Web Security Model

CS155 Computer and Network Security
Web Security

Web Security Model

Vulnerabilities and Attacks  (Project 2 Material!)

Transport Layer Security — TLS, HTTPS

User Authentication and Session Management
Web Security Goals

Safely browse the web
Visit a variety of web sites without incurring harm

*Integrity*: Site A cannot affect session on Site B

*Confidentiality*: Site A cannot steal information from your device or Site B

Support secure web apps
Web-based applications should have same security properties as native applications
Attack Models

Malicious Website
Attack Models

**Malicious Website**

**Malicious External Resource**
Attack Models

Malicious Website

Network Attacker

Malicious External Resource
Attack Models

- Malicious Website
- Malicious External Resource
- Network Attacker
- Malware Attacker
HTTP Protocol

Protocol from 1989 that allows fetching of resources, such as HTML documents.

Clients and servers communicate by exchanging individual messages (as opposed to a stream of data).

URLs

http://cs155.stanford.edu:80/lectures?lec=08#slides
HTTP Request

GET /index.html HTTP/1.1
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
## HTTP Request

**GET**  
**Path**  
**Version**  
- `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`
## HTTP Flow

### Method Table

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GET</strong></td>
<td>Should only retrieve data not change state</td>
</tr>
<tr>
<td><strong>POST</strong></td>
<td>Used to submit an entity, often causing a change in state or side effects on the server.</td>
</tr>
<tr>
<td><strong>PUT</strong></td>
<td>Replaces all current representations of the target resource with the request payload.</td>
</tr>
<tr>
<td><strong>DELETE</strong></td>
<td>Deletes the specified resource</td>
</tr>
</tbody>
</table>

---

HTTP Request:

```plaintext
GET  /index.html  HTTP/1.1
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
```
HTTP/1.0 200 OK
Date: Sun, 21 Apr 1996 02:20:42 GMT
Server: Microsoft-Internet-Information-Server/5.0
Connection: keep-alive
Content-Type: text/html
Last-Modified: Thu, 18 Apr 1996 17:39:05 GMT
Set-Cookie: ...
Content-Length: 2543

<html>Some data... whatever ... </html>
HTTP/2

Major revision of HTTP released in 2015

Based on Google SPDY Protocol

No major changes in how applications are structured

Major changes (mostly performance):

- Allows pipelining requests for multiple objects
- Multiplexing multiple requests over one TCP connection
- Header Compression
- Server push
Cookies

An HTTP cookie is a small piece of data that a server sends to the web browser. The browser may store it and send it back with the next request to the same server.

**Session Management**
Logins, shopping carts, game scores, or anything else the server should remember.

**Personalization**
User preferences, themes, and other settings.

**Tracking**
Recording and analyzing user behavior.
HTTP/1.0 200 OK
Date: Sun, 21 Apr 1996 02:20:42 GMT
Server: Microsoft-Internet-Information-Server/5.0
Connection: keep-alive
Content-Type: text/html
Set-Cookie: trackingID=3272923427328234
Set-Cookie: userID=F3D947C2
Content-Length: 2543

<html>Some data... whatever ... </html>
Sending Cookie

HTTP Request

GET /index.html HTTP/1.1
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)
Cookie: trackingID=3272923427328234
Cookie: userID=F3D947C2
Referer: http://www.google.com?q=dingbats
Do Not Trust Cookies!

Client can send whatever content in a cookie!

```
name=balance, value=100
```

Generally you want to:

1) Store cryptographically protected secret

2) Unique (unforgeable) session identifier
Basic Rendering

Basic Browser Execution Model

Each browser window….

- Loads content
- Parses HTML and runs javascript
- Fetches sub resources (e.g., images, CSS, Javascript)

Post Fetch:

- Respond to events like onClick, onMouseover, onLoad, setTimeout
Frames

Windows may contain frames from different sources

Frame: rigid visible division

iFrame: floating inline frame

Why use frames?

Delegate screen area to content from another source

Browser provides isolation based on frames

Parent may work even if frame is broken
Document Object Model (DOM)

Javascript can read and modify page by interacting with DOM

Object Oriented interface for reading and writing website content

Browser takes HTML -> structured data (DOM is an OO representation)

**Examples:** document.alinkColor, document.URL, document.links

Also includes *Browser Object Model (BOM).* Access Window, Document, sometimes other state like history, browser navigation, cookies
<html>
  <ul id="t1">
    <li>Item 1</li>
  </ul>
</html>

<script>
  var list = document.getElementById('t1')
  var newitem = document.createElement('li')
  var newtext = document.createTextNode(text)
  list.appendChild(newitem)
  newitem.appendChild(newtext)
</script>
Islamic State claims it was behind Sri Lanka bombings

Officials raised the death toll in the Easter attacks to 321.

