Security

Samenvatting

Manuel Mol .

Samenvatting

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut postea variari voluptas distinguique possit, augeri amplificarique non possit. At etiam Athenis, ut e patre audiebam facete et urbane Stoicos irridente, statua est in quo a nobis philosophia defensa et collaudata est, cum id, quod maxime placeat, facere possimus, omnis voluptas assumenda est, omnis dolor repellendus. Temporibus autem quibusdam et.

Samenvatting

Published Jun 07, 2024

Introduction to Information Security

Three important security principles are:

1. Confidentiality: Only authorized users should be able to access data.
2. Integrity: Data should not be altered by unauthorized users.
3. Availability: Data should be available when needed.

AAAA properties:

1. Authenticity: The identity of the user should be verified.
2. Authorization: The user should have the necessary permissions.
3. Accuracy: The data should be accurate.
4. Accountability: The user should be accountable for their actions. (logs)

STRIDE:

1. Spoofing: Pretending to be someone else.
2. Tampering: Altering data.
3. Repudiation: Denying an action.
4. Information Disclosure: Unauthorized access.
5. Denial of Service: Preventing access.
6. Elevation of Privilege: Gaining unauthorized access.

Stride steps

1. Define the key assets and security requirements for the system
2. Design data flow diagram(s) for the system
3. Draw trust boundaries
4. Identify threats
5. Mitigate threats with controls
6. Validate that threats were mitigated

Successful attack:

1. System susceptibility: Vulnerability
2. Threat accessibility: Attack surface
3. Threat capability: recourses, tools, knowledge

Why is security hard?

1. Complexity: More complex systems are harder to secure.
2. Afterthought: Security is often added after the fact.
3. Benefits are evident best after a failure

Security is needed to prevent and counteract the unwanted consequences

Trade-offs:

1. Security vs. Usability
2. Security unaware users want security
3. Security has cost, but becomes only a direct gain when a failure occurs
4. Failure can cost less than prevention
5. Algo is secure but the implementation is not
6. Practical security is ofter weaker than theoretical security

7. Complexity increases the attack surface

Risk management:

Enterprise Risk Management:

1. Identify risks
2. Evaluate risks
3. Mitigate risks
4. Monitor risks
5. Review risks

Risk assessment:

1. Assess risk and determine need
2. Implement policy and controls
3. Promote awareness
4. Monitor and evaluate

Principles, Best Practices and Standards

Design principles:

- Separation of duties: No single person should have all the power
- Least privilege: Only the necessary permissions

Standards and best practices:

- ISO/IEC 27001: Information Security Management System
- NIST: National Institute of Standards and Technology
- ANSI: American National Standards Institute

Network security:

OSI model: 7 layers

1. Physical: Fiber, copper
2. Data link: Ethernet, switch, bridge
3. Network: IP, ICMP, ARP
4. Transport: TCP, UDP
5. Session: API, sockets
6. Presentation: FTP, SSL, IMAP
7. Application: HTTP, DNS, SMTP

Fragmentation
Splitting data into smaller parts

Network layer:

- **IPsec:** Security protocols used in VPNs
- **Authentication header:** a header that shows that the data hasn't been tampered with
- **ESP header:** does more that AH, also encrypts the data and authenticates it.

- **IKE:** a protocol used to establish a secure connection, a handshake
- **ICMP redirect attack:** router can redirect with ICMP packets. Attacker can forge these packets to redirect to their own machine.

Transport layer:

- **TCP three way handshake:** SYN (ask server), SYN-ACK (server ack, asks client), ACK (client ack)
- **SYN Flooding Attack:** DoS with SYN packets

Presentation:

- **SSH:** Secure Shell, encrypted terminal. Steps:
    1. Client initiates connection
    2. Server sends public key
    3. Negotiate encryption
    4. Open session

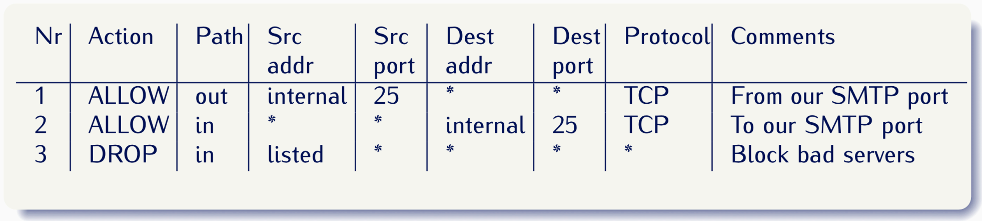
Application layer:

