CS 88: Security and Privacy

08: Web Security: HTTP and Cookies

02-15-2024

slides adapted from Dave Levine, Vitaly Shmatikov, Christo Wilson
SQL Injection

Can chain together statements, and can modify existing statements
Not Just SQL!

User

Front end

Forms a string containing user input

Back end

Executes this string as a command or query

Database
NoSQL storage
Javascript
eval(…)

Injection vulnerabilities are a generic issue!
PREVENTING INJECTION ATTACKS

Validate all the inputs!
Most injection attacks trick application into interpreting data as code.

This changes the semantics of a query or command generated by the application.

Make sure unsafe inputs cannot change the meaning of query.
A basic web architecture

Much of the user data is part of the browser

DB is a separate entity, logically (and often physically)
Where Does the Attacker Live?

Much of the user data is part of the browser

DB is a separate entity, logically (and often physically)
Web Architecture: Simplified View

Client Side

- Document Renderer
- HTML Parser

Protocols

- Gopher
- FTP
- HTTP

Server Side

Web servers: Responsible for securely parsing input data
- PHP, Ruby, ASP, JSP

Web Browser
Responsible for securely confining Web content presented by visited websites
Web Applications

• Big trend: software as a Web-based service
  • Online banking, shopping, government, bill payment, tax prep, customer relationship management, etc.
  • Cloud-hosted applications

• Application code split between client and server
  • Client (Web browser): JavaScript
  • Server: PHP, Ruby, Java, Perl, ASP …

• Security is rarely the main concern
  • Poorly written scripts with inadequate input validation
  • Inadequate protection of sensitive data
Top Web Vulnerabilities

• SQL injection
  • Malicious data sent to a website is interpreted as code in a query to the website’s back-end database

• XSRF (CSRF) - cross-site request forgery
  • Bad website forces the user’s browser to send a request to a good website

• XSS (CSS) – cross-site scripting
  • Malicious code injected into a trusted context (e.g., malicious data presented by a trusted website interpreted as code by the user’s browser)
Overview

• The Web Model
  • What components make up today’s browsers and web servers?
  • How has this functionality evolved over time?
  • What security model governs the browser?
Overview: The Web Model

• What is the web?
• What components make up today’s browsers and web servers?
• How has this functionality evolved over time?
• What security model governs the web browser?
What is the web?

• **Web (World Wide Web)**: A collection of data and services
  • Data and services are provided by **web servers**
  • Data and services are accessed using **web browsers** (e.g. Chrome, Firefox)
• The web is not the Internet
  • The Internet describes *how* data is transported between servers and browsers
Elements of the Web

• **URLs**: How do we uniquely identify a piece of data on the web?
• **HTTP**: How do web browsers communicate with web servers?

• Data on the webpage can contain:
  • **HTML**: A markup language for static webpages
  • **CSS**: A style sheet language for defining the appearance of webpages
  • **Javascript**: a programming language for running code in the web browser
Elements of the Web

• Data on the webpage can contain:
  • **HTML**: A markup language for **static** webpages
  • **CSS**: A style sheet language for defining the appearance of webpages
  • **Javascript**: a programming language for running code in the web browser
What IS A Web Browser?
Web Browser: Basic Execution Model

• Each browser window or frame:
  • Loads content
  • Renders:
    • Processes HTML and scripts to display the page
    • May involve images, subframes, etc.
  • Responds to events

• Events
  • User actions: OnClick, OnMouseover
  • Rendering: OnLoad, OnUnLoad
  • Timing: setTimeout(), clearTimeout()
Generating a static webpage: HTML
<!doctype html>
<html>
<head>
  <title>Hello World</title>
</head>
<body>
  <h1>Hello World</h1>
  <img src="/img/my_face.jpg"></img>
  <p>
    I am 12 and what is this?
  </p>
  <img src="http://www.images.com/cats/adorablekitten.jpg"></img>
</body>
</html>

