
MONASH University
MONASH
INFORMATION
TECHNOLOGY
FIT5192 Lecture 11:
Securing Enterprise Web Applications
GROUP
OF EIGHT
AUSTRALIA

MONASH University

Last Lecture

- Understand the role that web services can have for enterprise web applications and the web as a whole.
- Review the REST approach to web services
- Review JSON processing approaches with the Java EE 7 platform.

GROUP OF EIGHT AUSTRALIA

MONASH University

This Lecture

- Identify common security issues that can impact enterprise web applications and approaches to prevent common security implementations.
- Understand how we can use JAAS to implement user authentication within Java EE applications.

GROUP OF EIGHT AUSTRALIA

Security

Importance of Security

- While security is always a priority for almost any application development, it has an even greater role for the Enterprise
- **Enterprises often deal with more confidential data as opposed to non-business applications:**
  - Employee data
  - Customer data
  - Stored payment credentials
  - And so on…
- As a result, developers need to be aware of common issues surrounding web security and how to appropriately protect against attack vectors

Web Application Security

- **Applications which are available externally to many clients via the Internet provide us with a few challenges:**
  - Without appropriate firewall rules, our application is easily accessible to outside connections.
  - Clients can *potentially* send different types of data to our application.
    - We need to validate that the data being received is of the expected message format.
  - A complex web application architecture comprised of many different web servers and endpoints create additional potential attack vectors.

Open Web Application Security Project (OWASP)

- **OWASP is an open-source project focusing on web application security and includes a wide variety of different organisations.**
- Many projects supported by OWASP help validate many different aspects of web applications and their underlying technologies.
- **The OWASP project Top 10 provides a frequently updated list that helps identify the most serious security risks for a range of different organisations.**
  – Let's take a look at the identified key issues impacting various different web applications today…

Top 10 Web Security Issues (OWASP)

1. Injection
2. Broken Authentication and Session Management
3. Cross-Site Scripting (XSS)
4. Insecure Direct Object References
5. Security Misconfiguration
6. Sensitive Data Exposure
7. Missing Function Level Access Control
8. Cross-Site Request Forgery (CSRF)
9. Using Known Vulnerable Components
10. Unvalidated Redirects and Forwards

From : https://www.owasp.org/index.php/Top_10_2013-Top_10

SQL Injection

- **Code injection technique that is the most common security issue plaguing many web applications in the past (and present!).**
  - **An attacker is able to execute their own malicious SQL statement against the web application should a vulnerability be exploited.**
- Easy to exploit when applications are reliant on user input data which is used in conjunction with an SQL database.
  - **Username/Password for Authentication**

SQL Injection: Example (1)

- Scenario: User 'forgets' their password and wants to retrieve it.
  – Application will send an email to the user with a password reset link if they provide a valid email associated with the user account.

