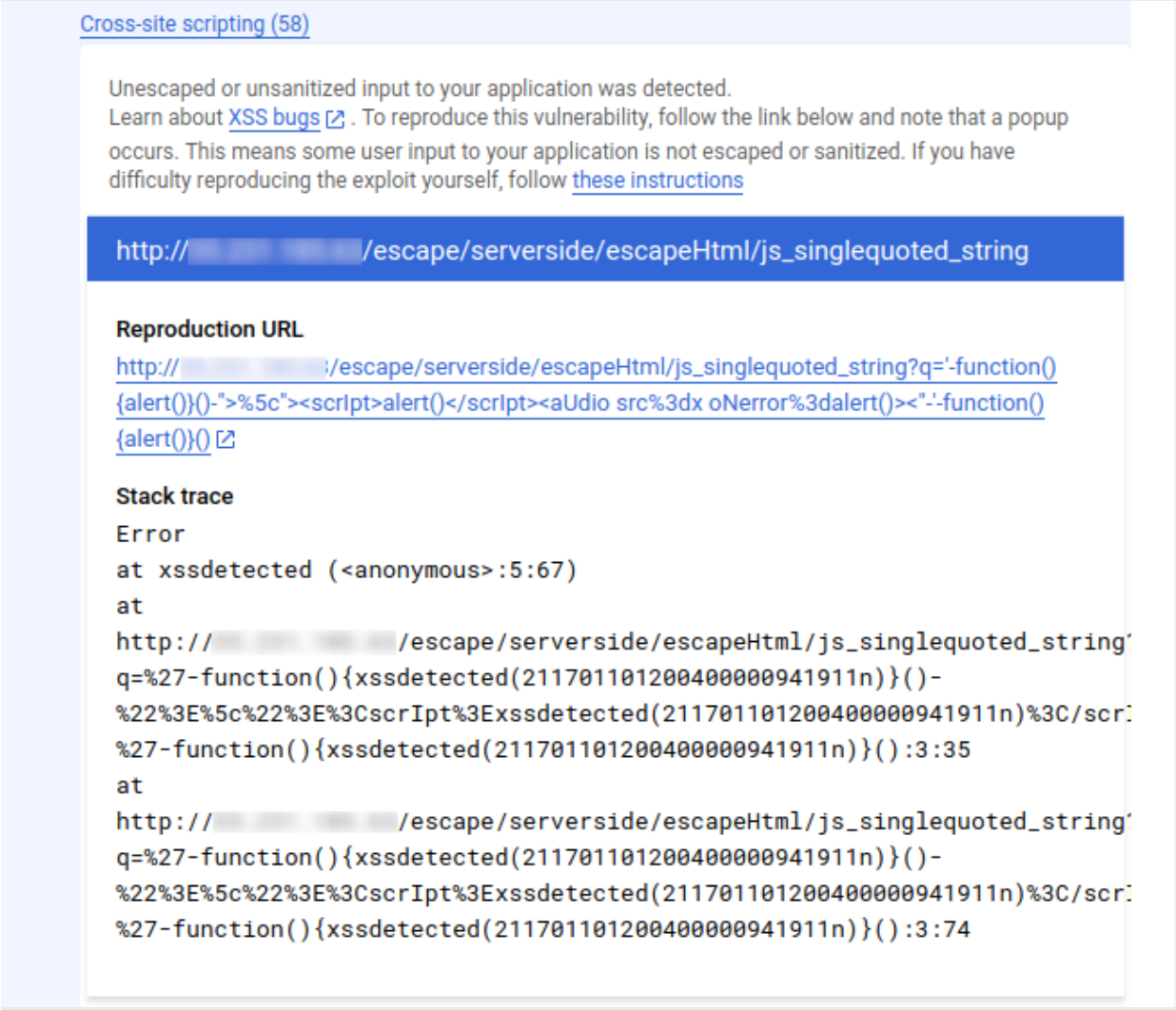
This page describes how to interpret, reproduce, and remediate Web Security Scanner findings.

