CS144: Sessions

- HTTP is a stateless protocol. The server's response is purely based on the single request, not anything else.
- Q: How does a web site like Amazon can “remember” a user and customize its results? How can they implement “shopping cart”? How does it know that multiple HTTP requests are coming from the same user?
  - Q: Use source IP?

Cookie

- Cookies allow a server to ask a client to remember "name=value" pairs and send them back in all future requests.
- Example
  - From server (in the response):
    ```
    Set-Cookie: username=john; path=/; domain=ucla.edu; expires=Wed, 21 Oct 2031 07:28:00 GMT;
    ```
    - Request client to “set” the cookie `username=john`
    - `path` and `domain` specify the path and the domain to which the cookie should be sent
      - if not specified, the cookie will be sent in all future requests to this server
    - `expire` specifies when the cookie expires
      - if not specified, the cookie becomes “transient” (= session cookie) and is valid during current browsing session
      - server can “erase” a cookie by setting the expiration date to a past time
  - In all future requests to the specified domain and path, client add:
    ```
    Cookie: username=john
    ```
- **Same-origin policy:** The client sends a cookie only to the domain from which it got the cookie.
– No cross-domain cookie exchange is allowed.
– Q: Why same-origin policy?

– Q: Can we use cookie(s) to identify a user across multiple domains? It is possible given the same-origin policy?
  * third-party cookie

Authentication and session management

• Q: How can we authenticate a user? How do we verify that the user is really who they claim to be?

• Q: How can we let users authenticate once, without asking for authentication for every request?

• Q: After authentication, what should we store in the cookie?

• “username” vs “session ID”
  – Q: Any problem with storing username as cookie?

  – Session ID:
    * All session-related “states” reside on the server
A unique identifier is associated with a session
* Store the session ID in the cookie
* The server obtains session related “states” from local “session data store” using session ID

– Q: Why is it helpful? Can’t a malicious user send a different session ID?

– Q: Pros and Cons between signed states vs session ID

• Note: Be very careful about what we store in cookie
  – Cookie theft and cookie poisoning
  – secure; attribute
    * With secure; attribute set, the cookie is sent back only over https
    * Protects against cookie theft
  – Signed cookie:
    * Secret-key encrypted signature added to the main cookie data
    * Protects against cookie poisoning
  – Attaching expiration date
    * Makes sure that cookie useable only for a short period of time
    * Even if the cookie is stolen after a while, it is no longer valid

JSON Web Token (JWT)

• Web standard to represent and exchange client-managed states with protection against tempering
• Format: header.payload.signature
• Header: Base64-encoded JSON object, with (typically) two fields, \texttt{alg} (hashing algorithm) and \texttt{typ} (token type)

  – Example

  
  ```json
  {
    "alg": "HS256",
  }
  ```

Junghoo "John" Cho (cho@cs.ucla.edu)
JWT

```
"typ": "JWT"
}
```

- **Payload**: Base64-encoded JSON object to represent the main information
  - Example

```
{
  "iss": "http://oak.cs.ucla.edu",
  "jti": "3gxylhd",
  "iat": 11183763,
  "exp": 11253352,
  "user": "jungwoo"
}
```

  * "Registered claims (=fields)": `iss` (issuer), `jti` (JWT ID), `iat` (issued at, # seconds since 1970-01-01T00:00:00Z UTC), `exp` (expires at), `sub` (subject), `aud`(audience), ...
  * No claim is required

- **Signature**: Base64-encoded secret-key encrypted hash on `header.payload`
  - Example

```
HMACSHA256(  
  base64UrlEncode(header) + "." + base64UrlEncode(  
    payload),  
  "my secret password"
)
```

- **Example JWT**:

```
eyJ0eXAi0iJKV1QiLCJhbGciOiJIUzI1NiJ9 // header
.eyJrZXkiOiJ2YWwiLCJoYXQiOjE0MjI2MDU0NDV9 // payload
eUiabuiKv-8PYk2AkGY4Fb5KMWorYBLw261JPQD51M // signature
```

- JWT can be remembered by the browser either as a cookie or by JavaScript code in localStorage
References

- Cookie: RFC 6265
- JSON Web Token: https://jwt.io/introduction/