Reading Quiz
Announcements

• Midterm Exam on October 3\textsuperscript{rd}
  • 1-2.30pm and at 4 - 5.30pm @ SCI 199
  • Review in Class on Thursday
  • Dunkin Donuts + Coffee!
  • Accommodations: not informed me? do so TODAY!!!

• Thanks for your feedback last week!
  • 50%: Flipped classroom: Yay! 50%: Flipped classroom: No!
  • First upper-level G2 for majority of the class
  • Flipped on Tuesday
  • Content on Thursday + 10 minutes of ask anything!
    • Everything will be recorded
  • Solutions to worksheets: up today on Edstem
  • Feedback is important!
SQL Injection

```php
$result = mysql_query("select * from Users
where(name='".$user.' and password='".$pass.');\n");
$result = mysql_query("select * from Users
where(name='"spongebob.' or 1=1);#
DROP TABLE Users; --
' and password='whocares');\n");
```

Can chain together statements, and can modify existing statements
The underlying issue

```
$statement = $db->prepare("select * from Users
  where(name=? and password=?);");
```

Prepare is only applied to the leaves, so the structure of the tree is fixed.

Parametrized SQL statement.
- compile the query first
- plug inputs later
The underlying issue

```php
$statement = $db->prepare("select * from Users
  where(name=? and password=?);");
```

Prepare is only applied to the leaves, so the structure of the tree is fixed.
Not Just SQL!

Injection vulnerabilities are a generic issue!
PREVENTING INJECTION ATTACKS

validate all the inputs!
Most injection attacks trick the 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?

Client

- Browser: renders the webpage
- Private data
- Malware attacker

Server

- Web Server hosts the web page
- Database
- Web server attacker
- Network attacker

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
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?

• Attacks Against Clients
  • Cross Site Scripting (XSS) and Response Splitting
  • Cross Site Request Forgery (CSRF)
  • Clickjacking

• Attacks Against Servers
  • SQL Injection
  • Unrestricted Uploads
  • CGI shell injection
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
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
Interacting with web servers: dynamic content

http://facebook.com/delete.php

Path to the resource


arguments

server generates the content on the fly
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
HTTP Overview

2. Browser establishes connection with server.
   Looks up “some.host.name.tld”
   connects //more on this later
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
   [Blank line]
   (Data data data data data...)

Slide 25
HTTP Overview

5. Browser renders the response, fetches any additional objects, and closes the connection.
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 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 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 Body
Example

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

<html><head><title>Demo Server</title></head><body>.....</body></html>
The Swarthmore IT department decides to come up with a new header field 
Trusted_Source in HTTP response fields. This field is used to tag all 
websites that have swarthmore as the origin, as trusted. How can 
swarthmore ensure deployability? Can this scheme be subverted by a 
malicious user?

A. Cannot be deployed, hence cannot be subverted.  
B. Can be deployed, cannot be subverted  
C. Can be deployed, can be subverted 
D. Hard to deploy, can be subverted.

Discussion Question: HTTP Extensible Headers

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  
Referer: http://google.com?q=swarthmore.edu

HTTP/1.0 200 OK  
Content-Type: text/html  
Content-Length: 1299  
Date: Sun, 01 Sep 2013 21:26:38 GMT  
Trusted_Source = Yes  
[Blank line]  
(Data data data data...)

A. Cannot be deployed, hence cannot be subverted.  
B. Can be deployed, cannot be subverted  
C. Can be deployed, can be subverted 
D. Hard to deploy, can be subverted.
State(less)

(XKCD #869, “Server Attention Span”)
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)?
Cookies are key-value pairs

Set-Cookie: edition=us; expires=Wed, 18-Feb-2015 08:20:34 GMT; path=/; domain=.zdnet.com
Set-Cookie: session-zdnet-production=590b97fpinqe4bg6de4dvvqy1; path=/; domain=.zdnet.com
Set-Cookie: user_agent=desktop
Set-Cookie: zdnet_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

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)
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...
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

Client

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>
Discussion Question: Let’s say the web-page at http://cute-puppies.com looks like the following:

```html
<html>
  <body>
    <p>Here is a GIF of puppies</p>
    <img src="https://cute-puppies.com/labrador.gif">
    <script type="text/javascript">
      src="http://yahoo.com/analytics.js"></script>
  </body>
</html>
```

Alice uses Mozilla Firefox. In her first browser tab, she has https://swarthmore.edu open. In a second tab, she opens http://cute-puppies.com. In a third tab, she opens http://cute-puppies.com once again. Which entities know that the same person visited cute-puppies.com twice?