By SHAHANK BENGALI
Modern Website

The LA Times homepage includes 540 resources from nearly 270 IP addresses, 58 networks, and 8 countries.

CNN—the most popular news site—loads 361 resources.

Many of these aren’t controlled by the main sites.
Modern Website

Google Analytics Javascript (served from Google)
Modern Website

Google Analytics Javascript (served from Google)

Ad served from third party provider
Modern Website

Ad served from third party provider

Google Analytics Javascript (served from Google)

jQuery Javascript Library (served from MaxCDN)

Ad inside of frame
Modern Website

Google Analytics Javascript (served from Google)

Ad served from third party provider

Los Angeles Times

Ad inside of frame

jQuery Javascript Library (served from MaxCDN)

Local Javascript
Same Origin Policy

Theme: A web browser only should permit scripts contained in web page A to access data in web page B if both web pages have the same origin.

How: separate content with different trust levels (origins) into different frames, restrict communication between frames.
## What is an Origin?

### scheme://domain:port

<table>
<thead>
<tr>
<th>Compared URL</th>
<th>Outcome</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.example.com/dir/page2.html">http://www.example.com/dir/page2.html</a></td>
<td>Success</td>
<td>Same protocol, host and port</td>
</tr>
<tr>
<td><a href="http://www.example.com/dir2/other.html">http://www.example.com/dir2/other.html</a></td>
<td>Success</td>
<td>Same protocol, host and port</td>
</tr>
<tr>
<td><a href="http://username:password@www.example.com/dir2/other.html">http://username:password@www.example.com/dir2/other.html</a></td>
<td>Success</td>
<td>Same protocol, host and port</td>
</tr>
<tr>
<td><a href="http://www.example.com:81/dir/other.html">http://www.example.com:81/dir/other.html</a></td>
<td>Failure</td>
<td>Same protocol and host but different port</td>
</tr>
<tr>
<td><a href="https://www.example.com/dir/other.html">https://www.example.com/dir/other.html</a></td>
<td>Failure</td>
<td>Different protocol</td>
</tr>
<tr>
<td><a href="http://en.example.com/dir/other.html">http://en.example.com/dir/other.html</a></td>
<td>Failure</td>
<td>Different host</td>
</tr>
<tr>
<td><a href="http://example.com/dir/other.html">http://example.com/dir/other.html</a></td>
<td>Failure</td>
<td>Different host (exact match required)</td>
</tr>
<tr>
<td><a href="http://v2.www.example.com/dir/other.html">http://v2.www.example.com/dir/other.html</a></td>
<td>Failure</td>
<td>Different host (exact match required)</td>
</tr>
</tbody>
</table>
Frame Isolation

Each frame in a window has its own origin (proto://host:port)

Frame can only access data with the same origin

- Make HTTP requests, read/write DOM, access local storage
- Frame cannot access data associated with a different origin

Parent window cannot access data within a child frame (if it has a different origin)
Bounding Origins

Origins are defined for windows and frames
What’s Isolated? (Objects)

Each origin has local client side resources that are protected.

Examples:
- Cookies (local state)
- DOM storage
- DOM tree
- Javascript namespace
- Permission to use local hardware (e.g., camera or GPS)
SCRIPT EXECUTION

Scripts execute with the privileges of their parent frame/window’s origin.

**Pros:**

- You can load jQuery from a CDN and use it to manipulate your page.

**Cons:**

- The Google analytics script you included can also manipulate your page.
Modern Website

Google Analytics Javascript (served from Google)

Islamic State claims it was behind Sri Lanka bombings

Officials raised the death toll in the Easter attacks to 321.

By SHASHANK BENGALI
Modern Website

Google Analytics Javascript (served from Google)
Modern Website

Google Analytics Javascript (served from Google)

Ad served from third party provider

Los Angeles Times

Islamic State claims it was behind Sri Lanka bombings

Officials raised the death toll in the Easter attacks to 321.

