TOWERS WATSON

# Security Training

Jacob Meline

June 19, 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

## 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
  - 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>
```

- Employing defensive code in the UI to ensure that the current frame is the most top level window