Web Security Scanner cross-site scripting (XSS) injection testing simulates an injection attack by inserting a benign test string into user-editable fields and then performing a variety of user actions. Custom detectors observe the browser and DOM during this test to determine if an injection was successful and assess its potential for exploitation.

If the JavaScript contained within the test string cleanly executes, it starts the Chrome debugger.

Following is an example of an XSS alert in the vulnerable parameter `q=`



Because the test string was able to execute, we know that it's possible to inject and run JavaScript on this page. If an attacker found this issue, they could execute JavaScript of their choosing as the user (victim) who clicks on a malicious link.

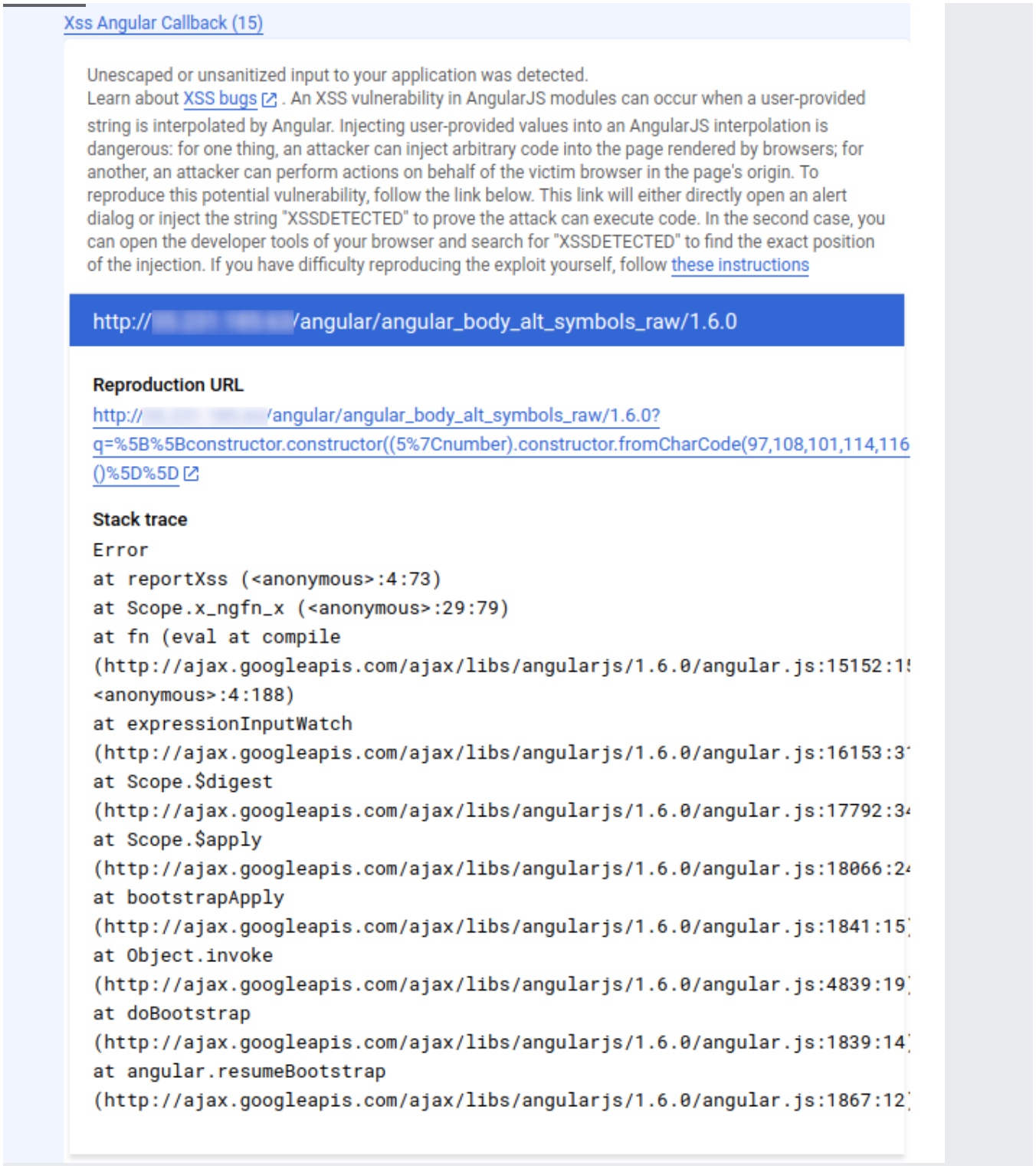
In some circumstances, the application under test might modify the test string before it is parsed by the browser. For example, the application might validate the input or limit the size of a field. When the browser tries to run this modified test string, it is likely to break and throw a JavaScript execution error. This is an an injection issue, but it might not be possible to exploit it. You need to manually verify if the test string modifications can be evaded to confirm if the issue is an XSS vulnerability. For detailed information, see Cross-site scripting (https://www.google.com/about/appsecurity/learning/xss/).