HTML may embed other resources from the same origin

... or from other origins (cross origin embedding)
JavaScript

• **Language executed by the browser**
  - Scripts are embedded in Web pages

• Inline
  - `<a onclick="doSomething();"></a>`

• Embedded
  - `<script>alert('Hello');</script>`

• External
  - `<script src="/js/main.js"></script>`

• Potentially malicious website gets to execute some code on user’s machine

“Java is to JavaScript as car is to carpet”
Event-Driven Script Execution

```javascript
function whichButton(event) {
    if (event.button==1) {
        alert("You clicked the left mouse button!")
    } else {
        alert("You clicked the right mouse button!")
    }
}
</script>

<body onmousedown="whichButton(event)">
...  
</body>
```

Script defines a page-specific function

Function gets executed when some event happens
Elements of the Web

- **URLs**: How do we uniquely identify a piece of data on the web?
- **HTTP**: How do web browsers communicate with web servers?
Interacting with web servers

Protocol: ftp
           https
           tor

Hostname/server
• translated to an IP address by DNS

Path to the resource
index.html is static content
i.e., a fixed file returned by the server
Interacting with web servers

http://www.cs.swarthmore.edu/~chaganti/index.html

Protocol: ftp https tor

Hostname/server translated to an IP address by DNS

Path to the resource
index.html is **static content**
  i.e., a fixed file returned by the server

http://facebook.com/ delete.php

Path to the resource
delete.php is **dynamic content**
  i.e., a server generates the content on the fly
Interacting with web servers: dynamic content

http://facebook.com/delete.php Path to the resource


server generates the content on the fly
URL Escaping


• URLs are designed to contain printable, human-readable characters (ASCII)
  • include non-printable characters in the URL?
• URLs have special characters that have assigned meaning (?, #, /)
URL Escaping


• What if we want to use a special character in the URL?
  • Solution: URL encoding
  • Notation: Percent sign (%) followed by the hexadecimal value of the character
  • Example: %20 = ' ' (spacebar) %35 = '#' (hash sign)
    %50 = '2' (printable characters can be encoded too!)

• Security issues: makes scanning for malicious URLs harder
  • Suppose you want to block all requests to the path /etc/passwd
  • What if an attacker makes a request to %2F%65%74%63%2F%70%61%73%77%64?
HTTP and the Web

First, a review...

- web page consists of objects
- object can be HTML file, JPEG image, Java applet, audio file,…
- web page consists of base HTML-file which includes several referenced objects
- each object is addressable by a URL, e.g.,

```
http://www.cs.swarthmore.edu/~chaganti/index.html
```

Protocol:  
- ftp  
- https  
- tor

Hostname/server:  
- translated to an IP address by DNS

Path to the resource
HTTP: Hypertext transfer protocol

- client/server model
  - **client**: browser that requests, receives, (using HTTP protocol) and “displays” Web objects
  - **server**: Web server sends (using HTTP protocol) objects in response to requests
HTTP Overview

1. User types in a URL.
   http://some.host.name.tld/directory/name/file.ext

   host name
   path name
2. Browser establishes connection with server. Looks up “some.host.name.tld” connects
3. Browser requests the corresponding data.
   GET /directory/name/file.ext HTTP/1.0
   Host: some.host.name.tld
   [other optional fields, for example:]
   User-agent: Mozilla/5.0 (Windows NT 6.1; WOW64)
   Accept-language: en
HTTP Overview

4. Server responds with the requested data.
   HTTP/1.0 200 OK
   Content-Type: text/html
   Content-Length: 1299
   Date: Sun, 01 Sep 2013 21:26:38 GMT
   
   (Data data data data data...)
HTTP Request Header

GET / HTTP/1.1
Host: www.reddit.com
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,...
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2)...
DNT: 1
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Cookie: reddit_session=...
HTTP Overview

5. Browser renders the response, fetches any additional objects, and closes the connection.
HTTP Response Header

HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Cache-Control: no-cache
Pragma: no-cache
x-frame-options: SAMEORIGIN
x-content-type-options: nosniff
x-xss-protection: 1; mode=block
Vary: accept-encoding
Content-Encoding: gzip
Content-Length: 24824
Server: '; DROP TABLE servertypes; '--
Date: Mon, 10 Mar 2014 22:44:23 GMT
Connection: keep-alive

Cache directives
Anti-framing
Disable content sniffing
Enable anti-XSS filter
More cache directives

