
    - I.e., after reordering, the policy should still accept and deny the same packets
  - Removing unnecessary rules by
    - Removing redundant rules
    - Combining rules if possible

Removing Unnecessary Rules

- **Removing redundant rules**

No.	Protocol	Src IP	S - Port	Dest IP	D - Port	Action
i	TCP	190.150.140.38	188	190.180.39.*	80	drop
j	TCP	190.150.140.38	188	190.180.39.180	80	drop

- **Combining several rules …**

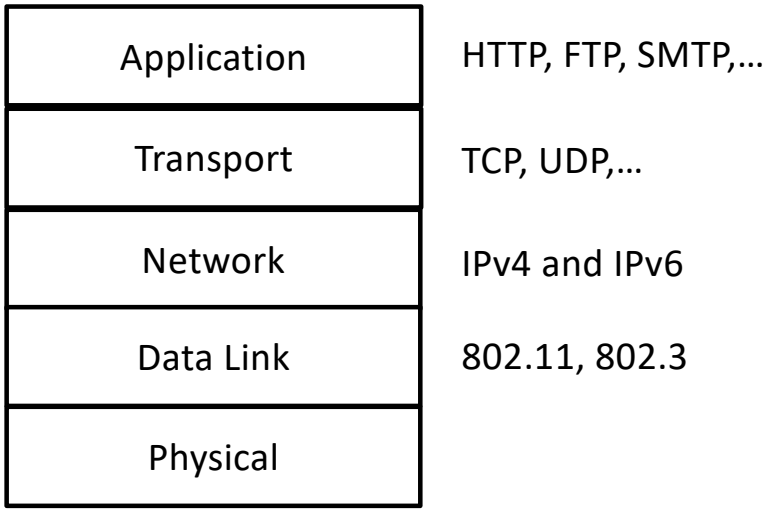
No.	Protocol	Src IP	S - Port	Dest IP	D - Port	Action
i	TCP	190.150.140.38	188	190.180.39.*	80	accept
j	UDP	190.150.140.38	188	190.180.39.*	80	accept

- **… to one**

No.	Protocol	Src IP	S - Port	Dest IP	D - Port	Action
i	*	190.150.140.38	188	190.180.39.*	80	accept

Firewall Types

- **Firewalls can be categorized into three general classes**
  - Packet filters
  - Stateful firewalls
  - Application layer firewalls