There are various ways to fix this problem. The recommended way is to escape all output and use a templating system that supports contextual auto-escaping.

A cross-site scripting (XSS) vulnerability in AngularJS modules can occur when a user-provided string is interpolated by Angular. Injecting user-provided values into an AngularJS interpolation can allow the following attacks:

- An attacker can inject arbitrary code into the page rendered by browsers.
- An attacker can perform actions on behalf of the victim browser in the page's origin.

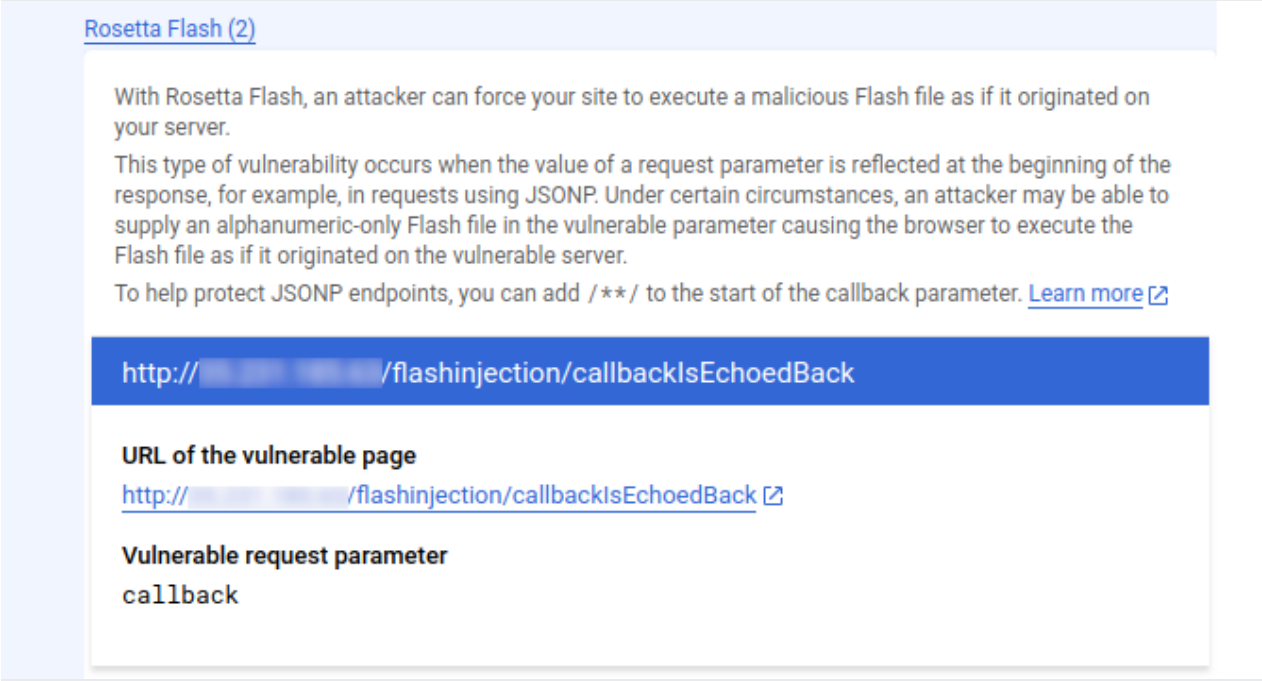
Following is an example of a breakage alert that shows an Angular XSS injection issue.