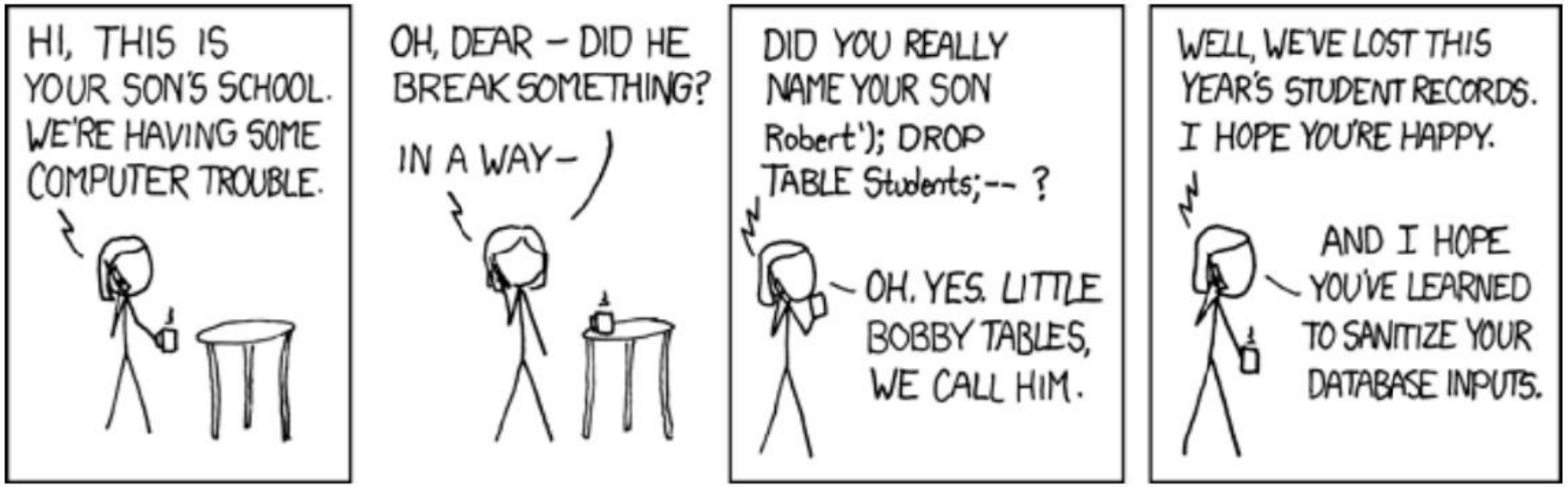
- **URL for getting email address:**
  http://localhost/app/forgotPassword?email=Matthew.Kairys@monash.edu

SQL Injection: Example (2)

- **Attempting to execute SQL Injection:**
  ```
  http://localhost/app/forgotPassword?
  email=email AND email=(SELECT COUNT(*) FROM
  USERS);--
  ```
- If the input is not filtered to prevent SQL execution, the above statement returns the number of records for any given record name.
- **If the attacker wanted to be really evil…**

  ```
  http://localhost/app/forgotPassword?
  email=email AND email=(DROP TABLE USERS);--
  ```

SQL Injection



Blocking Injection Attacks

Preventing SQL Injections with JPA

- DO NOT create queries like this:

```
String query = "SELECT * FROM USERS WHERE
email= " + emailFromUrl;
```

- **The above approach does not escape the URL GET parameter and enables malicious SQL statements to be executed.**
- Creating queries (dynamic or named) via JPA and parameterised values by default protects against this type of attack.
- **Source:** http://xkcd.com/327/

Working with Passwords

- Handling passwords in a secure manner is a critical consideration when developing a web application.
- **Storing passwords as plain Strings within a database is a massive security oversight:**
  - Should web administrators be able to view passwords?
  - What happens if the database is compromised?

Password Hashing

- To store passwords in a more secure manner, we can use one-way hashing algorithms to obtain a completely different encrypted value.

  Example using MD5:

  - 'password‘ => '5f4dcc3b5aa765d61d8327deb882cf99'

- **By storing passwords in this manner, we can hash user password inputs and compare them with the stored value in the database.**

  – We cannot obtain the value of the password otherwise unless we attempt to brute-force the password by generating hashes for each potential password combination

Password Hashing: DB Example

- User Table Example:

USER		
ID	USERNAME	PASSWORD
1	**Admin**	**5f4dcc3b5aa765d61d8327deb882cf99**
2	Chris	2ab96390c7dbe3439de74d0c9b0b1767

Rainbow Table Attacks

- Unfortunately storing passwords using the previous hash approach is still not very secure!
- Majority of hash algorithms are designed for high performance.
  - With the computational hardware available today common passwords are easy to be computed via brute-force method.
- Rainbow tables are precomputed key-value pair tables which enable the reverse lookup of hash algorithms against common phrases.
  - These are easily available online and for purchase.