Server version
Status code
Status message
Resource MIME type, charset
Content encoding
Content length
Connection type
Timestamp
Example

GET / HTTP/1.1
Host: demo.cs.swarthmore.edu

HTTP/1.1 200 OK
Vary: Accept-Encoding
Content-Type: text/html
Accept-Ranges: bytes
ETag: "316912886"
Last-Modified: Wed, 04 Jan 2017 17:47:31 GMT
Content-Length: 1062
Date: Wed, 05 Sep 2018 17:27:34 GMT
Server: lighttpd/1.4.35

Response headers

Response Body
Example

GET / HTTP/1.1
Host: demo.cs.swarthmore.edu

<html><head><title>Demo Server</title></head><body>
....
</body></html>

Response Headers

Response Body
Anatomy of Request

HTTP Request

- **method**: GET
- **path**: /index.html
- **version**: HTTP/1.1

**headers**:
- Accept: image/gif, image/x-bitmap, image/jpeg, */*
- Accept-Language: en
- Connection: Keep-Alive
- User-Agent: Mozilla/1.22 (compatible; MSIE 2.0; Windows 95)
- Host: www.example.com
- Referer: http://www.google.com?q=dingbats

**body** (empty)
HTTP Response

HTTP/1.0 200 OK

Date: Sun, 21 Apr 1996 02:20:42 GMT
Server: Microsoft-Internet-Information-Server/5.0
Content-Type: text/html
Last-Modified: Thu, 18 Apr 1996 17:39:05 GMT
Content-Length: 2543
Set-Cookie: aldkfj2314

<html>Some data... announcement! ... </html>
HTTP Methods

GET: Get the resource at the specified URL (does not accept message body)

POST: Create new resource at URL with payload

PUT: Replace target resource with request payload

PATCH: Update part of the resource

DELETE: Delete the specified URL
HTTP Methods

Not all methods are created equal — some have different security protections.

**GETs** should not change server state; in practice, some servers do perform side effects:

- Old browsers don’t support **PUT**, **PATCH**, and **DELETE**
- Most requests with a side effect are **POSTs** today
- Real method hidden in a header or request body

Never do...

Get

```
http://bank.com/transfer?fromAcct=X&toAcct=Y&amount=1000
```
Goals of Web Security: Safely Browse the Web

• Safe to visit an evil website
  • sandboxing Javascript
  • privilege separation

• Safe to visit two pages at the same time,
  • same-origin policy

• Safe delegation
Web Security Model

**Subjects**

“Origins” — a unique $\textit{scheme://domain:port}$

**Objects**

DOM tree, DOM storage, cookies, javascript namespace, HW permission

**Same Origin Policy (SOP)**

**Goal:** Isolate content of different origins

- **Confidentiality:** script on evil.com should not be able to \textit{read} bank.ch
- **Integrity:** evil.com should not be able to \textit{modify} the content of bank.ch
Same Origin Policy

• rule that prevents one website from tampering with other unrelated websites.
  • enforced by browser
Every webpage has an **origin** defined by its URL with three parts:

- **Protocol**: The protocol in the URL
- **Domain**: The domain in the URL’s location
- **Port**: The port in the URL’s location
  - If no port is specified, the default is 80 for HTTP and 443 for HTTPS

- **https://cs.swarthmore.edu:443/assets/lock.PNG**
- **http://cs.swarthmore.edu/assets/images/404.png**
- **80 (default port)**
Bounding Origins — Windows

Every Window and Frame has an origin.
Origins are blocked from accessing other origin’s objects

attacker.com cannot...
- read or write content from bank.com tab
- read or write bank.com's cookies
- detect that the other tab has bank.com loaded
Beyond loading individual resources, websites can also load other *websites* within their window

- Frame: rigid visible division
- iFrame: floating inline frame

Allows delegating screen area to content from another source (e.g., ad)
Bounding Origins — Frames

Every Window and Frame has an origin
Origins are blocked from accessing other origin’s objects

**attacker.com** cannot...
- read content from **bank.com** frame
- access **bank.com**'s cookies
- detect that has **bank.com** loaded
Same-Origin Policy

