Security

Arnav Gupta

December 10, 2024

Contents

1 Security

The protection afforded to an automated information system in order to attain applicable objectives of preserving integrity, availability, and confi-

dentiality of info system resources.

Confidentiality: preventing unauthorized parties from accessing info or even being aware of the existence of info

Integrity: only authorized parties can manipulate info in authorized ways

Availability: resources accessible by authorized parties on all appropriate occasions

Must check before performing requested action:

- authenticity: is agent's claimed identity authentic or is someone masquerading as agent
- integrity: is the request the one the agent made or did someone tamper with it
- authorization: has proper authority granted permission to this agent to perform this action

1.1 Guard Model

Complete mediation: every request for the resource goes through the guard

Guard performs authentication (is principal who they claim to be) and authorization (does principal have access to performance request on resource).

Guard model can suffer from adversarial (targeted) attacks, where one successful attack can bring down a system.

Securing a system starts by specifying goals (policy) and assumptions (threat model).

Even though things can go wrong, systems that use guard model avoid common pitfalls.

1.2 Threats

Potential security violations caused either by a planned attack by an adversary or unintended mistakes by legitimate users.

3 broad categories of threats:

- **unauthorized info release**: unauthorized person can read and take advantage of info stored in computer or being transmitted over networks
  - extends to traffic analysis, where adversary observes only patterns of info use and from those patterns can infer info content
- **unauthorized info modification**: unauthorized person can make changes in stored info or modify messages that cross a network
  - adversary might engage in behaviour to sabotage system or trick receiver of message to divulge useful info or take unintended action
  - does not require that adversary be able to see info it changes
- **unauthorized denial of use**: adversary prevents authorized user from reading or modifying info, even though adversary cannot either

1.3 Goals and Principles

Narrow view of security can miss preventing some unauthorized actions.

A system's security can be insecure, secure, or unknown.

- to prove insecure, find security hole
- to prove secure means to show there can be no security hole
- typical outcome is unknown security holes exist

1.3.1 Design Principles

Least Privilege: give each component only the privileges it requires

Fail-safe Defaults: deny access if explicit permission is absent

Economy of Mechanism: adopt simple security mechanisms

Complete Mediation: ensure every access is permitted

Open Design: do not rely on secrecy for security

Separation of Privilege: introduce multiple parties to avoid exploitation of privileges

Least Common Mechanism: limit critical resource sharing to only a few mechanisms

Psychological Acceptability: make security mechanisms usable

Defense in Depth: have multiple layers of countermeasures

2 Authentication

Establish the following:

- who is the principal making the request?
  - guard must establish origin of the message
- is the request the one that the principal made
  - guard must establish integrity of the message

Steps to authenticate:

1. real-world person configures the guard
   (a) agreeing on a method by which the guard can later identify the principal identifier for the person
2. identity verification, occurs at later times using the agreed upon method

Verification methods:

- unique physical property of the user (faces, voices, fingerprints)
- unique thing the user has (ID)
- unique thing the user knows (password)

Most apps require users to set a password to reconnect to an account:

- salted hash of password is stored in DB
- passwords in future login can be compared to stored hash

Email address or phone number can also be used to prove identity.

Clients make requests to access user data and access must be protected.

HTTP requests should be sent using HTTPS (TLS): encrypts data in transit. Request/response data cannot be intercepted:

- TLS authenticates server using certificates
- TLS does not authenticate client

Can send password in each request, but storing the password locally is a security risk since all backend apps see the password (potentially in logs too). Adversary with access to server can see it.

Hash should be easy to compute, but it should be

- difficult to compute unhashed value from hashed value
- difficult to find another input with the same hashed value
- hashed value as short as possible, but long enough for low collision probability

Risk of **dictionary attack**: adversary compiles list of passwords and computes cryptographic has of these strings and compares result to value stored in computer system or uses computer program that tries login with these strings.

Using salt means that adversary would need separate rainbow table for every possible salt.

To avoid transmitting passwords repeatedly, server sends cookie which can be used to authenticate for some period of time.

Adversaries can create their own cookies, even with hash or server key.

Best to use session key:

- when user logs in, backend generates random session key, stores it in user account DB, and returns it to client
- client includes session key in all future requests, often in a header
- every request handler then checks auth token

Client device might send sign in request directly to auth service. Other microservices ask auth service to check auth tokens.

2.1 API Keys

Auth token that's valid for a long time. Used for 3rd parties to gain programmatic access to system. Can also be used for backend microservices to authenticate with each other, using local cache to check quickly without reading from DB.

API Key can be in query params, headers, or body. Headers are especially good since they are separate from request-specific params.

2.2 Phishing

To protect against phishing attacks, must avoid sending password to server directly but still allow valid servers to authenticate users.

Challenge response protocol: password never sent directly, instead it is hashed with random number

- adversary-owned servers will only ever learn hashed password, which the password cannot be recovered from

2.2.1 Two Factor Authentication

For added security, some services require more than a password, like a random challenge code (to email or SMS).

Passwords can be misused, so some services use email exclusively for login.

3 Authorization

Architecture Access Control Models: decide whether access to a protected resource should be granted or denied

Discretionary access control: based on requestor's identity, resource, and whether requestor has permission to access

Mandatory access control is policy based.

3.1 Ticket System

Each guard holds a separate ticket for each object it is guarding.

Principal holds a separate ticket for each different object it is authorized to use.

To authorize principal for access, authority gives principal a matching ticket for object. Principal can pass ticket to other principals.

To revoke principal's permissions, authority must hunt down principal and take ticket back or change guard's ticket and reissue tickets to any other authorized principals.

3.2 List System

Principal has token identifying principal and guard holds list of tokens that correspond to set of principals that authority has authorized.

To revoke access, authority removes principal's token from guard's list.

3.3 Mandatory Access Control

Every user has some access level and can access info at their level and below.

3.4 Known-Key Attacks

Adversary steals key to communicate or get info.

3.5 Replay Attacks

Same as known-key but with replaying the same session.

3.6 Misrepresentation

Make an authorized user seem unreliable.

3.7 Collusion

Same as misrepresentation, except done by multiple adversaries.

3.8 Fraudulent Actions

Adversary does not fulfill their part of a contract.

3.9 Addition of Unknowns

Unsure about safety of interacting with new entrants into the system.

3.10 Trust Management

Trust: level of subjective probability with which agent assesses that another agent will perform a particular action in a context that affects their actions

Reputation: expectation about entity's behaviour based on past behaviour, can be used to determine trust

2 types of trust management systems:

- credential and policy-based
- reputation-base