Example: Rainbow Table

- Let's check what the password was for 'admin':

MD5 HASH	PASSWORD
C75fd00c9126c6940629d96357587aa3	passworda
4de98e9fc935c2d872482c23cccc70a2	passwordb
1b01dab91d4a847cd9c0838537b3bb44	passwordc
5f4dcc3b5aa765d61d8327deb882cf99	passwordd
a826176c6495c5116189db91770e20ce	passworde

- **We can prevent these type of attacks through the use of *password salting.***

Password Salting

- We can strengthen stored passwords by adding random data that is associated with the hashed password.
  - Salted value can be simply added with the password (password + passwordSalt).
  - Random data must be generated for each user for optimum security.
- **Example using MD5:**

```
Password = 'password';
passwordSalt = 'RN5_43<m84@|]5&66[33';
password = password + passwordSalt;
Hashed password
'password' => '4a935057ff675ffc83a4e46ca4f873dd'
```

Password Salting: DB Example

- **User Table Example:**

USER			
ID	USERNAME	PASSWORD	PASSWORD_SALT
1	admin	4a935057ff675ffc83a4e46ca4f873dd	RN5_43<m84@|]5&66[33
2	mkairys	939358daeb4229850179d1075d6086ed	[2![$0644D<Z7s1

Password Salting: Benefits

- By each user having their own salt attached with the password they input, we prevent the effectiveness of rainbow table attacks.
  - Long random data segments will significantly increase computational requirements to brute-force hashed passwords.
- If your database is compromised, the attacker would need to brute-force each individual user password with their assigned salt to obtain the true value.
  - Other algorithms can be used which are designed to take time to compute (E.g. bcrypt, scrypt) to also prevent the effectiveness of this approach.

MONASH
University

JAAS

Java Authentication and Authorization Service (JAAS)

- JAAS provides a set of APIs that enable various services within the Java EE platform to support authentication and enforce access control lists (ACL) upon user roles.
- In particular, we will be using this service to integrate user authentication into our web applications.

JAAS: Supported Authentication Methods

- The Java EE platform supports the following authentication mechanisms:
  - Basic authentication
  - Form-based authentication
  - Digest authentication
  - Client authentication
  - Mutual authentication
- **We will focus on Basic and Form-based authentication methods for this unit.**

Form-based Authentication via JAAS (1)

- We can setup rules in our web application configuration defining how we are going to validate users via JAAS and which sections of the application are restricted.
- The server will supply us with a cookie with an assigned session identifier (JSESSIONID) upon attempting to access a restricted area of the application.
- Must be submitted via post to a URL containing the String j_security_check.

  E.g. http://localhost/app/j_security_check

Form-based Authentication via JAAS (2)

- **We submit the username (j_username) and password (j_password) with the cookie to the server.**
  - If successful, we will be authenticated with the server and redirected back to the original protected resource.
  - If successful but we lack the required permissions to access the resource, we will receive a permissions error and redirected to an error page.
  - If unsuccessful due to incorrect user credentials, we will be redirected to an error page.
    - This error page will most likely have the login form again to enable the user to retry authentication.

Demo

- **Let's do a walkthrough of the process required to setup a web/enterprise application with user authentication functionality.**
  - Breakdown of the Lab exercise.
  - Highlighting the process in NetBeans to setup user authentication.

Summary

- Identify common security issues that can impact enterprise web applications and approaches to prevent common security implementations.
  - SQL Injections are easy to execute, easy to prevent!
  - Passwords should always be salted for added security.
- Understand how we can use JAAS to implement user authentication within Java EE applications.
  - **Make sure you do your readings as they explain more complex user authentication approaches in Java EE:**
  - In particular, additional configuration options and annotation methods for specifying access permissions based on roles

Next Lecture

- **Start the Module 2 on ASP.NET!**

- **Organise Assignment Demo/Interviews**
- **Interviews from Wednesday next week during your normal tutorial time**
  - All interviews will hopefully be completed by *Friday this week.*

MONASH University

See you in the Studio !

Readings

MONASH University

- **Part X *Security* in The Java EE 7 Tutorial**

Chapter 47 to 50 cover the various JEE security technologies