- Two webpages have the same origin if and only if the protocol, domain, and port of the URL all match exactly:
  - String matching:
    - The protocol, domain, and port strings must be equal

<table>
<thead>
<tr>
<th>First domain</th>
<th>Second domain</th>
<th>Same origin?</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://cs88.swat.org">http://cs88.swat.org</a></td>
<td><a href="https://cs88.swat.org">https://cs88.swat.org</a></td>
<td></td>
</tr>
<tr>
<td><a href="http://cs88.swat.org">http://cs88.swat.org</a></td>
<td><a href="http://swat.org">http://swat.org</a></td>
<td></td>
</tr>
</tbody>
</table>
Same-Origin Policy: Two websites with different origins can’t interact with each other.

Example: If cs88.org embeds google.com, the inner frame cannot interact with the outer frame, and the outer frame cannot interact with the inner-frame.

So what happens when...

1. JavaScript runs with the origin of the page that loads it? E.g., cs88.org fetches Javascript from Google analytics.
2. Websites fetch and display images from other origins? E.g. if we include `<img src="http://google.com/logo.jpg" on http://cs88.org`, the image has origin http://google.com.
3. We load frames such as `<iframe src="http://google.com"></iframe> on cs88.org?
Same-Origin Policy

- Two websites with different origins cannot interact with each other
  - Example: If cs88.org embeds google.com, the inner frame cannot interact with the outer frame, and the outer frame cannot interact with the inner-frame
- Exception: JavaScript runs with the origin of the page that loads it
  - Example: If cs88.org fetches JavaScript from google.com, the JavaScript has the origin of cs88.org
  - Intuition: cs88.org has “copy-pasted” JavaScript onto its webpage
- Exception: Websites can fetch and display images from other origins
  - However, the website only knows about the image’s size and dimensions (cannot actually manipulate the image)
- Exception: Websites can agree to allow some limited sharing
  - Cross-origin resource sharing (CORS)
  - The postMessage function in JavaScript
Same-Origin Policy: Summary

- Rule enforced by the browser: Two websites with different origins cannot interact with each other.
- Two webpages have the same origin if and only if the protocol, domain, and port of the URL all match exactly (string matching).
- Exceptions
  - JavaScript runs with the origin of the page that loads it.
  - Websites can fetch and display images from other origins.
  - Websites can agree to allow some limited sharing.
State(less)

(XKCD #869, “Server Attention Span”)
State(less)

• **Original web**: simple document retrieval

• **Maintain State?** Server is not required to keep state between connections
  
  ...often it might want to though

• **Authentication**: Client is not required to identify itself
  
  • server might refuse to talk otherwise though
• Server stores state, indexes it with a cookie

• Send this cookie to the client

• Client stores the cookie and returns it with subsequent queries to that same server
Browser Cookie Management

• **Cookie Same-origin ownership**
  - Once a cookie is saved on your computer, only the Web site that created the cookie can read it.

• **Variations**
  - Temporary cookies
    - Stored until you quit your browser
  - Persistent cookies
    - Remain until deleted or expire
  - Third-party cookies
    - Originates on or sent to a web site other than the one that provided the current page
Third-party cookies

• Get a page from merchant.com
  • Contains `<img src=http://doubleclick.com/advt.gif>`
  • Image fetched from DoubleClick.com
    • DoubleClick knows IP address and page you were looking at
• DoubleClick sends back a suitable advertisement
  • Stores a cookie that identifies "you" at DoubleClick
• Next time you get page with a doubleclick.com image
  • Your DoubleClick cookie is sent back to DoubleClick
  • DoubleClick could maintain the set of sites you viewed
  • Send back targeted advertising (and a new cookie)
• Cooperating sites
  • Can pass information to DoubleClick in URL, …
Cookie issues

• Cookies maintain record of your browsing habits
  • Cookie stores information as set of name/value pairs
  • May include *any* information a web site knows about you
  • Sites track your activity from multiple visits to site
• Sites can share this information (e.g., DoubleClick)
• Browser attacks could invade your “privacy”
Browser Fingerprinting

- Browser sends HTTP head information, which includes
  - User agent: e.g., “Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/38.0.2125.111 Safari/537.36”
  - HTTP header: e.g., “text/html, */* gzip, deflate en-US, en;q=0.8”
  - Javascript can collect font information, installed browser-plugin information
  - Using canvas, e.g., how to render emoji
  - Can achieve high entropy.
  - Can be used to track users/browsers.