By SHAHANK HENGAL
Modern Website

Google Analytics Javascript (served from Google)

Ad served from third party provider

Ad inside of frame

jQuery Javascript Library (served from MaxCDN)

Local Javascript
## Analogy to Operating Systems

<table>
<thead>
<tr>
<th>Subjects (Principals)</th>
<th>Operating System</th>
<th>Web Browser</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users (DAC)</td>
<td>System Calls, File System</td>
<td>Origins (MAC)</td>
</tr>
<tr>
<td>Process</td>
<td>DOM</td>
<td>Frame/Window</td>
</tr>
<tr>
<td>Subjects (Principals)</td>
<td>System Calls, File System</td>
<td>Origins (MAC)</td>
</tr>
<tr>
<td>Users (DAC)</td>
<td>System Calls, File System</td>
<td>Origins (MAC)</td>
</tr>
<tr>
<td>Process</td>
<td>DOM</td>
<td>Frame/Window</td>
</tr>
</tbody>
</table>
SOP: Frames

Frame A
Origin: a.domain.com

Frame B
Origin: b.domain.com
Domain Relaxation

You can change your `document.domain` to be a **super-domain**

a.domain.com -> domain.com **OK**

b.domain.com -> domain.com **OK**

a.domain.com -> com **NOT OK**
Domain Relaxation

You can change your document.domain to be a super-domain

a.domain.com -> domain.com **OK**

b.domain.com -> domain.com **OK**

a.domain.com -> com **NOT OK**

a.domain.co.uk -> co.uk
Domain Relaxation

You can change your document.domain to be a super-domain

a.domain.com -> domain.com  OK

b.domain.com -> domain.com  OK

a.domain.com -> com NOT OK

a.domain.co.uk -> co.uk NOT OK
A "public suffix" is one under which Internet users can (or historically could) directly register names. Some examples of public suffixes are .com, .co.uk and pvt.k12.ma.us. The Public Suffix List is a list of all known public suffixes.

The Public Suffix List is an initiative of Mozilla, but is maintained as a community resource. It is available for use in any software, but was originally created to meet the needs of browser manufacturers. It allows browsers to, for example:

- Avoid privacy-damaging "supercookies" being set for high-level domain name suffixes
- Highlight the most important part of a domain name in the user interface
- Accurately sort history entries by site

We maintain a fuller (although not exhaustive) list of what people are using it for. If you are using it for something else, you are encouraged to tell us, because it helps us to assess the potential impact of changes. For that, you can use the psl-discuss mailing list, where we consider issues related to the maintenance, format and semantics of the list. Note: please do not use this mailing list to request amendments to the PSL's data.

It is in the interest of Internet registries to see that their section of the list is up to date. If it is not, their customers may have trouble setting cookies, or data about their sites may display sub-optimally. So we encourage them to maintain their section of the list by submitting amendments.
Relaxation Attacks

What about: zakird.github.com -> github.com ?
Relaxation Attacks

Solution:

Both sides must explicitly set \texttt{document.domain} to share data.

Nowadays, user content on Github use \texttt{github.io} which is on the Mozilla Public Suffix List (PSL).
**postMessage**

**Sender:**
```javascript
targetWindow.postMessage(message, targetOrigin, [transfer]);
```

**targetWindow:** ref to window (e.g., from window.open, window.parent, window.frames)

**targetOrigin:** origin of targetWindow for event to be sent. Can be * or a URI

**message:** data to be sent

**Receiver:**
```javascript
window.addEventListener("message", receiveMessage, false);
function receiveMessage(event){
  if (event.origin !== "http://example.com")
    return
}
```
The **BroadcastChannel API** allows same-origin scripts to send messages to other browsing contexts. Simple pub/sub message bus between windows/tabs, iframes, web workers, and service workers.

```javascript
// Connect to the channel named "my_bus".
const channel = new BroadcastChannel('my_bus');

// Send a message on "my_bus".
channel.postMessage('This is a test message.');

// Listen for messages on "my_bus".
channel.onmessage = function(e) {
  console.log('Received', e.data);
};

// Close the channel when you're done.
channel.close();
```
SOP: HTTP Responses

Images, CSS, Fonts: can load from another origin, but cannot inspect their content. Similar to loading a frame from another origin.

Javascript: Similar to passive objects. Cannot view source, but you can call functions.

\[ f\.toString() \rightarrow \text{gives you source code} \]
XMLHttpRequests (XHR) allow developers to retrieve data from a URL in Javascript (e.g., AJAX Call)

You cannot issue requests cross origin

You can only read responses from the same origin

But it allows you to insert arbitrary header value when issuing request. (e.g. SOAPAction header)
CORS Example

Sometimes you want to allow another domain to access your resources

Servers can add **Access-Control-Allow-Origin** ACAO header that allows more permissive access
No CORS

Origin: example.com

$.ajax({url: "secure.com", success: function(result){
  $('#div1').html(result);
});