- **DNS:** Domain Name System, translates domain names to IP addresses
- **DNS poisoning:** Changing DNS records to redirect traffic to a malicious server

Firewalls:

A firewall can perform a few actions based on the rules:

- Allow: Allow the packet
- Reject: Block the packet, inform
- Drop: Silently discard the packet



Nr	Action	Path	Src addr	Src port	Dest addr	Dest port	Protocol	Comments
1	ALLOW	out	internal	25	*	*	TCP	From our SMTP port
2	ALLOW	in	*	*	internal	25	TCP	To our SMTP port
3	DROP	in	listed	*	*	*	*	Block bad servers

Figure 1: Firewall

Proxy firewalls: Inspects packets and can block based on content. Can do more intelligent filtering.

Intrusion Detection Systems:

An intrusion detection system collects events, processes them and alerts when something is wrong. You can source the data from:

- Network: Network-based IDS (packet sniffing)
- Host: Host-based IDS (logs)

Bayes' theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

- Left side of equation $P(A|B)$
  - Probability of event A given another event B (This situation is also called posterior outcome)
  - Signifies the updated probability of A, when a new evidence / event B occurs
- Right side of equation: (how it happens)
  - $P(A)$ = Probability of event A if there is no event B (This situation is also called prior outcome)
  - $P(B)$ = marginal likelihood. (probability of observing new event B)
  - $P(B|A)$ = The likelihood of B when event A is present

At its simplest, Bayes' Theorem takes a test result and relates it to the conditional probability of that test result given other related events. For high-probability false positives, the theorem gives a more reasoned likelihood of a particular outcome.

Event analysis:
- Signature-based: Known patterns, fast, but can't detect new attacks
- Anomaly-based IDS: Detects deviations from normal behavior (failed login attempts, total files deleted, etc.), can detect new attacks but hard

IPS:

An intrusion prevention system is like an IDS but can also take action. It can:
- IPS augmenting firewall: alter packets, strip out malware, terminate connections
- in-host IPS: terminate processes

Application security:

The **stack** and the **heap** are two memory areas in a program. The stack is used for function calls and the heap for dynamic memory allocation. The stack is faster but limited in size.

Stack:

- Calling function main:
  1. Push arguments of func in reverse order on the stack
  2. Push the return address where main wants to get back
  3. Jump to the func address
- Called function func:
  1. Push the current (main's) frame pointer to the stack
  2. Set new %ebp to the current %esp (where the stack ends now)
  3. Push local variables on the stack
- Return to main:
  1. Reset the stack frame: %esp = %ebp, %ebp = (%ebp)
  2. Jump back to the planned return address: %eip = 4(%esp)
- **%EBP:** base pointer, points to the base of the current stack frame
- **%ESP:** stack pointer, points to the top of the stack
- **%EIP:** instruction pointer, points to the next instruction to execute

Buffer overflow:

A buffer overflow is when a program writes more data to a buffer than it can hold. This can overwrite the return address %EBP. This can be used to execute arbitrary code. You can also inject shellcode and point %EIP to it.

We can use **NOP** instructions to slide to the shellcode. This is called a **NOP sled**.

Heap overflow:

A heap overflow is when a program writes more data to the heap than it can hold. This can overwrite the heap metadata and corrupt the heap. This can be used to execute arbitrary code. **Malloc** and **free** are common functions that can be exploited.

Heartbleed:

Heartbleed is a vulnerability in OpenSSL that allows an attacker to read memory from the server. This can include private keys, passwords, etc.

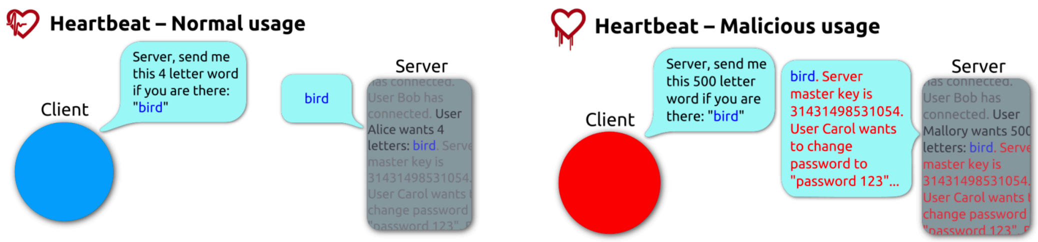

Heartbeat – Normal usage
Client
Server, send me this 4 letter word if you are there: "bird"
bird
Server
User Bob has connected. User Alice wants 4 letters: bird. Server master key is 31431498531054. User Carol wants to change password "password 123".
Heartbeat – Malicious usage
Client
Server, send me this 500 letter word if you are there: "bird"
bird. Server master key is 31431498531054. User Carol wants to change password to "password 123"...
Server
User Bob has connected. User Mallory wants 500 letters: bird. Server master key is 31431498531054. User Carol wants to change password "password 123".