To reproduce this potential vulnerability, follow the Reproduction URL link in the Google Cloud Console after you run the scan. This link will either directly open an alert dialog or inject the string `XSSDETECTED` to prove that the attack can execute code. In the case of injection, you can open the developer tools of your browser and search for `XSSDETECTED` to find the exact position of the injection.

Web Security Scanner might find a parameter that is reflected back at the beginning of a response. This is also known as Rosetta Flash. Under certain circumstances, an attacker can cause the browser to execute the response as if it were a Flash file provided by the vulnerable web application.

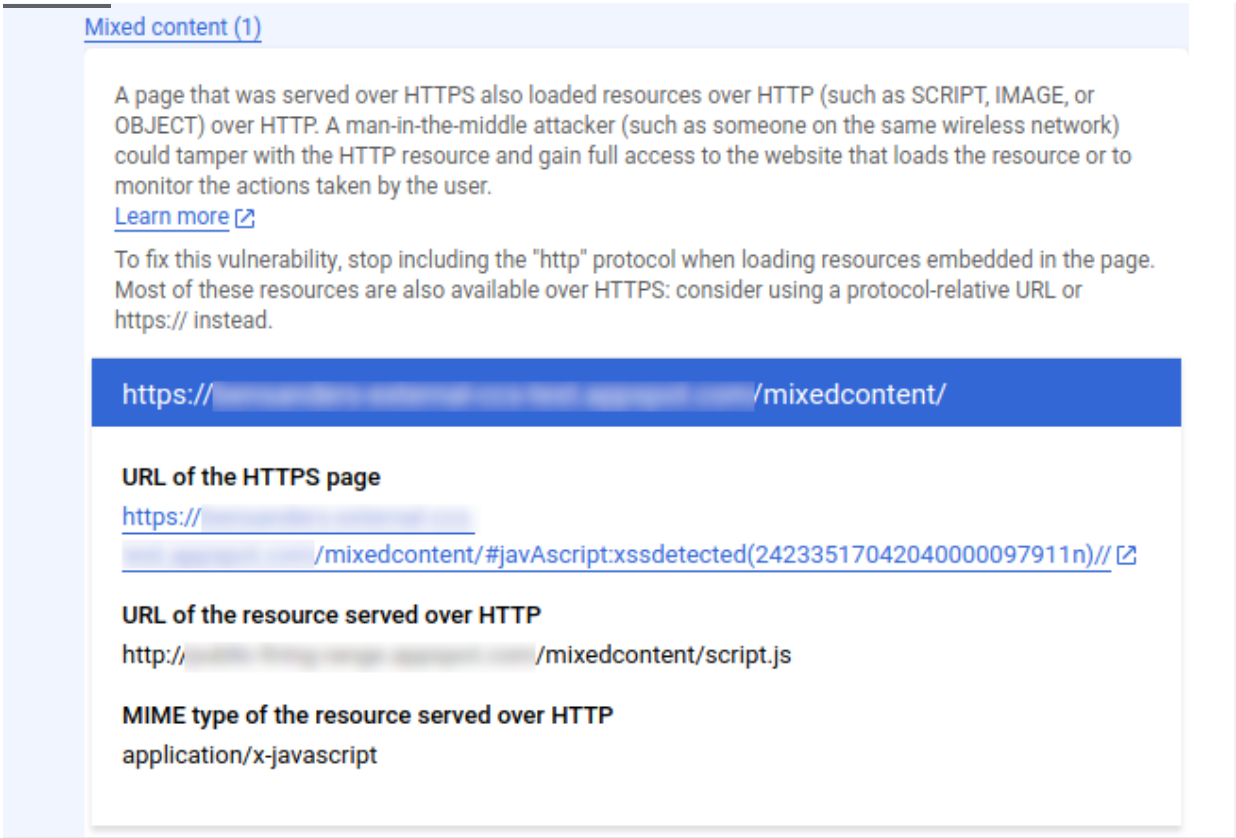
Following is an example of a Flash injection alert in the parameter `callback=`



