Report 09

URL: accounts.firefox.com

Sub resource Integrity (SRI) Hash Invalid

MEDIUM

The Subresource Integrity (SRI) Hash Invalid vulnerability refers to a security issue where the computed hash for a subresource is found to be invalid. SRI is a web security feature that allows web developers to ensure the integrity of externally hosted resources, such as scripts or stylesheets provided by third-party sources like Content Delivery Networks (CDNs). SRI protects against the possibility of these external resources being tampered with or modified during transit or by an attacker. It does this by comparing the expected hash value of the resource, as specified in the HTML element's integrity attribute, with the computed hash value of the fetched resource. If the hash values do not match, it indicates that the resource has been altered or tampered with, and SRI alerts the browser to block the resource from loading.

Affected URL: https://accounts.firefox.com/complete_signin

Identified Sub Resource(s) : https://accounts-static.cdn.mozilla.net/styles/1cd4f689.tailwind.out.css

Affected Components

- HTML Elements: The vulnerability primarily affects HTML elements that reference external subresources, such as scripts or stylesheets. Specifically, the elements where the integrity attribute is used to declare the expected hash value of the subresource are susceptible to the SRI Hash Invalid vulnerability.

- External Resources: The vulnerability impacts subresources that are hosted externally, typically provided by third-party sources like Content Delivery Networks (CDNs) or other remote servers. These resources include JavaScript files, CSS stylesheets, or other files referenced by the HTML elements.

- Web Browsers: The SRI Hash Invalid vulnerability affects the behavior of web browsers. When encountering HTML elements with the integrity attribute, browsers are responsible for comparing the computed hash value of the subresource with the expected hash value. If the computed hash is found to be invalid, the browser may block the subresource from loading.

- Web Application Frameworks or Libraries: If a web application framework or library incorporates SRI as part of its security measures, the vulnerability can impact the usage of SRI within those frameworks or libraries. Developers relying on these components to handle subresource integrity may need to address the vulnerability within the framework or library implementation.

Impact Assessment

- Compromised Integrity: The primary purpose of SRI is to ensure the integrity of externally hosted subresources. When the computed hash value for a subresource is invalid, it indicates that the integrity of the resource cannot be verified. This compromises the trustworthiness and reliability of the subresource, as it may have been tampered with or modified during transit.

- Security Risks: The SRI Hash Invalid vulnerability opens the door for potential security risks. If an attacker can manipulate the content of a compromised subresource and make the hash value invalid, it could lead to various attacks. For example, an attacker could inject malicious code into a script subresource, potentially resulting in cross-site scripting (XSS) attacks or unauthorized data manipulation.

- Browser Behavior: Browsers play a crucial role in enforcing SRI. When a browser encounters an HTML element with the integrity attribute, it compares the computed hash value with the expected value. If the hash is invalid, the browser may block the subresource from loading, impacting the functionality or appearance of the web page.

This behavior can affect the user experience and disrupt the intended functionality of the web application.

- Third-Party Dependencies: Many web applications rely on external resources provided by third-party sources, such as CDNs or libraries. The SRI Hash Invalid vulnerability exposes the risks associated with trusting these external dependencies. If a subresource fetched from a third-party source has an invalid hash value, it raises concerns about the overall security and trustworthiness of the web application, especially if critical functionalities depend on these resources.

- Trust and Reputation: The presence of the SRI Hash Invalid vulnerability can undermine user trust and damage the reputation of the web application or website. Users expect their interactions with websites to be secure and reliable. If a web application fails to ensure the integrity of externally hosted resources, it may lead to a loss of trust, reduced user engagement, and potential damage to the organization's reputation.

Steps to Reproduce

- Identify a web application or webpage that implements SRI for loading external subresources, such as scripts or stylesheets.

- Analyze the HTML source code of the webpage and locate the HTML elements that reference the external subresources. Look for script or link elements that include the integrity attribute, specifying the expected hash value.

- Intercept the network traffic between the web server and the client browser. This can be done using a network analysis tool or browser developer tools.

- Modify the content of the subresource referenced in the HTML element. This could involve injecting additional code, altering the existing code, or manipulating the resource in any way that would result in a different content than what was expected.

- Recalculate the hash value of the modified subresource using the appropriate hash algorithm (e.g., sha256, sha384, or sha512).

- Replace the original subresource on the server with the modified version, ensuring that the hash value specified in the integrity attribute does not match the computed hash of the modified subresource.

- Reload the webpage or trigger the action that would initiate the loading of the subresource with the modified content.

- Observe the behavior of the browser when it encounters the modified subresource. If the browser correctly implements SRI, it should compare the computed hash value with the expected hash value specified in the integrity attribute. If the hash values do not match, the browser should block the subresource from loading, indicating the presence of the SRI Hash Invalid vulnerability.

Proposed Mitigation or Fix

✓ Validate Subresource Integrity: Implement a comprehensive validation process for subresources that includes verifying the integrity of each subresource using the specified hash algorithm. This validation should be performed on the server-side before serving the subresource to clients.

✓ Secure Hash Algorithm: Ensure that a secure hash algorithm, such as sha256, sha384, or sha512, is used for calculating the hash values of subresources. These algorithms provide stronger cryptographic properties compared to weaker algorithms like MD5 or SHA-1.

✓ Hash Generation: Generate the hash values accurately and securely. Use reliable and trusted methods or tools to compute the hash values for subresources. Any changes made to the subresource content should result in a different hash value.

- ✓ Integrity Attribute: Include the integrity attribute in the HTML elements that reference external subresources, such as scripts or stylesheets. This attribute should contain the correct hash value calculated for each subresource.

- ✓ Cross-Origin Resource Sharing (CORS): Implement Cross-Origin Resource Sharing (CORS) policies to restrict access to external subresources. Set appropriate Access-Control-Allow-Origin headers to limit requests from specific origins and prevent unauthorized modifications to the subresources.

- ✓ Content Security Policy (CSP): Utilize Content Security Policy (CSP) headers to specify the allowed sources of scripts and other subresources. This provides an additional layer of protection by restricting the loading of resources from unauthorized or untrusted locations.

Using Subresource Integrity is simply to add integrity attribute to the script tag along with a base64 encoded cryptographic hash value.

<script src="https://code.jquery.com/jquery-2.1.4.min.js" integrity="sha384-R4/ztc4ZlRqWjqIuvf6RX5yb/v90qNGx6fS48N0tRxiGkqveZETq72KgDVJCp2TC" crossorigin="anonymous"></script>

The hash algorithm must be one of sha256, sha384 or sha512, followed by a '-' character.