Figure 2: Heartbleed

String formatting:

String formatting is when a program uses a format string to print data. If the format string is user-controlled, an attacker can read and write memory.

Race conditions:

Modify data between the check and the use. This can be done with multiple threads or processes.

Prevention techniques:

1. Memory safety: all its possible executions are memory safe. Things like Rust, GO and Java are memory safe.
2. Spatial Safety: check if you are writing to the correct memory
3. Temporal Safety: A temporal safety violation occurs when trying to access undefined memory

Spatial safety ensures access is to a legal region Temporal safety ensures it is still allocated and initialized (still valid)

Type safety:

Type safety ensures that operations on the object are always compatible with its type.

Avoiding exploitation:

- Make the exploitation steps more difficult or impossible
- Avoid bugs via secure coding and better code audit

Fix the architecture:

- Use a language that is memory safe
- Implement security checks

make stack and heap non-executable, so that the architecture will not support running anything there (non-code memory region).

Stack Canaries:

A stack canary is a random value placed before the return address. If the canary is overwritten, the program will exit. This can prevent buffer overflows.

ASLR:

Address Space Layout Randomization is a technique that places standard libraries and other useful elements in memory. This makes it harder for an attacker to predict where things are.

How can we detect an ongoing attack?:

If we observe that a program deviates from its expected behavior, then it might be compromised.

We can build a CFG to see if the program deviates from its normal flow. We can then monitor the program and see if it deviates from the CFG. This can be done by inserting labels in the code just before the target address of an indirect transfer. *indirect calls* will then be checked against the CFG.

In-line Reference Monitor

This is how we can detect an ongoing attack:

1. insert a label just before the target address of an indirect transfer
2. insert code to check the label of the target at each indirect transfer
3. stop execution if labels don't match

We can use detailed labeling to check if we aren't reusing code that should not be reused.

Symmetric Cryptography:

The goals of cryptography are:

- Confidentiality: Only authorized users can read the data
- Authentication: Verify the identity of the sender
- Non-repudiation = assuring that actions or commitments in the past cannot be denied

There are a few different classifications for cryptographic algorithms:

- **Symmetric:** Same key for encryption and decryption, mainly used for **confidentiality**

- **Public-key cryptography:** Different keys for encryption and decryption, mainly used for **authentication** and **non-repudiation**
- **Hash functions:** Used for **authentication**

Symmetric encryption uses the same key for encryption and decryption. The key should be kept secret. An example is a **caesar cipher** (rot13).

There are a few different techniques for symmetric encryption:
- **Substitution:** replace each letter with another
- **Diffusion**: replace the order of the letters
- **Using a key:** use a key to encrypt the data, the method is known but the key is secret

Stream ciphers:

Stream ciphers encrypt one bit at a time. They work like this:
- Generate a key stream
- XOR the key stream with the plaintext
- XOR the key stream with the ciphertext to decrypt

The one-time pad

The one-time pad is a stream cipher that uses a key as long as the plaintext. This is unbreakable if the key is truly random and only used once. It's not practical because the key needs to be as long as the plaintext.

When the key is reused, the one-time pad is no longer secure. This is called the **key reuse problem**.

Block Ciphers

Block ciphers encrypt a block of data at a time. They use a fixed block size. An example is the **Data Encryption Standard** (DES).

DES:

DES is a block cipher that uses a 56-bit key. It uses a 64-bit block size. It uses a Feistel network with 16 rounds. DES is no longer considered secure because the key is too short.

2DES meet-in-the-middle attack

The DES algorithm can be used twice with two different keys. When you have the plaintext and the ciphertext, you can use a meet-in-the-middle attack to find the key. You can encrypt the plaintext with all possible keys and decrypt the ciphertext with all possible keys. You can then compare the results to find the keys.

AES:

The Advanced Encryption Standard is a block cipher that uses a 128-bit block size. It uses a 128, 192 or 256-bit key. It uses a substitution-permutation network with 10, 12 or 14 rounds.

