TOWERS WATSON

Security Training

Jacob Meline
June 24, 2015

Contents

1 Introduction

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Hello world

2 SQL Injection

Further readings:

- acunetix's article

2.1 Why is SQL Injection a problem?

Example of attack

- A simple example of SQLi
- A more realistic example

Lets say that a form is asking for user credentials and enter "Tom," well, that is fine and all. If you enter Tom" you're going to have some vulunerability concerns. You can do Tom"; DROP ALL DATABASES; to do some serious damage. To mitigate this attack, you can simply put a slash before any quotation mark. This forces SQL to read it as a quotation mark and not a continuation of a command.

2.2 How to mitigate SQLi attacks?

Links to useful sites

- SQLi Prevention Cheat Sheet

3 Cross Site Request Forgery

4 Cross Site Scripting

Cross-Site Scripting is a vulnerability that doesn't sanitize user input properly. It allows an attacker to inject HTML or client side script such as Javascript into a website. It is commonly used to steal cookies. Cookies are used for authenticating, tracking, and maintaining specific information about users. There are three different types of Cross Site Scripting:

Persistent
: the malicious input originates from the website's database.

Non-Persistent
: the malicious input originates from the victim's request.

DOM-Based
: the vulnerability is in the client-side code rather than the server-side code.

4.1 How to mitigate Cross-Site Scripting

- The most important way to prevent XSS attacks is to perform secure input handling.
  - Most of the time, encoding should be performed whenever user input happens
  - In some cases, encoding has to be replaced by or complemented with validation
  - Secure input handling has to take into account which context of a page the user input is inserted into
  - To prevent all types of XSS attacks, secure input handling has to be performed in both client-side and server-side code
- Content Security Policy provides an additional layer of defense for when secure input handling fails

5 Clickjacking

5.1 What is Clickjacking?

OWASP gives a good example of what click jacking is:

imagine an attacker who builds a web site that has a button on it that says "click here for a free iPod". However, on top of that web page, the attacker has loaded an iframe with your mail account, and lined up exactly the "delete all messages" button directly on top of the "free iPod" button. The victim tries to click on the "free iPod" button but instead actually clicked on the invisible "delete all messages" button. In essence, the attacker has "hijacked" the user's click, hence the name "Clickjacking".

Basic ingredients to prepare for a clickjacking attack are:

Iframe This is a frame in the HTML that frames a webpage in it

Z-Index Decides the iframe index in the stack

Opacity Makes the iframe transparent

Position:Absolute Lines up the iframe with the dummy page

5.2 How to mitigate against it?

- Sending the proper X-Frame-Options HTTP response headers that instruct the browser to not allow framing from other domains
- Employing defensive code in the UI to ensure that the current frame is the most top level window
  - Frame-breaker By inserting this script into the header of your site, it is an easy way to break the iframe.

```
<script>
if (top != self){ top.location = self.location; }
</script>
```

6 Authentication

All approaches for human authentication rely on at least one of the following

Multi Authenication

Something you know (eg. a password) This is the most common kind of authentication used for humans. We use passwords every day to access our systems. Unfortunately, something that you know can become something you just forgot. And if you write it down, then other people might find it.

Something you have (eg. a smart card) This form of human authentication removes the problem of forgetting something you know, but some object now must be with you any time you want to be authenticated. And such an object might be stolen and then becomes something the attacker has.

Something you are (eg. a fingerprint) Base authentication on something intrinsic to the principal being authenticated. It's much harder to lose a fingerprint than a wallet. Unfortunately, biometric sensors are fairly expensive and (at present) not very accurate.

6.1 Account Lockout

In an account lockout attack, an attacker attempts to lock out user accounts by purposely failing the authentication process as many times as needed to trigger the account lockout functionality. This in turn prevents even the valid user from obtaining access to their account. For example, if an account lockout policy states that users are locked out of their accounts after three failed login attempts, an attacker can lock out accounts by deliberately sending an invalid password three times. On a large scale, this attack can be used as one method in launching a denial of service attack on many accounts. The impact of such an attack is compounded when there is a significant amount of work required to unlock the accounts to allow users to attempt to authenticate again.

6.1.1 Ebay Account Lockout Attack

At one time, eBay displayed the user-id of the highest bidder for a given auction. In the final minutes of the auction, an attacker who was wanting to outbid the current highest bidder could attempt to authenticate three times using the targeted account. After three deliberately incorrect authentication attempts, eBay password throttling would lock out the highest bidder's account for a certain amount of time. An attacker could then make their own bid and the legitimate user would not have a chance to place a counter-bid because they would be locked out of their account.

6.1.2 Mitigation Strategies

Brute force attacks are surprisingly difficult to completely mitigate. Careful design and implementation will assist in helping to minimize these attacks.

- Since passwords are not always the best option. Having a public-private key setup (Something you have) can help in some circumstances

- Time-increasing failed login. For example, after 3 unsuccessful logins, you could block the login for this account for 30 seconds. After the 30 seconds, if the 4th login is still unsuccessful, you stop it for 1 minute, then 5m, and you increase more and more

6.2 Creating Strong Passwords

Your password is too short

6.3 Password History

It is typical in a company to require its employees to have a new password every 6 - 12 months. On top of that, there can be time restrictions on the previous passwords so that even if old passwords were compromised, attackers wouldn't be able to brute force their way through a secure network.

6.4 What is the correct way of storing passwords?

Never store passwords as plaintext Store the hashes, never the actual passwords

Add a long, unique random salt to each password you store The point of a **Salt** is to make each password unique and long enough that brute force attacks are a waste of time. This allows the password to be immune to rainbow table attack.

Don't invent your own "clever" password storage scheme Use something that has been thoroughly tested.

Use a cryptographically secure hash If you use MD5, i'll hurt you. SHA-1 is another one to avoid. These don't add any salt to the passwords