To fix this, don't include user controllable data at the start of an HTTP response.

Web Security Scanner passively observes the HTTP traffic and detects when a request for a JavaScript or CSS file is performed over HTTP while in the context of an HTTPS page.

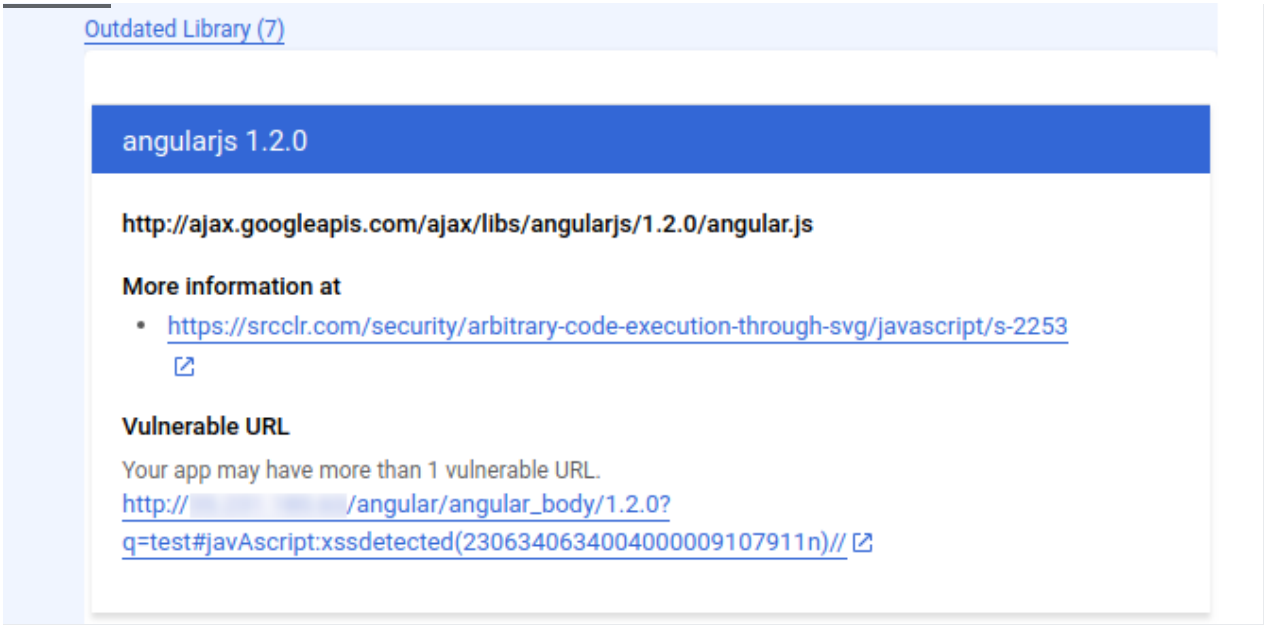
Following is an example of a mixed content alert in an HTTPS page `attribute_script`, including an HTTP resource from `http://irrelevant.google.com`



To fix this, use relative HTTP links, for example, replace `http://` with `//`.

