Cookies, Tea and err . . . Express Session

Jogesh K. Muppala

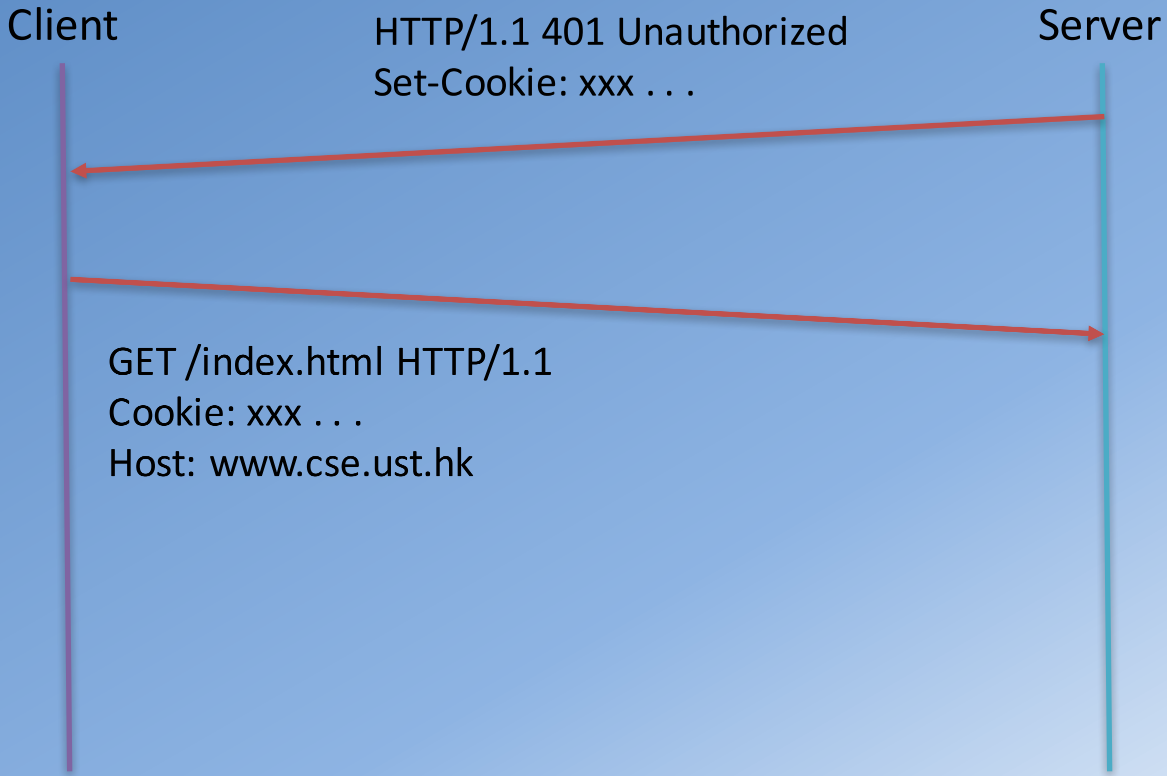
THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

HTTP Cookies

- Small piece of data sent from a web server and stored on the client side
- Each subsequent request from the client side should include the *cookie* in the request header

Cookies



HTTP/1.1 401 Unauthorized
Set-Cookie: xxx . . .

GET /index.html HTTP/1.1
Cookie: xxx . . .
Host: www.cse.ust.hk

Express and Cookies

- Server can set a cookie as follows in any of the middleware:

```
res.cookie(name,value,options)
```

- Cookies are parsed in Express server using the cookie-parser middleware

```
var cookieParser = require(‘cookie-parser’);
app.use(cookieParser());
```

- Cookie-parser parses incoming cookies and attaches them to request

```
req.cookies.name
```

Express and Signed Cookies

- Signed cookie: signed with a secret key on the server side
  - Digital signature with key-hash message authentication code (verifiable)
- Cookie parser supports signed cookies:

```
var cookieParser = require('cookie-parser');
app.use(cookieParser('secret key'));
```

- Parsed signed cookies made available as:

```
req.signedCookies.name
```

Express Sessions

- Used to track user sessions
  - Combination of cookie with session id and server-side storage of information indexed by session id
  - Session information:
    - Stored by default in-memory (wiped out when server restarts)
    - Stored in permanent store on server side
    - Distributed session store if using multiple replicated servers

express-session Middleware

```
var session = require('express-session');
var FileStore = require('session-file-store')(session);

app.use(session({
  name: 'session-id',
  secret: '12345-67890-09876-54321',
  saveUninitialized: true,
  resave: true,
  store: new FileStore()
}));
```

- Express session information available as:
  req.session.name

Express Session

- Different properties of the sessions options object:
  - **cookie**: Options object for the session ID cookie. The default value is { path: '/', httpOnly: true, secure: false, maxAge: null }.
  - **genid**: Function to generate the session ID. Default is to use uuid
  - **name**:The name of the session ID cookie to set in the response (and read from in the request).
  - **proxy**: Trust the reverse proxy when setting secure cookies.
  - **resave**: If true forces a session to be saved back to store even if it was not modified in the request.
  - **rolling**: Forces a cookie to be set on every request.
  - **saveUninitialized**: If true it forces a newly created session without any modifications to be saved to the session store.
  - **secret**: It is a required option and is used for signing the session ID cookie.
  - **store**: Session store instance. Default is to use memory store.
  - **unset**: Controls the handling of session object in the store after it is unset. Either delete orkeep the session object. Default is to keep the session object

Exercise: Using Cookies

- Set up your Express application to send signed cookies.
- Set up your Express application to parse the cookies in the header of the incoming request messages

Exercise: Express Sessions

- Set up your Express server to use Express sessions to track authenticated users
- Enable clients to access secure resources on the server after authentication.