Modes:

A mode is a way to use divide the plaintext into blocks and encrypt them. The ECB mode encrypts each block separately. This is not secure because identical blocks will have identical ciphertext. The CBC mode uses an IV (Initialization vector) to XOR the plaintext before encryption. Often the IV is the previous ciphertext block. CTR mode uses a counter to generate a key stream. This is parallelizable and less prone to errors.

Hash functions:

Hash functions take an input and produce a fixed-size output. They are used for integrity and authentication.

Hash functions should have a few properties:

- Deterministic: The same input should always produce the same output
- Fast: The hash should be fast to compute
- Pre-image resistance: It should be hard to find the input for a given hash
- Second pre-image resistance: It should be hard to find a different input for the same hash
- Collision resistance: It should be hard to find two inputs that produce the same hash

Hash functions without a secret key are used to assure data integrity. They allow to detect changes in the transmitted messages.

Hash functions can also be used for authentication (of data). In this case, a secret key is used, together with the message, and we call the function a Message Authentication Code (**MAC**).

Mac

A MAC is a hash function that uses a secret key. It can be used to verify the integrity of a message. The key should be kept secret.

Commonly used MACs are:

- HMAC: Hash-based MAC
- CMAC: Cipher-based MAC

Asymmetric Cryptography:

The idea is that everyone can encrypt a message with a public key, but only the person with the private key can decrypt it. This is used for authentication and non-repudiation.

Diffie-Hellman:

Agree on Public Values:
Alice and Bob publicly agree on two mathematical values:
- A large prime number (p) - This acts like a base number for calculations.
- A primitive root of p (g) - A special number that has specific mathematical properties with respect to p.

Private Key Generation:
- Alice: Chooses a random secret number (a) and keeps it private.
- Bob: Chooses a random secret number (b) and keeps it private.

Key Exchange (Public Communication):
- Alice: Calculates $A = g^a \pmod p$ and sends this value publicly to Bob.
- Bob: Calculates $B = g^b \pmod p$ and sends this value publicly to Alice.

Shared Secret Key Derivation (Private Calculation):
- Alice: Calculates the shared secret key using Bob's public value (B) and her private key (a): S = B^a (mod p).
- Bob: Calculates the same shared secret key using Alice's public value (A) and his private key (b): $S = A^b \pmod p$.

To prevent a MITM attack, Alice and Bob need to perform mutual authentication.

Digital signature:

A digital signature is a way to verify the authenticity of a message. It uses a private key to sign the message and a public key to verify the signature.

You as a sender can sign a message with your private key. The receiver can then verify the signature with your public key. This way, the receiver knows that the message is from you. Algorithms like **RSA** can be used for this.

How can we be sure that a public key belongs to a certain entity?

We can use a **Certificate Authority** (CA) to verify the identity of the entity. The CA signs the public key with its private key. The receiver can then verify the signature with the CA's public key. The CA is a trusted third party (TTP) that is trusted by the owner of the public key as well as other entities that use the public key.

Public-key encryption/decryption:

The sender can encrypt a message with the receiver's public key. The receiver can then decrypt the message with their private key. This way, only the receiver can read the message. Algorithms like **RSA** can be used for this.

Public-key algorithms require more computation time and are therefore slower than symmetric-key algorithms.

Digital signatures vs. public-key encryption:

- Digital signatures are used to verify the authenticity of a message.
  - **only the sender can sign, everyone can verify**
- Public-key encryption using public-key cryptography is used to encrypt a message.
  - **only the receiver can decrypt, everyone can encrypt**

Post-quantum cryptography:

Shor's algorithm allows to calculate discrete logarithms or to do integer factorization in polynomial time. This means that RSA and ECC are no longer secure. Post-quantum cryptography investigates how we can implement crypto after/if quantum computers become ubiquitous.

SSL-TLS:

- SSL and TLS are protocols that provide secure communication over the internet. They use a combination of symmetric and asymmetric encryption.
- The handshake is used to establish a secure connection.
- The server sends its certificate to the client.
- The client verifies the certificate with the CA.
- The client then generates a pre-master secret and sends it to the server.
- The server and client then generate the master secret from the pre-master secret.
- They then use the master secret to generate the session keys.

Web security:

SQL injection:

SQL injection is when an attacker injects SQL code into a query. This can be used to read or write data. You can prevent this by using prepared statements or by escaping user input.

```
e@m' OR 1=1); --
```

You can prevent this by verifying the input and using prepared statements. Delete suspicious characters. Replace problematic characters with safe ones:

- `'` → `\'`
- `"` → `\"`
- `;` → `\;`
- `-` → `\-`

Limit server privileges on the database and encrypt sensitive data.

