Authentication Methods in Modern Web Applications

WS - Seminar

Created by Luca Rinaldi

Agenda

- HTTP Authentication Framework
- Session Authentication Systems
- Token Authentication Systems
- Token vs Session
- Conclusions

HTTP Authentication Framework

Originally standardized in **rfc2617** by IETF(Internet Engineering Task Force) and than updated with:

- rfc7617 "The 'Basic' HTTP Authentication Scheme"
- rfc7616 "HTTP Digest Access Authentication"
- rfc6750 "The OAuth 2.0 Authorization Framework: Bearer Token Usage"

Basic HTTP Authentication *[rfc7617]*

A simple authentication system, in with the client send `user-id:password` encoded in Base64 in the `Authentication` header field

for example for user-id "Aladdin" and password "open sesame":

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

Digest HTTP Authentication *[rfc7616]*

It's a challenge response system, where the password is not transmitted in clear text.

The server send a random `nonce` and the client reply with `hash(user:password:nonce)`

```
Authorization: Digest username="Mufasa",
                      realm="http-auth@example.org",
                      uri="/dir/index.html",
                      algorithm=MD5,
                      nonce="7ypf/xlj9XXwfDPEoM4URrv/xwf94BcCAzFZH4GiTo0v",
                      nc=00000001,
                      cnonce="f2/wE4q74E6zIJEtWaHKaf5wv/H5QzzpXusqGemxURZJ",
                      qop=auth,
                      response="8ca523f5e9506fed4657c9700eebdbec",
                      opaque="FQhe/qaU925kfnzjCev0ciny7QMkPqMAFRtzCUYo5tdS"
```

Bearer Token *[rfc6750]*

A security token with the property that any party in possession of the token (a "bearer") can use.

Using a bearer token does not require a bearer to prove possession of cryptographic key material (proof-of-possession).

```
GET /resource HTTP/1.1
Host: server.example.com
Authorization: Bearer mF_9.B5f-4.1JqM
```

HTTP is stateless

We want to avoid to communicate at every request username and password.

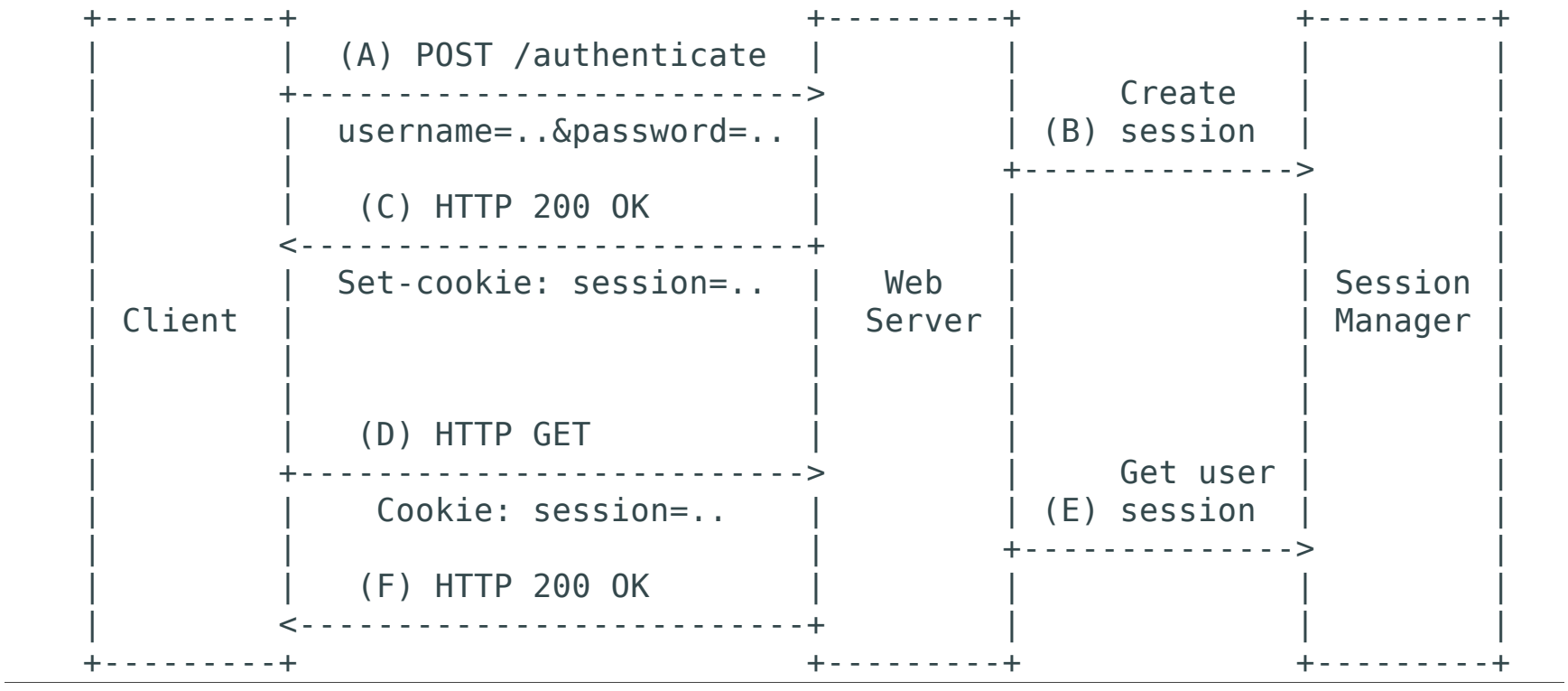
We want to save and retrieve specific data for every logged users.