A. cute-puppies.com operators  B. yahoo.com operators  C. Mozilla  D. Swarthmore
Browser: Basic Execution Model

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

Events
- User actions: OnClick, OnMouseover
- Rendering: OnLoad, OnUnload
- Timing: setTimeout(), clearTimeout()
<html>

...  

<p> The script on this page adds two numbers </p>

<script>
    var num1, num2, sum
    num1 = prompt("Enter first number")
    num2 = prompt("Enter second number")
    sum = parseInt(num1) + parseInt(num2)
    alert("Sum = " + sum)
</script>

...  

</html>
(i)Frames

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)
Event-Driven Script Execution

```javascript
<script type="text/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**
Document Object Model (DOM)

Javascript can read and modify page by interacting with DOM

- Object Oriented interface for reading/writing page content
- Browser takes HTML -> structured data (DOM)

```html
<p id="demo"></p>

<script>
  document.getElementById('demo').innerHTML = Date()
</script>
```
Modern Website

Which sections of this page generate cookies? How many third-party cookies?
Modern Website

- Third-party ad
- Google analytics
- Framed ad
- jQuery library
- Local scripts
Modern Website

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

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

Many of these aren’t controlled by the main sites.
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.”
Web Isolation

Safely browse the web

Visit a web sites (including malicious ones!) without incurring harm

**Site A** cannot steal data from your device, install malware, access camera, etc.

**Site A cannot affect session on Site B or eavesdrop on Site B**

Support secure high-performance web apps

Web-based applications (e.g., Google Meet) should have the same or better security properties as native desktop applications
Web Security Model

Subjects
“Origins” — a unique 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 read bank.ch
- Integrity: evil.com should not be able to modify the content of bank.ch
**Origins Examples**

**Origin defined as scheme://domain:port**

**All of these are different origins — cannot access one another**
- http://swarthmore.edu
- http://www.swarthmore.edu
- http://swarthmore.edu:8080
- https://swarthmore.edu

**These origins are the same — can access one another**
- http://swarthmore.edu
- http://swarthmore.edu:80
- http://swarthmore.edu.cs
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
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
HTTP Cookies

Set-Cookie: <cookie-name>=<cookie-value>
Cookies

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

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

- `POST /login` bank.com
- `GET /img/user.jpg` bank.com
- `GET /img/user.jpg` attacker.com

`<html><form>...</form></html>`
Cookie Same Origin Policy

Cookies use a different definition of origin than the DOM:

(domains, paths): (cs88.swarthmore.edu, /foo/bar)

A page can set a cookie for its domain or any parent domain
(as long as the parent domain is not a public suffix).
Can set a cookie for its path or any parent path.

Browser sends cookies that are in a URL’s scope. Cookies that...

belong to domain or parent domain

AND

are located at the same path or parent path
Setting Cookie Scope

Websites can set a scope to be any parent of domain and URL path

✔ cs88.swarthmore.edu *can* set cookie for cs88.swarthmore.edu
✔ cs88.swarthmore.edu *can* set cookie for swarthmore.edu
❌ swarthmore.edu *cannot* set cookie for cs88.swarthmore.edu

✔ website.com/* can* set cookie for website.com/
✔ website.com/login *can* set cookie for website.com/
❌ website.com *cannot* set cookie for website.com/login
Scoping Example

cookie domain is suffix of URL domain \( \land \) cookie path is a prefix of URL path
Scoping Example

| cookie domain is suffix of URL domain ∧ cookie path is a prefix of URL path |
|-----------------------------------------------|-----------------|------------------|
| checkout.site.com                            | No              | Yes              | No               |
| login.site.com                               | Yes             | Yes              | No               |
| login.site.com/my/home                        | Yes             | Yes              | Yes              |
| site.com/account                              | No              | Yes              | No               |

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