- https://panopticlick.eff.org/
Cookies are key-value pairs

Set-Cookie: key=value; options; ....

HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 08:20:34 GMT
Server: Apache
Set-Cookie: session-zdnet-production=6bhqcal10cbscaigu1sisac2p3; path=/; domain=zdnet.com
Set-Cookie: zdregion=MT15lJuMT15lJE1Mzp1czp1czp1ZDjmNWY5YTk0ODU1N2QYz5MNGU3MY1ZTRmN6
Set-Cookie: zdregion=MT15lJuMT15lJE1Mzp1czp1czp1ZDjmNWY5YTk0ODU1N2QYz5MNGU3MY1ZTRmN6
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=zdnet.com
Set-Cookie: session-zdnet-production=59ob97fipq4b6ide4dvvq11; path=/; domain=zdnet.com
Set-Cookie: user_agent=desktop
Set-Cookie: zdnets_ad_session=f
Set-Cookie: firstpg=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
X-UA-Compatible: IE=edge,chrome=1
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 18922
Keep-Alive: timeout=70, max=146
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<html> ...... </html>
Cookies are key-value pairs

Set-Cookie: key=value; options; ....

HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 08:20:34 GMT
Server: Apache
Set-Cookie: session-zdnet-production=6bhqca1i0cbciagu11sisac2p3; path=/; domain=zdnet.com
Set-Cookie: zdregion=MTI5LjJuMTI5LjE1Mzplczp1czp1czp1ZDjmNWW5YTdkODU1N2QYzM5NGU3M2Y1ZTRmNGQ1
Set-Cookie: zdregion=MTI5LjJuMTI5LjE1Mzplczp1czp1czp1ZDjmNWW5YTdkODU1N2QYzM5NGU3M2Y1ZTRmNGQ1
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com
Set-Cookie: session-zdnet-production=59ob97fpinqe4bg6lde4dvq11; path=/; domain=zdnet.com
Set-Cookie: user_agent=desktop
Set-Cookie: zdnet_ad_session=f
Set-Cookie: firsttpg=0
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
X-UA-Compatible: IE=edge,chrome=1
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 18922
Keep-Alive: timeout=70, max=146
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<html> ...... </html>
Cookies

Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com

Client

Browser

(Private) Data

Semantics

- Store “us” under the key “edition” (think of it like one big hash table)
Cookies

Set-Cookie: edition=us expires=Wed, 18-Feb-2015 08:20:34 GMT path=/; domain=.zdnet.com

Client

Browser

(Private) Data

Semantics

• Store “us” under the key “edition” (think of it like one big hash table)

• This value is no good as of Wed Feb 18…
Cookies

Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=zdnet.com

Semantics

- Store "us" under the key "edition" (think of it like one big hash table)
- This value is no good as of Wed Feb 18...
- This value should only be readable by any domain ending in .zdnet.com
Cookies

Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com

Semantics

- Store “us” under the key “edition” (think of it like one big hash table)
- This value is no good as of Wed Feb 18...
- This value should only be readable by any domain ending in .zdnet.com
- This should be available to any resource within a subdirectory of /
Cookies

Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com

Semantics
- Store “us” under the key “edition” (think of it like one big hash table)
- This value is no good as of Wed Feb 18...
- This value should only be readable by any domain ending in .zdnet.com
- This should be available to any resource within a subdirectory of /
- Send the cookie to any future requests to \(<domain>\)/\(<path>\)
Cookies: keeping “state” (cont.)

Amazon server creates ID 1678 for user

create entry

backend database

usual http request msg

usual http response msg

usual http request msg

usual http response msg

usual http request msg

usual http response msg

usual http request msg

usual http response msg

usual http request msg

usual http response msg

usual http request msg

usual http response msg

usual http request msg

usual http response msg

usual http request msg

usual http response msg

usual http request msg
What Are Cookies Used For?