Web Session Manager

With HTTP/1.1 and CGI programming language and framework start to implement Web Session Manager.

They maintain session with the users and identified it with a **sessionID**.

Web Session Flow



Web Session PHP example

```php
<?php
session_start();
if (!isset($_SESSION['user'])) {
    if ($_POST['username'] == 'luca' && $_POST['password'] == 'password'){
        $_SESSION['user'] = 'luca';
        $_SESSION['admin'] = true;
    }
} else {
  echo "username $_SESSION['user'] is logged ".
        ($_SESSION['admin'] ? "as admin" : "as user");
}
?>
```

Web Session security issue

- Session Hijacking thought:
  - observation
  - brute force
  - XSS
- CSRF (Cross-site request forgery), because it relies on cookies

Session limitation

- Centralize information
- Memory and cpu overhead
- Can't work with Cross Domain and CORS (Cross-origin resource sharing)
- Simple authentication flow

Tokens

It is a string that contain security credential to a login session.

It can be of two types:

- self-contained token
- opaque token

JSON Web Token *[rfc7519]*

It is a self-contained token with a set of keys/value pairs in JSON format.

It safeguard its integrity with JSON Web Signature (JWS) or JSON Web Encryption (JWE)

Am example of JWT:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
.
eyJpc3MiOiJsdWNhci5pbiIsImV4cCI6MTQ2OTI2ODcwOSwibmFtZSI6Imx1Y2EiLCJhZG1pbiI6dHJ1
.
5Z5tKUacfE-r_L56uaddeimgREpgk39Fbx6EJ3cuTJg
```

JWT Structure

Header:

```
{
    "alg": "HS256",
    "typ": "JWT"
}
```

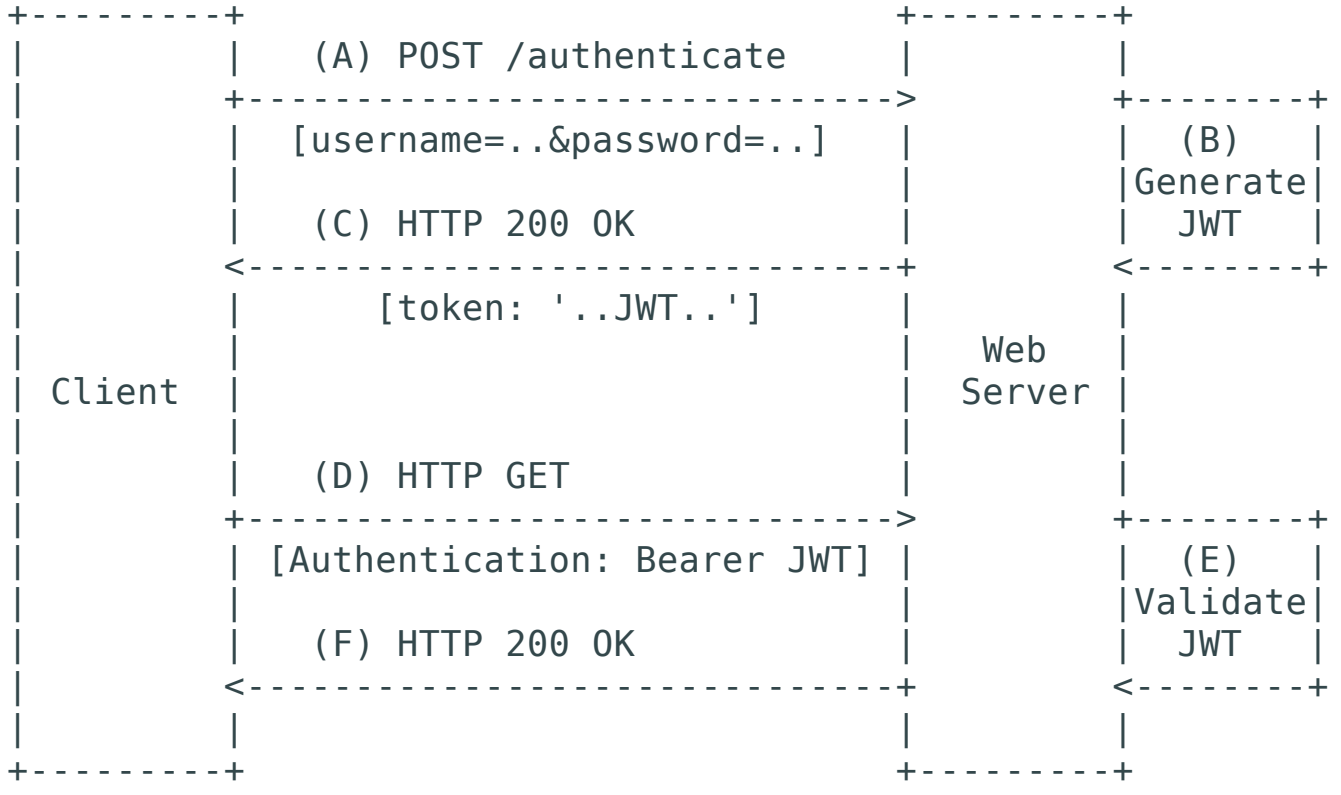
Payload:

```
{
    "iss": "lucar.in",
    "exp": 1469268709,
    "name": "luca",
    "admin": true,
}
```

Verify signature:

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

JWT Authentication Flow



OAuth 2.0 *[rfc6749]*

It enables a third-party application to obtain limited access to a service in behalf of a user.

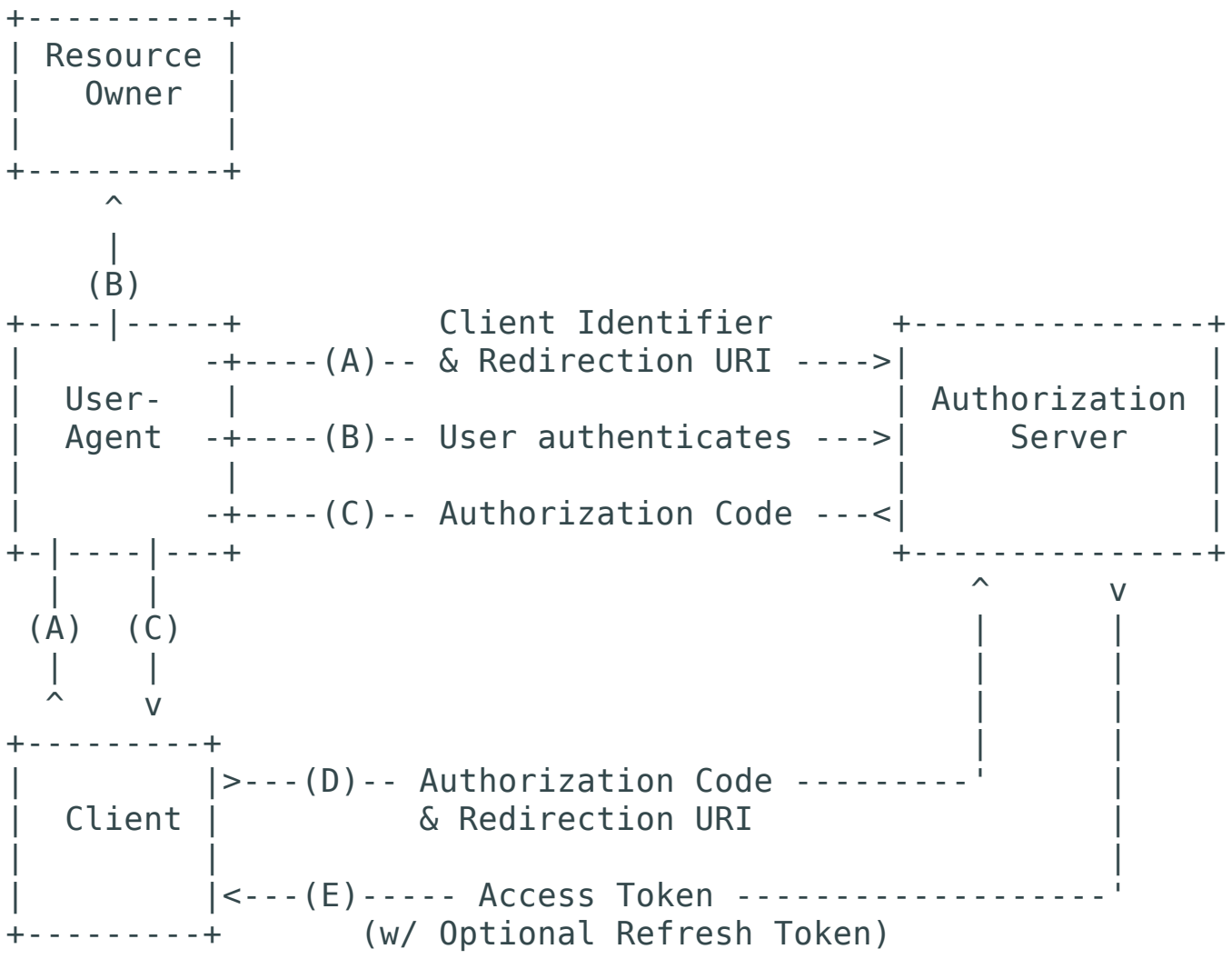
Three parties:

- resource owner (end-user)
- client (third-party application)
- resource and authorization server (service)

For example:

Draw.io, an online flow chart editor, that request the user to access their storage space on Dropbox, to save and load files.

OAuth Authentication Flow



Token Security issue

- XSS, it's possible to steal saved token
- Invalidation system prone to error
- They can contain sensible information

Why use Token

- Scalable
- Efficient (memory and CPU)
- CSRF immune
- Work with Cross Domain and CORS (Cross-origin resource sharing)
- Mobile ready

Conclusions

If correctly implemented either web session or token system can have strong security.

The type of authentication system really dependence on the goal of the project.

But token mechanisms are more general and ready for mobile and modern web application.

References

- RFC 7617 - **The 'Basic' HTTP Authentication Scheme** - IETF. (2015, September). Retrieved July 23, 2016, from https://tools.ietf.org/html/rfc7617
- RFC 7616 - **HTTP Digest Access Authentication** - IETF. (2015, September). Retrieved July 23, 2016, from https://tools.ietf.org/html/rfc7616
- RFC 6750 - **The OAuth 2.0 Authorization Framework: Bearer Token Usage** - IETF. (2012, October). Retrieved July 23, 2016, from https://tools.ietf.org/html/rfc6750
- **Web Based Session Management** - TechnicalInfo. (n.d.). Retrieved July 23, 2016, from http://technicalinfo.net/papers/WebBasedSessionManagement.html
- **Session Management Cheat Sheet** - OWASP. (2016, June 1). Retrieved July 23, 2016, from https://www.owasp.org/index.php/Session_Management_Cheat_Sheet
- **Session hijacking attack** - OWASP. (2014, August 14). Retrieved July 23, 2016, from https://www.owasp.org/index.php/Session_hijacking_attack
- RFC 7519 - **JSON Web Token (JWT)** - IETF. (2015, May). Retrieved July 23, 2016, from https://tools.ietf.org/html/rfc7519
- **Critical vulnerabilities in JSON Web Token libraries** - Auth0. (2015, March 31). Retrieved July 23, 2016, from https://auth0.com/blog/2015/03/31/critical-vulnerabilities-in-json-web-token-libraries/
- RFC 6749 - **The OAuth 2.0 Authorization Framework** - IETF. (2012, October). Retrieved July 23, 2016, from https://tools.ietf.org/html/rfc6749
- **Cookies vs. Tokens: The Definitive Guide** - DZone Integration. (2016, June 2). Retrieved July 23, 2016, from https://dzone.com/articles/cookies-vs-tokens-the-definitive-guide