Web Security Scanner might find that the version of an included library is known to contain a security issue. This is a signature-based scanner that attempts to identify the version of the library in use and checks this against a known list of vulnerable libraries. False positives are possible if the version detection fails or if the library has been manually patched.

Following is an example of an outdated library alert due to the use of `jquery-1.8.1.js`.



Fix this by updating to a known secure version of the included library.

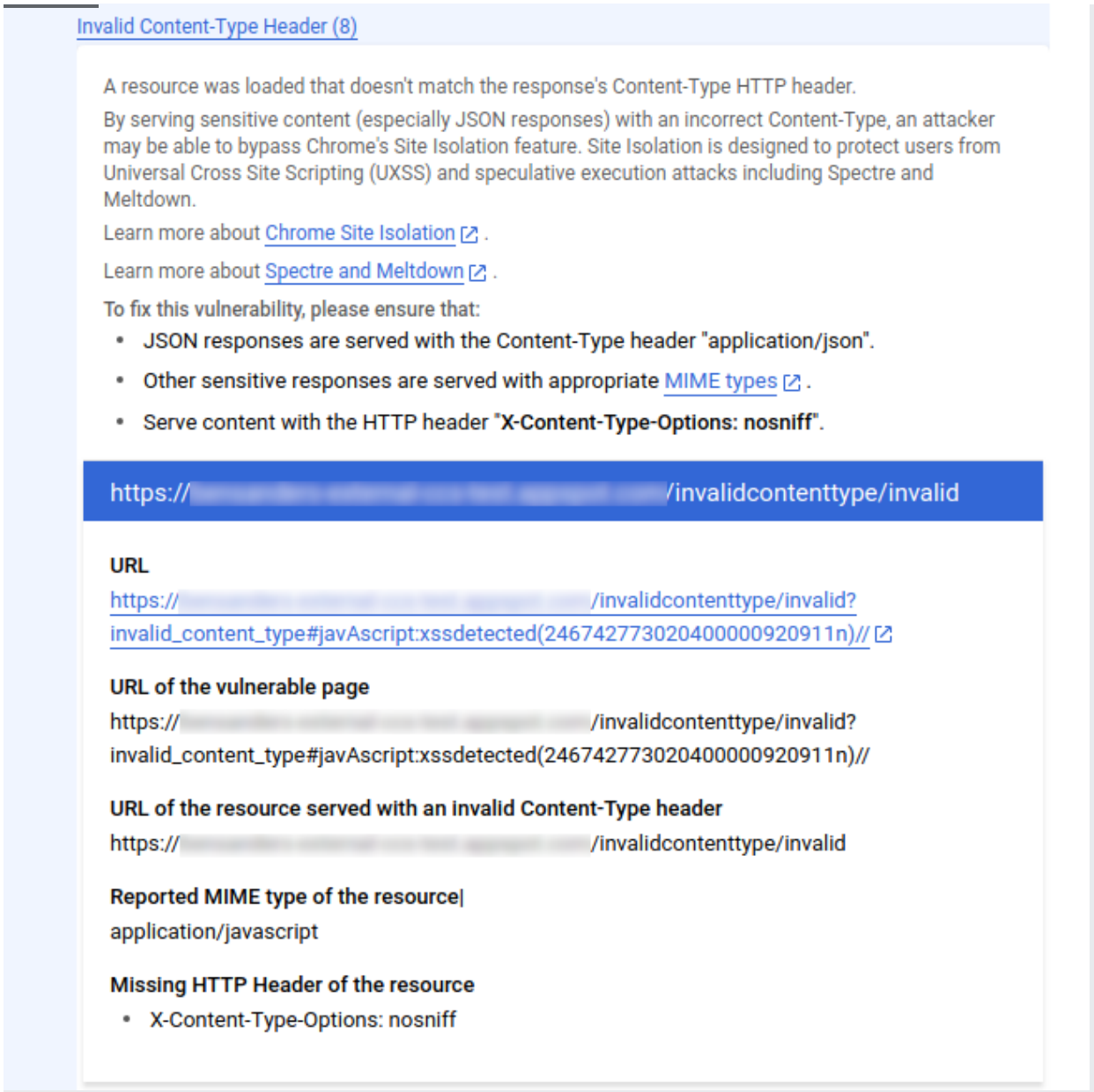
Web Security Scanner might find that the application appears to be transmitting a password field in clear text.