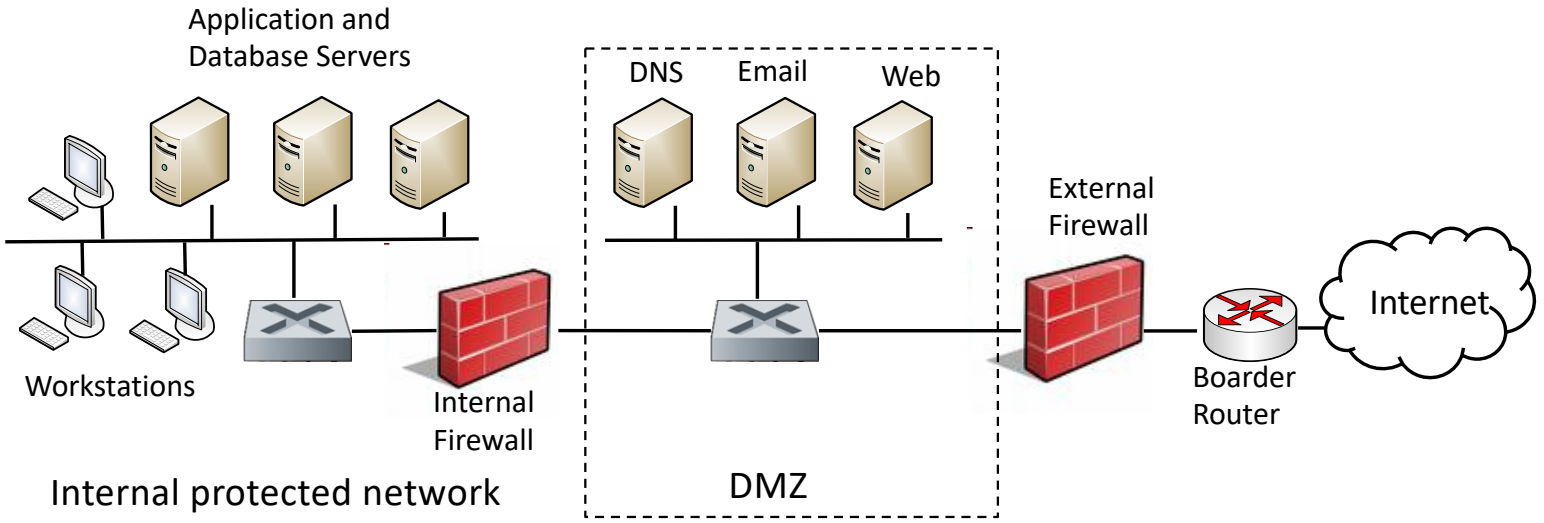
Why Stateless Filtering Is Not Enough

- **In TCP connections, ports with numbers less than 1024 are permanently assigned to servers**
  - 20,21 for FTP, 23 for telnet, 25 for SMTP, 80 for HTTP…
- **Clients use ports numbered from 1024 to 49151**
  - They must be available for clients to receive responses
- **Dynamic and/or Private Ports: 49152 through 65535**
- **What should a firewall do if it sees, say, an incoming request to some client's port 5612?**
  - It must allow it: this could be a server's response in a previously established connection…
  - …OR it could be malicious traffic
  - Can't tell without keeping state for each connection

Stateful Packet Firewalls

- **Stateful firewalls perform the same operations as packet filter**
- **But they enable connection tracking**
  - E.g., if no stateful packet filter is used, allowing internal users to connect to any external webserver will require two rules
    - One for outgoing traffic to any webserver
    - One for incoming traffic to any user regardless of whether a user requested traffic from that webserver
  - A stateful firewall can support a more restrictive policy that allows incoming traffic from webservers only in response to requests by users
  - Dynamically add a rule to the policy that allows return packets when a connection is started
  - Delete this rule when the connection is closed
    - Typically based on timers as it is hard for the firewall to reliably determine whether a connection is closed

Firewall Placement: Using a Demilitarized Zone (DMZ)



- **Internal firewall adds more strict filtering capabilities compared to external firewall**
- **Internal firewall provides two-way protection to DMZ**
  - Filter attacks from DMZ towards internal network and vice versa
- **Multiple internal firewalls can be used to protect portions of internal network from each other**

Application Layer Firewalls

- **Can filter traffic at the network, transport, and application layer**
- **Typically come with proxy capabilities**
  - Application proxies are intermediaries for network connections
  - If a user on the internal network wants to connect to an application server on the external network
    - The proxy (here the firewall) would terminate the connection
    - The proxy would then create a connection to the external server
- **The firewall can thus inspect the content of the packets**
  - Like an intrusion detection system
- **Application layer firewalls and other security devices are being merged into one device**
  - E.g. intrusion prevention systems combine firewalls with intrusion detection
    - Can often filter packets as well as inspect packet contents for viruses, spam, attack signatures

Overview

- **Access Control**
  - Access control matrices and lists
  - Discretionary access control
  - Access control on UNIX-based systems
  - Role-based access control
  - Attribute-based access control
- **Firewalls**
  - Firewalls policy
  - Firewall types
  - Placement of firewalls
- **Intrusion Detection Systems**
  - Components of an IDS
  - Performance and Base-rate fallacy
  - Anomaly vs. Misuse-based detection
  - Host-based vs. network-based
  - Example: SNORT

Intrusion Detection

- **Definitions from IETF RFC 4949 ”Internet Security Glossary”**

Security Intrusion: A security event, or a combination of multiple security events that constitutes a security incident in which an intruder gains, or attempts to gain, access to a system (or system resource) without having authorization to do so