Maintaining state:

HTTP is stateless. You can use cookies or sessions to maintain state. Cookies are stored on the client side and sessions are stored on the server side. Another way to maintain state is to use hidden fields in forms. This is prone to tampering.

Cookies:

- Cookies are stored on the client side
- They can be used to store session information
- They can be used to track users
- A stolen cookie can be used to **hijack a session**

Session Hijacking:

Holder of a session cookie can access the site with the privileges of an authenticated user. This can be done by stealing the session cookie.

You can get these cookies by:
- Sniffing the network (HTTPS)
- Compromising the server or the client (harden machines)
- Predicting the cookies (use random session IDs)
- Redirect the user to a malicious site (mark cookies as secure)

From the server side, you can:
- Set sensitive cookies to be very short-lived
- Invalidate the cookie when the user logs-out
- Make cookies different from session to session

CSRF

Cross-Site Request Forgery is when an attacker tricks a user into performing actions on a different site. This can be done by sending a link to the user. The goal is to issue a request to the server via the user's browser.

Protections:
- The browser will set the REFERER field indicating the page that hosted the clicked link.
- Adding a secret token to the form that is not known to the attacker.

Client-side:

Browsers can execute JavaScript. This can be used to manipulate the DOM. This can be used to steal cookies or to perform actions on behalf of the user.

XSS is an attack where an attacker injects JavaScript into a page. This can be used to steal cookies or to perform actions on behalf of the user.

Same Origin Policy
The same-origin policy is a security measure that prevents a script from one site from accessing data on another site. This is to prevent cross-site scripting attacks.

Cookies vs Tokens:

Cookies are:
- Stored on the client side
- Sent with every request
- Can be stolen with XSS
- Better when we need (backwards) compatibility

Tokens are:
- Client stores and transmits the user's claims and info
- Better for stateless client-server communications
- Better when cross-domain and cross-origin requests need to be enabled

Tracking:

There are a few ways to track users:

- Cookies: stored on the client side
- Browser fingerprinting (user-agent, screen size, etc.)
- Information (identity) trackers: (phone number, email, etc.)
- Tracking scripts (Google Analytics, etc.)

Other web security issues:

- Command injection: when an attacker injects commands into a system
- Directory traversal: when an attacker can access files outside of the web root
- File inclusion: when an attacker can include files from the server
- File upload: when an attacker can upload files to the server

Secure software development:

Principles:

- Least privilege: Only give the necessary permissions
- Verify input: Always verify input
- Use Safe Functions: Use safe functions like prepared statements
- Use safe libraries: Use libraries that are secure

Code reviews

Code reviews are a way to find bugs and vulnerabilities in the code. They can be done manually or automatically. Examples:

- Pull request reviews
- Pair programming
- Over the shoulder reviews
- Tool assisted reviews
- etc...

Mobile security:

API levels:

Android often releases new versions of the OS. These have new features, ans new API levels. Apps specify there minimum and target api.

Apps:

Apps have no single entry point or main function. Users interact with the GUI, the app interacts with the system APIs. These APIs are often event driven. An app consists of a few different components:

- **Activities:** A screen with a user interface
- **Service:** A background process
- **Content provider:** A way to share data between apps, database interface
- **Broadcast receiver:** receiver/dispatcher for *Intent* messages

All components can interact asynchronously through *Intents*. Intents can be explicit or implicit. Implicit intents are resolved by the system. App A may declare that it can handle actions (intents) of certain types. When another app wants to make this action (sends the implicit intent), the system will know it can count on A. The user may be offered to choose from a set of apps (including A). Think of opening a google maps url in the browser or google maps.

APK files
Apps are distributed as APK files. These are zip files with a specific structure. They contain the compiled code, resources, and a manifest file. The manifest file contains metadata about the app. These files are signed with a certificate.

Permissions:

There are a few different types of permissions:
Install-time permissions:
- Normal permissions: These are granted automatically (*internet access*)
- signature: granted to trusted packages (*system apps*)

Run-time permissions:
- dangerous— granted if the user approves at run-time (*notifications*)

Special permissions:
- granted by the user in the settings (*install unknown apps*)

Issues with permissions:
- Confused deputy attack: when an app uses permissions from another app
- Request more permissions than needed
- Users do not understand permissions
- Permissions evolve over time, which makes them confusing.

Advertising ID
The advertising ID is an alternative for the IMEI. It is used for advertising purposes. It can be reset by the user. It is used to track users across apps.