• Authentication
  • The cookie proves to the website that the client previously authenticated correctly

• Personalization
  • Helps the website recognize the user from a previous visit

• Tracking
  • Follow the user from site to site;
  • Read about iPads on CNN and see ads on Amazon 😱
  • How can an advertiser (A) know what you did on another site (S)?
HTTP Request/Responses with Cookies

Response

HTTP/1.1 200 OK
Date: Tue, 18 Feb 2014 08:20:34 GMT
Server: Apache
Set-Cookie: session-zdnet-production=6bhqca1i0cbbiagu11sisac2p3; path=/; domain=zdnet.com
Set-Cookie: zregion=MTI5Lj1uMTI5LjE1Mzp1c3p1c3p1ZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN6
Set-Cookie: zregion=MTI5Lj1uMTI5LjE1Mzp1c3p1c3p1ZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN6
Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=zdnet.com
Set-Cookie: session-zdnet-production=59ob97fpinqe4bg6Ide4dvvq11; path=/; domain=zdnet.com

Subsequent visit

HTTP Headers

http://zdnet.com/
GET / HTTP/1.1
Host: zdnet.com
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20100113 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Cookie: session-zdnet-production=59ob97fpinqe4bg6Ide4dvvq11 zregion=MTI5Lj1uMTI5LjE1Mzp1c3p1c3p1ZDJmNWY5YTdkODU1N2Q2YzM5NGU3M2Y1ZTRmN6
Cookies and Privacy

Cookies permit sites to learn a lot about you

- supply name and e-mail to sites (and more!)
- third-party cookies (ad networks) follow you across multiple sites.
Why use cookies?

• Tracking users
  • Advertisers want to know your behavior
  • Ideally build a profile *across different websites*
    - Read about iPad on CNN, then see ads on Amazon?!
  • How can an advertiser (A) know what you did on another site (S)?
Why use cookies?

• Tracking users
  • Advertisers want to know your behavior
  • Ideally build a profile *across different websites*
    - Read about iPad on CNN, then see ads on Amazon?!
  • How can an advertiser (A) know what you did on another site (S)?

S shows you an ad from A; A scrapes the referrer URL
Why use cookies?

- **Tracking users**
  - Advertisers want to know your behavior
  - Ideally build a profile *across different websites*
    - Read about iPad on CNN, then see ads on Amazon?!
  - How can an advertiser (A) know what you did on another site (S)?

```
S shows you an ad from A; A scrapes the referrer URL
```

Option 1: A maintains a DB, indexed by your IP address

**Problem: IP addrs change**
Why use cookies?

- **Tracking users**
  - Advertisers want to know your behavior
  - Ideally build a profile *across different websites*
    - Read about iPad on CNN, then see ads on Amazon?!
  - How can an advertiser (A) know what you did on another site (S)?

  S shows you an ad from A; A scrapes the referrer URL

<table>
<thead>
<tr>
<th>Option 1: A maintains a DB, indexed by your IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem: IP addr change</td>
</tr>
<tr>
<td>- “Third-party cookie”</td>
</tr>
<tr>
<td>- Commonly used by large ad networks (doubleclick)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option 2: A maintains a DB indexed by a <em>cookie</em></th>
</tr>
</thead>
</table>
Cookie tracking
Cookie tracking
Cookie tracking

Snippet of reddit.com source
Cookie tracking

Snippet of reddit.com source

Our first time accessing adzerk.net
Cookie tracking

I visit reddit.com

HTTP Headers
GET /reddit/ads.html?sr=reddit.com,loggedout&bus2=http://www.reddit.com
Host: static.adzerk.net
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20100113 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive

Referer: http://www.reddit.com/

HTTP/1.1 200 OK
Date: Thu, 19 Feb 2015 17:37:51 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive

Set-Cookie: __cfduid=dc3a93cd30ca47b76600d63cde283e9b81424367471; expires=Fri, 19-Feb-16 17:37:51 GMT; path=/; domain=adzerk.net...

