Web Application Attack Techniques

Mark Shtern
Popular attack targets

• Web
  – Web platform
  – Web application (12 issues per Application)
Web platform vulnerabilities

- Sample files in production environment
- Configured incorrectly
- Source code disclosure
- Canonicalization
- Server extensions
- Input validation (e.g. buffer overflow)
Web Application Vulnerabilities

- Identify applications running on ports
- Find version information (if possible)
- Look for exploits on the Internet
- Run the exploits against the target application.
Web Application

- Injections (*not only in web applications*) (one in five applications)
- Cross-Site Scripting (affected two thirds of applications 2011)
- Cross Site Request Forgery
- Remote code execution
- Format String
- Username enumeration
Web Application

• Broken Authentication and Session Management
• Insecure Direct Object References
• Security Misconfiguration
• Sensitive Data Exposure
• Missing Function Level Access Control
• Using Known Vulnerable Components
• Unvalidated Redirects and Forwards
Vector of Attack
SQL Injection

• The ability to inject SQL commands into the database engine through an existing application
What is SQL?

• SQL stands for **Structured Query Language**
• Allows us to access a database
• ANSI and ISO standard computer language
  – The most current standard is SQL99
• SQL can:
  – execute queries against a database
  – retrieve data from a database
  – insert new records in a database
  – delete records from a database
  – update records in a database
SQL Queries

• With SQL, we can query a database and have a result set returned

• A query looks like this:

  SELECT LastName
  FROM users
  WHERE UserID = 5;
SQL Data Manipulation Language (DML)

- SQL includes a syntax to update, insert, and delete records:
  - SELECT - extracts data
  - UPDATE - updates data
  - INSERT INTO - inserts new data
  - DELETE - deletes data
SQL Data Definition Language (DDL)

• The Data Definition Language (DDL) part of SQL:
  – Creates or deletes database tables
  – Defines indices (keys)
  – Specifies links between tables
  – Imposes constraints between database tables

• Some of the most commonly used DDL statements in SQL are:
  – CREATE TABLE - creates a new database table
  – ALTER TABLE - alters (changes) a database table
  – DROP TABLE - deletes a database table
Example

Common vulnerable login query

```
SELECT * FROM users
WHERE login = 'root'
AND password = '123'
```

(If it returns something then login!)
Example

formusr = root' or 1=1
formpwd = anything

Final query would look like this:

```
SELECT * FROM users
WHERE username = 'root' or 1=1
AND password = 'anything'
```
Countermeasures

• Do not use string concatenation or string replacement
• Use prepared or parameterized SQL statements, also known as prepared statements
• Encrypt the underlying data such that it cannot be disclosed in the case of a SQL injection-induced breach
• Validate the data being used in the SQL statement
Cross-Site Scripting (XSS)

- Scripting: Web Browsers can execute commands
  - Embedded in HTML page
  - Supports different languages (JavaScript, VBScript, ActiveX, etc.)
  - Most prominent: JavaScript
- “Cross-Site” means: Foreign script sent via server to client
  - Attacker makes Web-Server deliver malicious script code
  - Malicious script is executed in Client’s Web Browser
- Attack:
  - Steal Access Credentials, Denial-of-Service, Modify Web pages
  - Execute any command at the client machine
Simple XSS attack

• JSP page
  `<% out.println("Welcome " + request.getParameter("name"))%>
  http://example.com?name=test`

• Attack
  http://example.com?name=<script>alert("Attack")<script>
XSS Example

• Attacker
  – Posts forum message
    • Subject: “Get free money”
    • Body <script>attack code</script>

• WEB Server
  – Stores the post

• User
  – Reads the message
  – Malicious code executed
Cross-Site Scripting

• The three conditions for Cross-Site Scripting:
  – A Web application accepts user input
  – The input is used to create dynamic content
  – The input is insufficiently validated
Cross Site Request Forgery

• Exploits a website’s trust in the user/browser
• Generally involves websites that rely on the identity of the users
• Performs HTTP requests of the attacker’s choosing
• Intent is to trick a user into performing an HTTP request/action
• Attack is not “personal”
Cross Site Request Forgery

• Websites use URLs to specify requests for an action
• Example (from wikipedia)
  – <img src="http://bank.example/withdraw?account=bob&amount=1000000&for=mallory">
• Instead of the withdrawal happening from inside the banking website, an image in Mallory’s website attempts to trigger a transfer from Bob’s bank account to Mallory’s which will work if Bob’s bank cookie has not expired
Typical CSRF Process

• Attacker posts an IMG tag or other code that sends an HTTP request
• Code posted usually causes a request to be made to another site (hence the term “cross-site”)
• Victim loads page with bad code
• Victim unknowingly causes an HTTP request to be sent
Countermeasures

• Web application should insert random values, tied to the specified user’s session, into the forms it generates

• Web application should re-authenticate every time when users are about to perform a particularly dangerous operation
Automatic Tools

- **Burp/WebScarab**
- Proxy Server
- Spider tool
- Vulnerability scanner
- Repeater tool
- Sequencing tool
- Decode/Encode tool
Practice

- [https://google-gruyere.appspot.com/](https://google-gruyere.appspot.com/)
- [Burp/WebScarab](https://burp-web-scarab)