To protect sensitive information that passes between client and server, always take the following precautions:

- Use TLS/SSL certificates.
- Always use HTTPS on pages that include password fields.
- Make sure that form action attributes always point to an HTTPS URL.

Web Security Scanner might find that a resource that was loaded and doesn't match the response's Content-Type HTTP header.

Following is an example of an invalid Content-Type header alert.

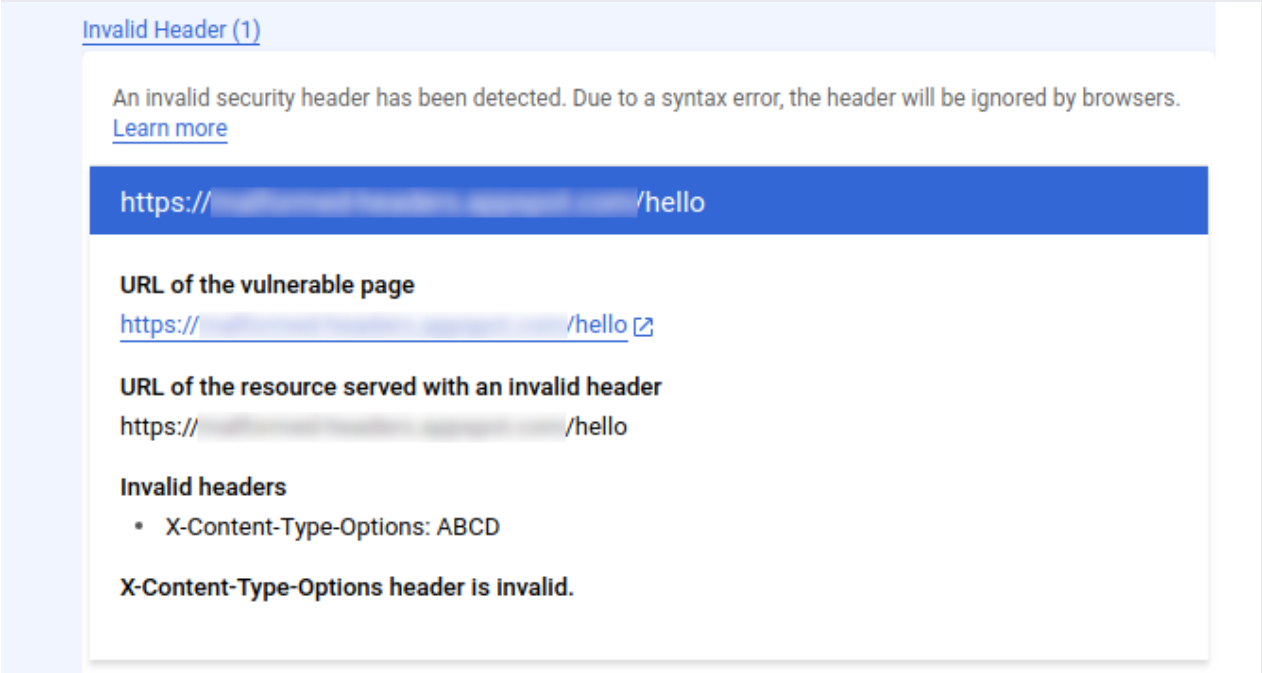


To fix this vulnerability, ensure that:

- JSON responses are served with the Content-Type header `application/json`
- Other sensitive responses are served with appropriate MIME types
- Serve content with the HTTP header `X-Content-Type-Options: nosniff`

Web Security Scanner might find that a security header has a syntax error. As a result, the header is ignored by browsers.