GET

Server: secure.com
CORS Success

Origin: example.com

$.ajax({url: "secure.com", success: function(result){
  $('#div1').html(result);
}});
CORS Success

Origin: example.com
$.ajax({url: "secure.com", success: function(result){
    $('#div1').html(result);
}});
CORS Wildcard

**Origin:** example.com

$.ajax({url: "secure.com", success: function(result){
    $('#div1').html(result);
}});

**Server:** secure.com

Header: Access-Control-Allow-Origin: *

GET OPTIONS

GET

DATA
CORS Failure

Origin: example.com

$.ajax({url: "secure.com", success: function(result){
    $('#div1').html(result);
}});

Server: secure.com

Header:
Access-Control-Allow-Origin: bank2.com
SOP: Cookies

Cookies allow server to store small piece of data on the client

Client sends cookie back to server next time the client loads a page

Sending cookies only to the right websites really Important

  - Don’t send cookie for bank.com to attacker.com if authentication token
SOP: Cookies

Cookies use a separate definition of origins.

**DOM SoP:** Origin A can access Origin B if matches:

\[(\text{scheme, domain, port})\]

**Cookie SoP:** Cookies are scoped based on

\[(\text{[scheme], domain, path})\]

cs155.stanford.edu/foo/bar
A page can set a cookie for its own domain or any parent domain, as long as the parent domain is not a public suffix.

The browser will make a cookie available to the given domain including any sub-domains

<table>
<thead>
<tr>
<th>Subdomain</th>
<th>Allowed</th>
<th>Disallowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>login.site.com</td>
<td>other.site.com</td>
<td></td>
</tr>
<tr>
<td>site.com</td>
<td>com</td>
<td></td>
</tr>
<tr>
<td>othersite.com</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SOP: Cookie Scope Setting

A page can set a cookie for its own domain or any parent domain, as long as the parent domain is not a public suffix.

The browser will make a cookie available to the given domain including any sub-domains.

- **Allowed**:
  - login.site.com
  - other.site.com
  - zakird.github.io
can set cookies for github.io (unless github.com is on Public Suffix List)

- **Disallowed**:
  - site.com
  - othersite.com
  - othersite.com

You don’t know who set a cookie when you receive it.
What Cookies are Sent?

Browser *always* sends all cookies in a URL scope’s:

- Cookie’s domain is domain suffix of URL’s domain
- Cookie’s path is a prefix of the URL path
Cookie Scoping Example

Cookie 1:
name = mycookie
value = mycookievalue
domain = login.site.com
path = /

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

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

<table>
<thead>
<tr>
<th>Domain</th>
<th>Cookie 1</th>
<th>Cookie 2</th>
<th>Cookie 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkout.site.com</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>login.site.com</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>login.site.com/my/home</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>site.com/my</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Problem with HTTP Cookies

Network Attacker
Can Observe/Alter/Drop Traffic

HTTPS Connection

domain: bank.com
name: authID
value: auth
Problem with HTTP Cookies

HTTP Connection

Network Attacker
Can Observe/Alter/Drop Traffic

Attacker tricks user into visiting http://bank.com
Problem with HTTP Cookies

Network Attacker
Can Observe/Alter/Drop Traffic

HTTPS Connection

Attacker tricks user into visiting http://bank.com
Secure Cookies

A secure cookie is only sent to the server with an encrypted request over the HTTPS protocol.
Interaction with DOM

Cookie SOP:

x.com/a does not see cookies for x.com/b

Dom SOP:

x.com/a can access the DOM of x.com/b

Path separation is done for efficiency not security:

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

What happens when your bank includes Google Analytics Javascript? Can it access your Bank’s authentication cookie?
Bank Loads Google Analytics

Javascript is running with Origin’s privileges. Can access document.cookie.

Nothing prevents:

HttpOnly Cookies

You can set setting to prevent cookies from being access via the DOM

Set-Cookie: id=a3fWa; Expires=Wed, 21 Oct 2015 07:28:00 GMT; Secure; HttpOnly
Which Cookie is Sent?

attacker.com

<html>
  <img src="https://bank.com"/>
</html>
Which Cookie is Sent?

attacker.com

<html>
  <img src="https://bank.com"/>
</html>

All the cookies for bank.com are sent with this request
Which Cookie is Sent?

attacker.com

<html>
  <img src="https://bank.com/transfer?from=victim,to=attacker"/>
</html>
Which Cookie is Sent?

attacker.com

<html>
  <img src="https://bank.com/transfer?from=victim,to=attacker"/>
</html>

Known as Cross-site request forgery or CSRF Attack
Web Security Model

CS155 Computer and Network Security