A lot of vulnerabilities come from interactions among apps.

Mobile malware:

- Rooting device
- Making premium SMS/calls
- Turning phone into bot
- Stealing private information
- Adware and click fraud
- Hindering usage: hijacking screens, redirecting, installing more apps
- Ransomware
- Cryptojacking

Detecting malware can be done by using *static* or *dynamic* analysis. Static analysis is when you analyze the code without running it. Dynamic analysis is when you run the code and analyze the behavior.

Access Control:

Controlling access on a system:

- **Identification:** Who are you?
- **Authentication:** Prove it
- **Authorization:** What can you do?
- **Accountability:** What did you do?

Identity management system:

An **identity management system** is a system that manages the identity of users. It can be used to manage access to resources. It can be used to manage user accounts, roles, and permissions.

- Can be tricky for large organizations
- Various security requirements:
  - Privacy/anonymity/full transparency
  - Strong link to physical identity
  - Revocability

Identity based authentication links 3 entities:

- **User:** a physical person or machine
- **Identity:** entry in an IMS to which a user can be associated
- **Subject:** A binary or a shell script that can run

Audit
An audit reviews the records of the system to check compliance with the security policy. It can be done by an internal or external auditor. It can be done manually or automatically.

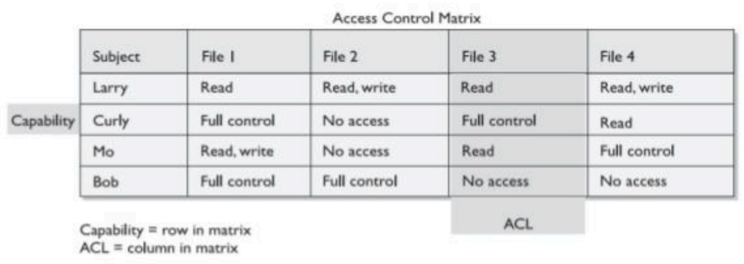
Access control policy: A policy that defines who can access what. It can be based on roles, attributes, or other factors.

Access control policies:

- **DAC:** Discretionary Access Control, the owner decides who can access the resource. *You can manage your own social media. but noy others*
- **MAC:** Mandatory Access Control, the system decides who can access the resource. *Military security, organization wide policies*
- **NAC:** Non-Discretionary Access Control, *admin decides who can access resources*
  - **RBAC:** Role-Based Access Control, access is based on roles. roles have access to objects, users have roles. *HR can access HR data, have HR role*
  - **ABAC:** Attribute-Based Access Control, access is based on attributes. *Only users with a certain clearance can access the data*

Access Control Lists (ACLs) vs. Capabilities:

ACLs: Access Control Lists are lists that define who can access a resource. They can be attached to the resource or the user. **Capabilities** are rules that define what a user can do. They can be attached to the user or the resource.



Access Control Matrix

Subject	File 1	File 2	File 3	File 4
Larry	Read	Read, write	Read	Read, write
Curly	Full control	No access	Full control	Read
Mo	Read, write	No access	Read	Full control
Bob	Full control	Full control	No access	No access

Capability
ACL

Capability = row in matrix
ACL = column in matrix

Figure 3: Access Control Matrix

An **Access Control Matrix** is a way to represent who can access what. It can be used to define access control policies.

Access Control Privileges in computer systems:

Subject classes:

- **Owner:** the creator/owner of the object
- **Group:** a group of users with the same permissions
- **World:** least privileged users

Privileges:

- **Read:** (r) read the object
- **Write:** (w) write to the object
- **Execute:** (x) execute the object

Separation of Duties:

Separation of duties is a security principle that states that no single person should have all the power. It can be used to prevent fraud and errors. It can be used to prevent conflicts of interest.

Example
At least 2 professors evaluate a bachelor thesis; a student cannot be a student assistant in a course they take.

Multi-level Security Policies:

- **The Bell-La Padula Model:** A model that defines how information can flow in a system. It has two rules: the simple security rule (no subject may read data from a higher level) and the *-property rule (no subject may write data to a lower level).
- **Biba model:** Data may be visible to many users, but can only be modified by a user with the right clearance. Has two rules: the simple integrity rule (no subject may write data to a higher level) and the *-integrity rule (no subject may read data from a lower level). *Everyone can read a website, but only admins can edit it*
- **Chinese wall model:** A model that prevents conflicts of interest. It prevents a user from accessing data that could create a conflict of interest. *A lawyer cannot work for two competing companies*