Intrusion Detection: A security service that monitors and analyzes system events for the purpose of finding, and providing real-time or near real-time warning of attempts to access system resources in an unauthorized manner

Logical Components of an Intrusion Detection System

Sensors

- Collect data from a monitored part of the system
  - Input to a sensor can, e.g., include network packets, log files, or system call traces recorded on a particular system

Analyzers

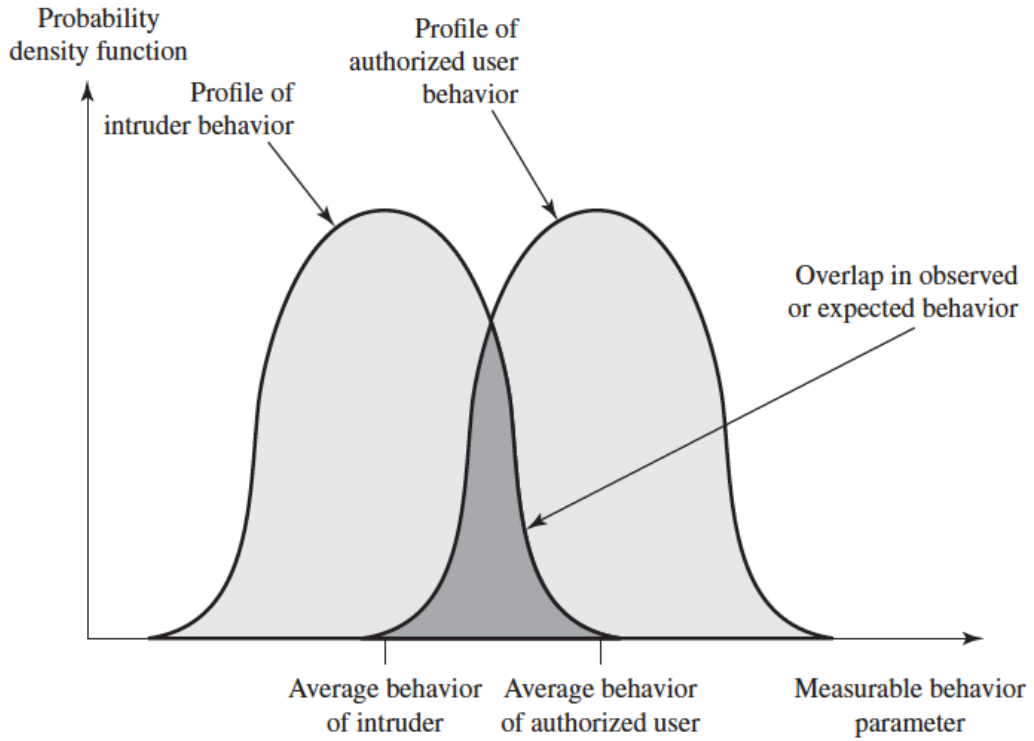
- Analyzers receive and store data collected by one or more sensors
- Tries to determine if an intrusion has occurred
- Output may include
  - Evidence supporting the detection
  - Guidance about appropriate actions to take

User Interface

- Enables user to view output from analyzer or control behavior of the IDS's components

Basic Principles (1)

- **Basic assumption underlying intrusion detection systems**
  - Behavior of intruders differ from that of legitimate users in a quantifiable way



Basic Principles (2)

- **Typical behavior of an intruder differs from typical behavior of an authorized user, but there is an overlap in these behaviors**
- **So, any intrusion detection approach will make mistakes**
  - If it tries to catch all intruders, it will typically sometimes raise false alarms, i.e., cause false positives
  - If it tries to limit false alarms it will typically miss some attacks, i.e., cause false negatives

TP = True Positives = Attacks rising alarm
FN = False Negatives = Attacks not rising alarm
FP = Benign behavior rising alarm
TN = Benign behavior not rising alarm