Later, I go to reddit.com/r/security

HTTP Headers
GET /reddit/ads.html?sr=security,loggedout&bus2=http://www.reddit.com
Host: static.adzerk.net
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.2.11) Gecko/20100113 Ubuntu/9.04 (jaunty) Firefox/3.6.11
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive

Referer: http://www.reddit.com/r/security

Cookie: __cfduid=dc3a93cd30ca47b76600d63cde283e9b81424367471
Cookie tracking

I visit reddit.com

We are only sharing this cookie with *.adzerk.net; but we are telling them about where we just came from

Later, I go to reddit.com/r/security
Cookies and Privacy

Cookies permit sites to learn a lot about you

You could turn them off ...but good luck doing anything on the internet!
Cookie Policy
Cookie Policy

- **Cookie policy**: A set of rules enforced by the browser
  - When the browser receives a cookie from a server, should the cookie be accepted?
  - When the browser makes a request to a server, should the cookie be attached?
- **Cookie policy is not the same as same-origin policy**
Login Session

GET /loginform HTTP/1.1
cookies: []
Login Session

GET /loginform HTTP/1.1
cookies: []

HTTP/1.1 200 OK
cookies: []

<html><form>...</form></html>
Login Session

GET /loginform HTTP/1.1
cookies: []

POST /login HTTP/1.1
cookies: []
username: chaganti
password: swarthmore

HTTP/1.1 200 OK
cookies: []

<html><form>…</form></html>
Login Session

GET /loginform HTTP/1.1
cookies: []

HTTP/1.1 200 OK
cookies: []

POST /login HTTP/1.1
cookies: []
username: chaganti
password: swarthmore

HTTP/1.1 200 OK
cookies: [session: e82a7b92]

GET /account HTTP/1.1
cookies: [session: e82a7b92]
Can the following attack succeed?

If we have a google analytics Javascript running on bank.com’s login page. Assume that the site has no frames, and everything on this page has the same origin. Can google analytics see Alice’s session cookie on bank.com?

A. Yes      B. No      C. Maybe      D. Something Else
Cookies

“In scope” cookies are sent based on origin regardless of requester

- **bank.com/login**
  - POST /login
  - bank.com

- **bank.com/**
  - GET /img/user.jpg
  - bank.com

- **attacker.com**
  - GET /img/user.jpg
  - bank.com
Aside: Domain Hierarchy

- **Domains**
  - Located after the double slashes, but before the next single slash
  - Written as several phrases separated by dots

- **Domains can be sorted into a hierarchy**
  - The hierarchy is separated by dots

![Domain Hierarchy Diagram]

- . (root)
- .edu
  - swarthmore.edu
  - mit.edu
- .org
  - cs88.org
- .com
  - piazza.com
  - google.com
. (root)

.edu

swarthmore.edu

cs.swarthmore.edu

.edu is a top-level domain (TLD), because it is directly below the root of the tree.

swarthmore.edu is a subdomain of edu

cs.swarthmore.edu is a subdomain of swarthmore.edu
Cookie Policy: Setting Cookies

- When the browser receives a cookie from a server, should the cookie be accepted?
- Server with domain X can set a cookie with domain attribute Y if
  - The domain attribute is a domain suffix of the server’s domain
    - X ends in Y
    - X is below or equal to Y on the hierarchy
    - X is more specific or equal to Y
  - The domain attribute Y is not a top-level domain (TLD)
  - No restrictions for the Path attribute (the browser will accept any path)
- Examples:
  - mail.google.com can set cookies for Domain=google.com
  - google.com can set cookies for Domain=google.com
  - google.com cannot set cookies for Domain=com, because com is a top-level domain
Cookie Policy: Sending Cookies

- When the browser makes a request to a server, should the cookie be attached?
- The browser sends the cookie if both of these are true:
  - The domain attribute is a domain suffix of the server’s domain
  - The path attribute is a prefix of the server’s path
Cookie Policy: Sending Cookies

Quick method to check cookie sending:
Concatenate the cookie domain and path. Line it up below the requested URL at the first single slash.

If the domains and paths all match, then the cookie is sent.
Quick method to check cookie sending:
Concatenate the cookie domain and path. Line it up below the requested URL at the first single slash.

