ECE 473/573
Cloud Computing and Cloud Native Systems
Lecture 15 Web Security

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

October 6, 2025

Outline

Web Security

JSON Web Token (JWT)

Reading Assignment

- This lecture: Web Security
- Next lecture: 7

Outline

Web Security

JSON Web Token (JWT)

HyperText Transfer Protocol Secure (HTTPS)

- A.k.a. HTTP over SSL or HTTP over TLS.
  - HTTP communication entirely on top of TLS (over TCP), usually use port 443.
  - Provide confidentiality and integrity.
  - Usually server authentication only, but client authentication could also be added.
- Domain name authentication
  - HTTPS server certificates need to include matching domain names and/or ip addresses for the connection to be considered secure by browsers.
  - Provide protection against IP address spoofing and DNS spoofing.
  - CA certificates can also be included with new browser installations – don't install browser from unknown sources!

Authentication for RESTful Services

- Usually, RESTful requests are protected by HTTPS with server authentication only, and clients are authenticated with other means like username and password.
- It is not practical and not secure to send those information for <u>every</u> RESTful request.
  - Users only expect to input those information once.
  - Keeping those information in the memory of user's computer increases the risk of them been stolen by other malicious processes.
- How can we authenticate the user once and preserve the authenticated user across multiple RESTful requests?

Cookie Based Authentication

- Cookie: a HTTP mechanism that allows HTTP servers to pass information back to HTTP clients.
  - As key-value pairs via the Set-Cookie HTTP response header.
  - Clients are required to send the cookie back for the following requests.
- Cookie based authentication
  - The first RESTful request would be a login request that contains user account name and password.
  - The server backend will authenticate the user and send back a cookie containing an randomly generated string, which is usually known as the access token or the session key.
  - All following requests will contain that access token to identify the user.

Threats for Cookie Based Authentication

- Quality of access token
  - Attackers may trick the server backend to use a known token.
  - Attackers may successfully guess the access token.
  - The access token should be generated randomly after each successful login and be long enough.
- Attackers may read the access token from HTTP packets.
  - HTTPS should be used to protect all HTTP communications.
- Attackers may steal the access token from user's computer.
  - The server backend should define when cookies expire and reject expired cookies.
  - The server backend should mark cookies as session cookies for browsers to make better protection.
  - The server backend could further check for unusual uses of access tokens, e.g. unusual ip addresses.

Cross-Site Scripting (XSS)

- Assume that there is a bug in a bank website that returns a web page containing whatever included in the HTTP request.
  - Actual bugs may be more subtle.
- An attacker, aware of the bug, create a HTTP/HTTPS request to the website containing a short JavaScript code.
  - It reads the access token and sends it back to a server controlled by the attacker.
- The attacker may trick a normal user to send the request.
- The browser then displays the returned web page and runs the malicious script.
  - The malicious script runs in a web page generated by the legitimate bank website so has full control over the access token.

XSS Mitigation

- Same-origin policy: scripts (running in a browser) are only allowed to access resources in the same website.
  - So the malicious script cannot simply send the stolen access token back to a different server controlled by the attacker.
- RESTful services: separate code and data
  - Server-side scriptings are more likely to have XSS issues as HTTP/HTTPS requests and responses are usually interpreted directly.
  - It will be less risky if browsers do not interpret RESTful responses as runnable scripts – still, efforts are required on both backend and frontend.

Outline

Web Security

JSON Web Token (JWT)

Motivation

- How to support cookie based authentication in microservices?
  - First, a login service authenticates the user, generates the access token, and returns it to the user as a cookie.
  - This login service should also manage a token database by adding newly generated ones and removing expired ones.
  - The other services can verify the access tokens using the database.
- An extra round-trip to the database is added when processing every RESTful requests.
  - Impact performance even if we use a distributed database to handle scalability and availability.
- Is it possible for individual services to verify access tokens without a database?

JSON Web Token (JWT)

- JSON Web Token (JWT): RFC 7519, 2015
  - A JSON message format that utilizes cryptography for protection.
  - A string format to encode the JSON message so it can be included into any URL.
- JWT strings work as access tokens for cookie based authentication.
  - Instead of being random, JWT allows to include structural information into access tokens.
  - Services use pre-shared secret keys or public/private key pairs to generate and verify JWT strings – no need to use a database.
  - Still, as this is still cookie based authentication, all previous discussed threats remain possible.

JWT Message and Token Structure

- Header: `{"alg": "RS256", "typ": "JWT"}`
  - `alg`: signature algorithm
  - `typ`: message type, usually `JWT`
- Payload: `{"sub": "admin", "iat": 1759276800, "exp": 1759280400}`
  - Include claims (data) to be signed.
  - `sub`: subject, unique user id.
  - `iat`: issued at time, when the JWT token is generated
  - `exp`: expiration time, when the JWT token should be rejected
  - Both `iat` and `exp` are in Unix time, i.e. number of seconds since 01/01/1970 00:00:00 UTC.
- Signed JWT token: `xxxxxx.yyyyyy.zzzzzz`
  - A string that can be included into any URL.
  - `xxxxxx`: Base64url encoding of header
  - `yyyyyy`: Base64url encoding of payload
  - `zzzzzz`: apply signature algorithm to `xxxxxx.yyyyyy` and then encode the result bytes using Base64url encoding.

JWT Signature Algorithms

- With symmetric cryptography, a MAC algorithm is used.
  - E.g. "HS256" – use HMAC algorithm with SHA-256 hash.
  - Need a pre-shared secret among all services.
  - Every service can both issue and verify JWT tokens – if one service is compromised then the whole system is compromised.
  - Good if there is only one service. Avoid if multiple services are involved.
- With public-key cryptography, a digital signature is used.
  - E.g. "RS256" – use RSA digital signature with SHA-256 hash.
  - Need to deploy public keys of services issuing tokens (e.g. the login service) to those verifying tokens (e.g. other services).
  - Services without private keys cannot issue tokens – as long as the login service is not compromised the system remains secure.
  - Implementations choose how exactly the public keys are deployed safely to prevent man-in-the-middle attacks, from pre-shared public keys to PKI setups.

Summary

- Cookie based authentication is widely used for web services. Therefore, it is critical to protect cookies for web security.
- JWT makes cookie based authentication scalable but does not prevent cookies to be stolen or misused.