- **Ideally one would want an IDS to**
  - Maximize the **detection rate**, i.e., the ratio of detected to total attacks
    - = Recall = TP / (TP + FN)
  - Minimizing the **false alarm rate**, i.e., ratio of false positive to all negatives
    - = False Positive Rate = FP / (FP + TN)

Problem: Base Rate Fallacy

- **It is very difficult to meet this goal of high detection rate and low false alarm rate**
- **In general**
  - if the actual numbers of intrusions is low compared to the number of legitimate uses of a system
  - Then if an alarm is raised the probability that indeed an attack takes place is very low unless the detection is extremely discriminative
- **This phenomenon is called the base rate fallacy**

Reminder: Conditional Probability and Bayes Theorem

- **Suppose two events A and B occur with probability $\Pr(A)$ and $\Pr(B)$, respectively**
- **Let $\Pr(A \cap B)$ be probability that both A and B occur**
- **What is the conditional probability that A occurs given that B has occurred?**

$$\Pr(A \mid B) = \frac{\Pr(A \cap B)}{\Pr(B)}$$

- **Applying this twice we get Bayes Theorem**

$$\Pr(B \mid A) = \frac{\Pr(A \mid B)\Pr(B)}{\Pr(A)}$$

Law of Total Probability

- **Suppose mutually exclusive events E_1, … ,E_n together cover the entire set of possibilities**

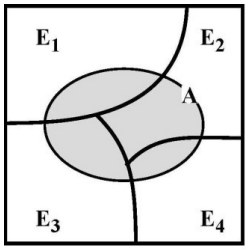
- **Then probability of <u>any</u> event A occurring is**

$$\Pr(A) = \sum_i \Pr(A \mid E_i) \cdot \Pr(E_i)$$

Intuition: since E_1, … ,E_n cover entire

probability space, whenever A occurs,

some event E_i must have occurred



Example for Base-Rate Fallacy

- **Assume**
  - 1% of traffic is SYN floods, 99% of traffic is valid connections
  - IDS's detection rate is 90%, i.e. IDS classifies 90% of SYN floods as attack
  - IDS's false alarm rate is 1%, i.e. IDS classifies 1% of valid connections as attack

- **What is the probability that a connection flagged by IDS as a SYN flood really is a valid connection?**

$$\Pr(\text{valid} \mid \text{alarm}) = \frac{\Pr(\text{alarm} \mid \text{valid}) \bullet \Pr(\text{valid})}{\Pr(\text{alarm})}$$

$$= \frac{\Pr(\text{alarm} \mid \text{valid}) \bullet \Pr(\text{valid})}{\Pr(\text{alarm} \mid \text{valid}) \bullet \Pr(\text{valid}) + \Pr(\text{alarm} \mid \text{SYN flood}) \bullet \Pr(\text{SYN flood})}$$

$$= \frac{0.01 \bullet 0.99}{0.01 \bullet 0.99 + 0.90 \bullet 0.01}$$

≈ 52% chance that traffic is valid given an alarm is raised

General IDS Approaches

- **Anomaly Detection**
  - Collect data corresponding to behavior of legitimate users over a period of time
  - Built a model of normal behavior from it
  - Try to determine whether current behavior is of a legitimate user or of intruder by comparing it to the model

- **Signature or Heuristic detection**
  - Use a set of known malicious data patterns (signatures) or attack rules (heuristics) and compare them to currently observed data
  - Also known as **misuse detection**
  - Can only identify known attacks

In Other Words,…

- **Anomaly detection assumes that**
  - What is usual, is known
  - What is unusual, is bad
  - Problem
    - Does not necessarily detect undesirable yet usual behavior
    - False alarm rates can be high
    - Very hard to obtain (attack free) usual behavior
- **Misuse detection assumes that**
  - What is bad, is known
  - What is not bad, is good
  - Problem
    - Cannot detect new attacks, i.e. false negatives typically very high

