Client Side Testing

Instructor: C. Pu (Ph.D., Assistant Professor)

Lecture 08

puc@marshall.edu

Introduction

- Client-Side testing is concerned with the execution of code on the client, typically natively within a web browser or browser plugin.

- The execution of code on the client-side is distinct from executing on the server and returning the subsequent content.

Testing for JavaScript Execution: Summary

- A JavaScript Injection vulnerability is a subtype of Cross Site Scripting (XSS) that involves the ability to inject arbitrary JavaScript code that is executed by the application inside the victim's browser.

- This vulnerability can have many consequences, like disclosure of a user's session cookies that could be used to impersonate the victim, or, more generally, it can allow the attacker to modify the page content seen by the victims or the application behavior.

Testing for JavaScript Execution: How to Test

- Such vulnerability occurs when the application lacks of a proper user supplied input and output validation.
- JavaScript is used to dynamically populate web pages, this injection occur during this content processing phase and consequently affect the victim.
- When trying to exploit this kind of issues, consider that some characters are treated differently by different browsers.

Testing for JavaScript Execution: How to Test

- https://www.youtube.com/watch?v=Pavl4MYFfSw

Testing for HTML Injection: Summary

- HTML injection is a type of injection issue that occurs when a user is able to control an input point and is able to inject arbitrary HTML code into a vulnerable web page.

- This vulnerability can have many consequences, like disclosure of a user’s session cookies that could be used to impersonate the victim, or, more generally, it can allow the attacker to modify the page content seen by the victims.

Testing for HTML Injection: How to Test

- This vulnerability occurs when the user input is not correctly sanitized and the output is not encoded.
- An injection allows the attacker to send a malicious HTML page to a victim.
- The targeted browser will not be able to distinguish (trust) the legit from the malicious parts and consequently will parse and execute all as legit in the victim context.

Testing for HTML Injection: How to Test

- There is a wide range of methods and attributes that could be used to render HTML content.
- If these methods are provided with an untrusted input, then there is a high risk of XSS, specifically an HTML injection one.
- Malicious HTML code could be injected for example via innerHTML, that is used to render user inserted HTML code.
- If strings are not correctly sanitized, the problem could lead to XSS based HTML injection.
- Another method could be document.write()

Testing for HTML Injection: How to Test

- When trying to exploit this kind of issues, consider that some characters are treated differently by different browsers.
- The innerHTML property sets or returns the inner HTML of an element.
- An improper usage of this property, that means lack of sanitization from untrusted input and missing output encoding, could allow an attacker to inject malicious HTML code.

Testing for HTML Injection: How to Test

- https://www.youtube.com/watch?v=0M711nyRgn0

Testing for Clickjacking: Summary

- "Clickjacking" (which is a subset of the "UI redressing") is a malicious technique that consists of deceiving a web user into interacting (in most cases by clicking) with something different to what the user believes they are interacting with.

- This type of attack, that can be used alone or in combination with other attacks, could potentially send unauthorized commands or reveal confidential information while the victim is interacting on seemingly harmless web pages.

- The term "Clickjacking" was coined by Jeremiah Grossman and Robert Hansen in 2008.

Testing for Clickjacking: Summary

- A Clickjacking attack uses seemingly innocuous features of HTML and Javascript to force the victim to perform undesired actions, such as clicking on a button that appears to perform another operation.

- This is a “client side” security issue that affects a variety of browsers and platforms.

Testing for Clickjacking: Summary

- To carry out this type of technique the attacker has to create a seemingly harmless web page that loads the target application through the use of an iframe (suitably concealed through the use of CSS code).
- Once this is done, the attacker could induce the victim to interact with his fictitious web page by other means (like for example social engineering).
- Like others attacks, an usual prerequisite is that the victim is authenticated against the attacker’s target website.

Testing for Clickjacking: Summary

- Once the victim is surfing on the fictitious web page, he thinks that he is interacting with the visible user interface, but effectively he is performing actions on the hidden page.

- Since the hidden page is an authentic page, the attacker can deceive users into performing actions which they never intended to perform through an “ad hoc” positioning of the elements in the web page.

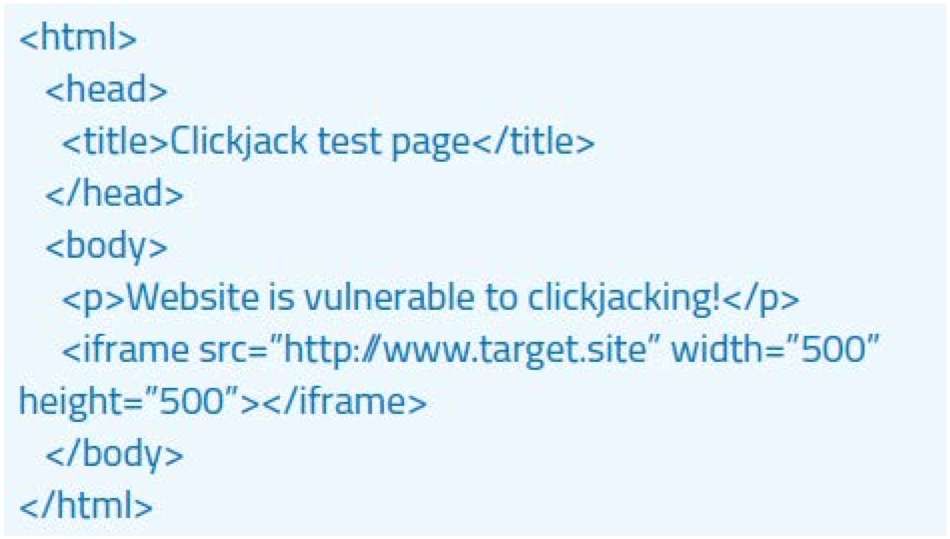
- The power of this method is due to the fact that the actions performed by the victim are originated from the authentic target web page (hidden but authentic).

Testing for Clickjacking: How to Test

- As mentioned above, this type of attack is often designed to allow an attacker to induce user’s actions on the target site.
- We have to discover if the website that we are testing has no protections against clickjacking attacks or, if the developers have implemented some forms of protection, if these techniques are liable to bypass.
- Once we know that the website is vulnerable, we can create a “proof of concept” to exploit the vulnerability.

Testing for Clickjacking: How to Test

- The first step to discover if a website is vulnerable, is to check if the target web page could be loaded into an iframe.
- To do this you need to create a simple web page that includes a frame containing the target web page.
- The HTML code to create this testing web page is displayed in the following snippet:



```
<html>
  <head>
    <title>Clickjack test page</title>
  </head>
  <body>
    <p>Website is vulnerable to clickjacking!</p>
    <iframe src="http://www.target.site" width="500"
height="500"></iframe>
  </body>
</html>
```

Testing for Clickjacking: How to Test

- Result Expected:
  - If you can see both the text “Website is vulnerable to clickjacking!” at the top of the page and your target web page successfully loaded into the frame, then your site is vulnerable and has no type of protection against Clickjacking attacks.
  - Now you can directly create a “proof of concept” to demonstrate that an attacker could exploit this vulnerability.