If the domain or path doesn’t match, then the cookie is not sent.
Scoping Example

Cookie 1
name = cookie1
value = a
domain = login.site.com
path = /

Cookie 2
name = cookie2
value = b
domain = site.com
path = /

Cookie 3
name = cookie3
value = c
domain = site.com
path = /my/home

Concatenate the cookie domain and path. Line it up below the requested URL at the first single slash.
Can the following attack succeed?

If we have a google analytics Javascript running on bank.com’s login page. Assume that the site has iframes. Can google analytics see Alice’s session cookie on bank.com?
Cookies and web authentication

• An extremely common use of cookies is to track users who have already authenticated

• If the user already visited http://website.com/login.html?user=alice&pass=secret with the correct password, then the server associates a “session cookie” with the logged-in user’s info

• Subsequent requests (GET and POST) include the cookie in the request headers and/or as one of the fields: http://website.com/doStuff.html?sid=81asf98as8eak

• The idea is for the server to be able to say “I am talking to the same browser that authenticated Alice earlier.”
Aside: Trust in Web Advertising

• Advertising, by definition, is ceding control of Web content to another party
• Webmasters must trust advertisers not to show malicious content
• Sub-syndication allows advertisers to rent out their advertising space to other advertisers
  • Companies like Doubleclick have massive ad trading desks, also real-time auctions, exchanges, etc.
• Trust is not transitive!
  • Webmaster may trust his advertisers, but this does not mean he should trust those trusted by his advertisers
Aside: Example of an Advertising Exploit

- Video sharing site includes a banner from a large US advertising company as a single line of JavaScript...
- ... which generates JavaScript to be fetched from another large US company
- ... which generates more JavaScript pointing to a smaller US company that uses geo-targeting for its ads
- ... the ad is a single line of HTML containing an iframe to be fetched from a Russian advertising company
- ... when retrieving iframe, “Location:” header redirects browser to a certain IP address
- ... which serves encrypted JavaScript, attempting multiple exploits against the browser

[Provos et al.]
Aside: Third-Party Widgets

- Make sites “prettier” using third-party widgets
  - Calendars, visitor counters, etc.

- Example: free widget for keeping visitor statistics operates fine from 2002 until 2006

- In 2006, widget starts pushing exploits to all visitors of pages linked to the counter

  http://expl.info/cgi-bin/ie0606.cgi?homepage
  http://expl.info/demo.php
  http://expl.info/cgi-bin/ie0606.cgi?type=MS03-11&SP1
  http://expl.info/ms0311.jar
  http://expl.info/cgi-bin/ie0606.cgi?exploit=MS03-11
  http://dist.info/f94msIrfum67dh/winus.exe

[Provos et al.]
Login Session

GET /loginform HTTP/1.1
cookies: []

POST /login HTTP/1.1
cookies: []
username: chaganti
password: swarthmore

GET /account HTTP/1.1
cookies: [session: e82a7b92]

GET /img/user.jpg HTTP/1.1
cookies: [session: e82a7b92]
**Session Tokens: Security**

- If an attacker steals your session token, they can log in as you!
  - The attacker can make requests and attach your session token
  - The browser will think the attacker’s requests come from you
- Servers need to generate session tokens *randomly and securely*
- Browsers need to make sure malicious websites cannot steal session tokens
  - Enforce isolation with cookie policy and same-origin policy
- Browsers should not send session tokens to the wrong websites
  - Enforced by cookie policy
Session Token Cookie Attributes

What attributes should the server set for the session token?

- **Domain and Path**: Set so that the cookie is only sent on requests that require authentication
- **Secure**: Can set to True so the cookie is only sent over secure HTTPS connections
- **HttpOnly**: Can set to True so JavaScript can’t access session tokens
- **Expires**: Set so that the cookie expires when the session times out

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>token</td>
</tr>
<tr>
<td>Value</td>
<td>{random value}</td>
</tr>
<tr>
<td>Domain</td>
<td>mail.google.com</td>
</tr>
<tr>
<td>Path</td>
<td>/</td>
</tr>
<tr>
<td>Secure</td>
<td>True</td>
</tr>
<tr>
<td>HttpOnly</td>
<td>True</td>
</tr>
<tr>
<td>Expires</td>
<td>{15 minutes later}</td>
</tr>
</tbody>
</table>

*(other fields omitted)*