Anomaly Detection

- **Training: develop a model of legitimate behavior**
  - Collect and process sensor data from normal operation of monitored system
  - May occur at distinct times or may be a continuous of monitoring and evolving the model
- **Detection: Compare observed sensor data to the trained model**
  - Classify as normal or anomalous activity
- **Detection approaches**
  - **Statistical**: analysis of the observed behavior using univariate, multivariate, or time-series models of observed metrics
  - **Knowledge-based**: approaches use an expert system that classifies observed behavior according to a set of rules that model legitimate behavior
  - **Machine-learning**: approaches automatically determine a suitable model from features extracted from the sensor data using machine-learning techniques
    - E.g. Bayesian networks, Markov models, neural networks, fuzzy logic, clustering, but also classifiers such as SVMs, Random Forests, deep neural networks,…
    - Often make use of attack data as well, i.e. train model with benign AND malicious data

Misuse Detection

- **Signature approaches**
  - Match large collection of known attack patterns against data monitored on a system or in transit over the network
  - Signatures need to be specific enough to minimize false alarm rate but still detect malicious data
  - Typically, low cost with respect to time and resources
  - But: significant effort necessary to review new attacks and generate signatures
- **Rule-based heuristic identification**
  - Rules that identify suspicious behavior
  - Typically, specific to the machine and operating system monitored
  - Often derived from analyzing attack tools and scripts collected on the Internet
  - **SNORT** is a rule-based network intrusion detection system

Misuse Detection

- **Set of patterns defining a behavioral signature likely to be associated with attack of a certain type**
  - Example: buffer overflow
    - A setuid program spawns a shell with certain arguments
    - A network packet has lots of NOPs in it
    - Very long argument to a string function
  - Example: SYN flooding (denial of service)
    - Large number of SYN packets without ACKs coming back
    - …or is this simply a poor network connection?
- **Attack signatures are usually very specific and may miss variants of known attacks**
  - Why not make signatures more general?

Extracting Misuse Signatures

- **Use invariant characteristics of known attacks**
  - Bodies of known viruses and worms, port numbers of applications with known buffer overflows, RET addresses of overflow exploits
  - Hard to handle mutations
    - Polymorphic viruses: each copy has a different body
- **Big research challenge: fast, automatic extraction of signatures of new attacks**
- **Honeypots are useful for signature extraction**
  - Try to attract malicious activity, be an early target

Host-based Intrusion Detection Systems (HIDS)

- Examines user and software activity on a specific host
- Aims to detect both attacks from the outside as well as internal attack
- Can use anomaly or misuse-based detection approach
- But: provide only a local view on an attack
- Are only able to detect attacks when they already hit the target system

Data Sources for HIDS

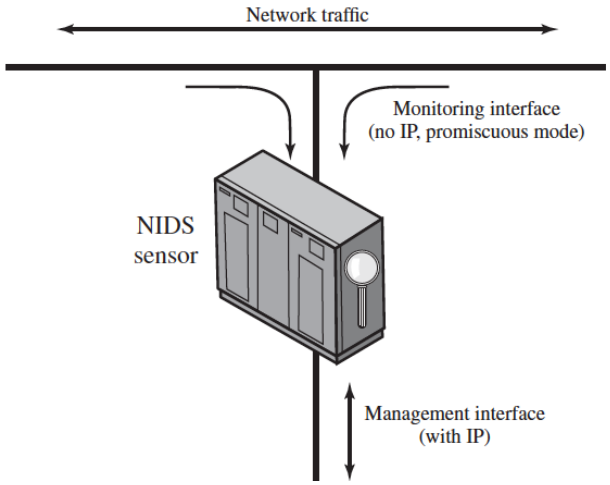
- **System call traces:** Record sequences of system calls made by processes on the system
  - Works well on Linux and Unix systems
  - Difficult on Windows systems as use of DLLs hides which process uses which system calls (use of DLL function calls proposed as alternative)
- **Audit (log file) records:** Operating systems include software that collects information on user activity
  - Problem: audit records may not include relevant information; intruder may manipulate the records
- **File integrity checksums:** Detect intruder activity on a system by periodically scanning critical files for changes
  - Use message authentication code to compute checksums, compare them to a known baseline
  - Tripwire is a well-known system using this approach
- **Registry access:** monitor access to the registry on Windows machines

