Web Authentication

Thierry Sans

Several Methods

- **Local** authentication with login and password
- **Token-based** authentication
- **Third party** authentication

Local Authentication

How to store and verify password?

- ~~Clear~~

Data can be hacked

- ~~Encrypted~~

A key is needed to store
and verify passwords

- ~~Hash~~

Weak passwords have known hash

- Salted Hash

Salt and hash must be stored

Basic Authentication (stateless)

(Standard) RFC 2617

- Login and password are sent in **clear** (Base64 encoding) in **the headers** `"authorization"`

```
$ curl -u login:password http://url

$ curl http://admin:password@url
```

Session Authentication (stateful)

(Standard) RFC 6265

1. The user **enters a login and password** and the frontend send them to the backend (POST request)
2. The backend **verifies the login/password** based on information stored on the server (usually in the database)
3. The backend **stores user information in a session**
4. The backend **grants access to resources** based on the information contained in the session

Do/Don't with passwords

- On the client side, do send passwords in:
  - ✓ headers (automatic with basic authentication)
  - ✓ body (POST request with session authentication)
  - ◉ never in the URL
- On the server, do store passwords as
  - ✓ salted hash passwords only
  - ◉ never in clear

Token-based Authentication

HMAC

(Standard) RFC 2104

For each authenticated HTTP request,
the frontend **computes and send a message digest that combines the user's secret and some request arguments**

- ✓ User's password never transit back and forth (except the first time it is exchanged maybe)
- ✓ Digest can be send in clear

JSON Web Token

(Standard) RFC 7519

Encode user information in a string that is URL safe (token)

Token are authenticated and often encrypted

✓ Web token can be used for stateful but yet session-less authentication

Third-party Authentication

Single-Sign-On (SSO)

- **Pubcookie** (a.k.a webiso) 1998
- **OpenID** 2005
- **SAML** (Security Assertion Markup Language) 2005
- **OAuth** 2010
- **Mozilla Persona** 2011

among others …

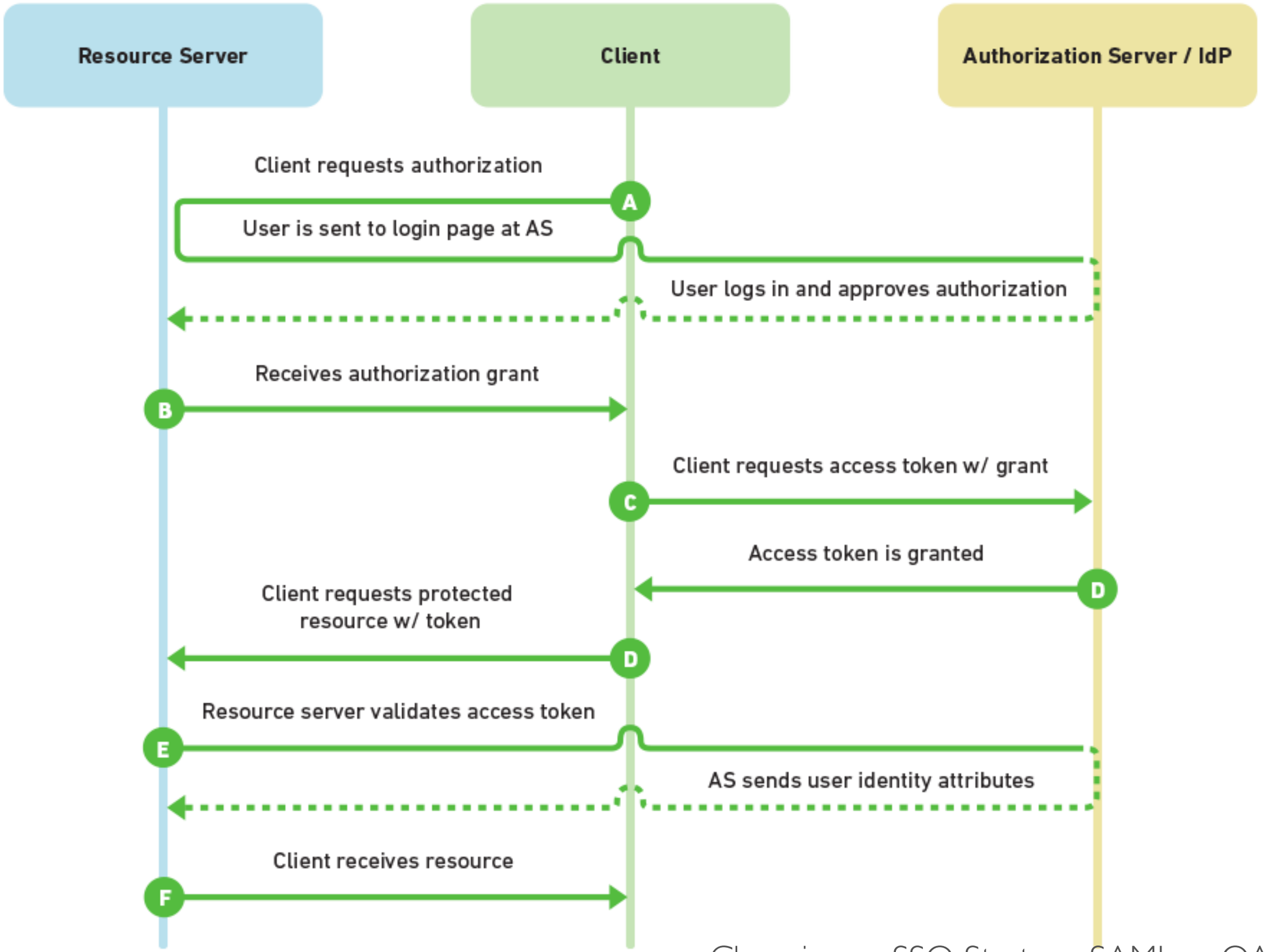
OAuth 2

(Standard) RFC 6749

1. The backend **redirects the user to the third-party login-page**
2. Third-party **asks and verify the login/password** based on the third-party user information
3. Third party **redirects the user back to the application** with a OAuth token and verifier in the url
4. Backend **verifies the token** with third party
5. Backend **starts a session**

➡ User's login/password never transit by the application frontend nor backend

OAuth 2.0 Flow



source: Choosing an SSO Strategy: SAML vs OAuth2