Network-Based Intrusion Detection System (NIDS)

- **Monitors traffic at selected points on a network**
  - Examines traffic for a large number of hosts with a variety of devices and software
  - Examines traffic packet by packet in real-time or close to real-time
  - May examine network-, transport- and/or application-level protocol activity
  - Typically included in perimeter security infrastructure, e.g. firewall
- **Challenge**
  - Arranging the monitoring to minimize the number of agents but cover the complete network
- **Agent must have same view of traffic as destination**
  - TTL tricks, fragmentation may obscure this
  - End-to-end encryption defeats content monitoring
    - Not traffic analysis, though

Types of NIDS Sensors

- **Inline sensor**
  - Inserted into a network segment such that monitored traffic must pass the sensor
  - E.g., incorporate directly in firewall or as standalone component
- **Passive sensor**
  - Monitors copy of network traffic, actual traffic does not pass through it



Logging of Alerts

- **When a sensor detects potential violations it**
  - Sends an alert
  - Logs information related to the event
- **Typical information logged by an NIDS sensor includes**
  - Timestamp
  - Connection or session ID
  - Event or alert type and Rating (e.g., severity, impact, confidence,…)
  - Network, transport, and application layer protocol
  - Source and destination IPs and ports, number of bytes transmitted over the connection
  - Decoded payload data
  - State-related information (e.g., authenticated username)

Summary

- **In this chapter we looked at basic non-cryptographic security mechanisms**
- **Access Control**
  - Access Controls implement an access policy
  - An access policy determines which subjects have which rights over which objects
    - As opposed to authentication which determines who is who
  - Access Controls can be implemented on all layers of a system
    - Hardware, operating system, middleware, application
  - In discretionary access control
    - Each object has an owner, access to object is at the owner's discretion
  - In mandatory access control
    - a global policy determines access to all objects
  - In role-based access control roles are used as subjects and users are assigned one or more roles

Summary

- In attribute-based access control
  - Access is granted based on attributes of subjects, objects and the environment they act in

- **Network Firewalls**
  - Control network traffic flow to and from one network or network segment to another
  - In packet flters IP packets are accepted or blocked dependent on
    - Header Information in TCP/IP headers
  - In stateful firewalls additional state is kept on previously inspected packets and acceptance / denial depends on this state
  - In application layer firewalls
  - Application layer content is inspected

Summary

- **Intrusion Detection Systems**
  - Consist of sensors that collect information and analyzers that receive information from sensors
  - **Network-based** intrusion detection systems
    - focus on sensors that collect information on network traffic
  - **Host-based** intrusion detection systems
    - focus on sensor that collect information on individual hosts
  - Intrusion detection systems try to detect attacks and provide evidence
    - Underlying assumption: attack will be visible in the collected information
  - The **goal** of an IDS is to
    - Mximize the **detection rate**, i.e., the ratio of detected to total attacks and to
    - Minimizing the **false alarm rate**, i.e., ratio of false positive to all negatives

Summary

- Main approaches for IDS
  - Anomaly-based approach
    - Model normal behavior, classify anormal behavior as attack
  - Misuse-based approach
    - Model attack behavior and try to detect it
  - Model both normal and attack behavior and try to distinguish it
- Snort is an example for a
  - host-based intrusion detection system
  - that collects information on network traffic of a host
  - it uses a misuse-based approach

References

- **W. Stallings, Cryptography and Network Security: Principles and Practice, 8th edition, Pearson 2022**
  - Chapter 21: Network Endpoint Security
    - Firewalls, Intrusion Detection Systems
- **Wenliang Du, Computer Security a Hands-on Approach, 3rd edition, 2022**
  - Chapter 1: Linux Security Principle
    - Access Control