Following is an example of an invalid security header finding.



Valid headers are described in the following sections.

A valid referrer policy (https://www.w3.org/TR/referrer-policy/#referrer-policies) contains one of the following values:

- An empty string
- `no-referrer`
- `no-referrer-when-downgrade`
- `same-origin`
- `origin`
- `strict-origin`
- `origin-when-cross-origin`
- `strict-origin-when-cross-origin`
- `unsafe-url`

A valid X-Frame-Options header can only have the following values:

- `DENY`: disallow all framing
- `SAMEORIGIN`: allow framing if the top-level URL is same origin
- `ALLOW-FROM URL`

`ALLOW-FROM URL` is not supported by Chrome. Multiple X-Frame-Options are not allowed.

A valid X-Content-Type-Options header can only have one value: `nosniff`.

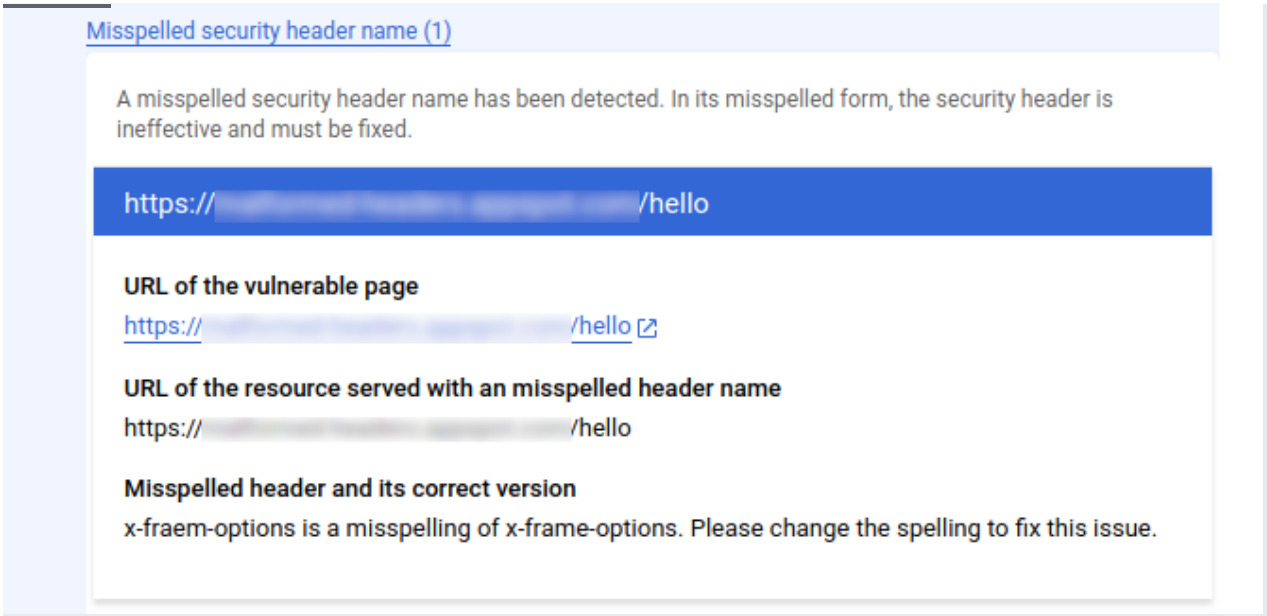
A valid X-XSS-Protection header must start with either `0` ("disable") or `1` ("enable"). Then, only if you enable the protection, you can add up to two options:

- `mode=block` will show a blank page instead of filtering the XSS
- `report=URL` will send reports to `URL`

Options need to be separated by semicolons, for example `1; mode=block; report=URI`. Make sure that you don't have a trailing semicolon.

Web Security Scanner might find a misspelled security header name. In its misspelled form, the security header is ineffective and must be fixed.

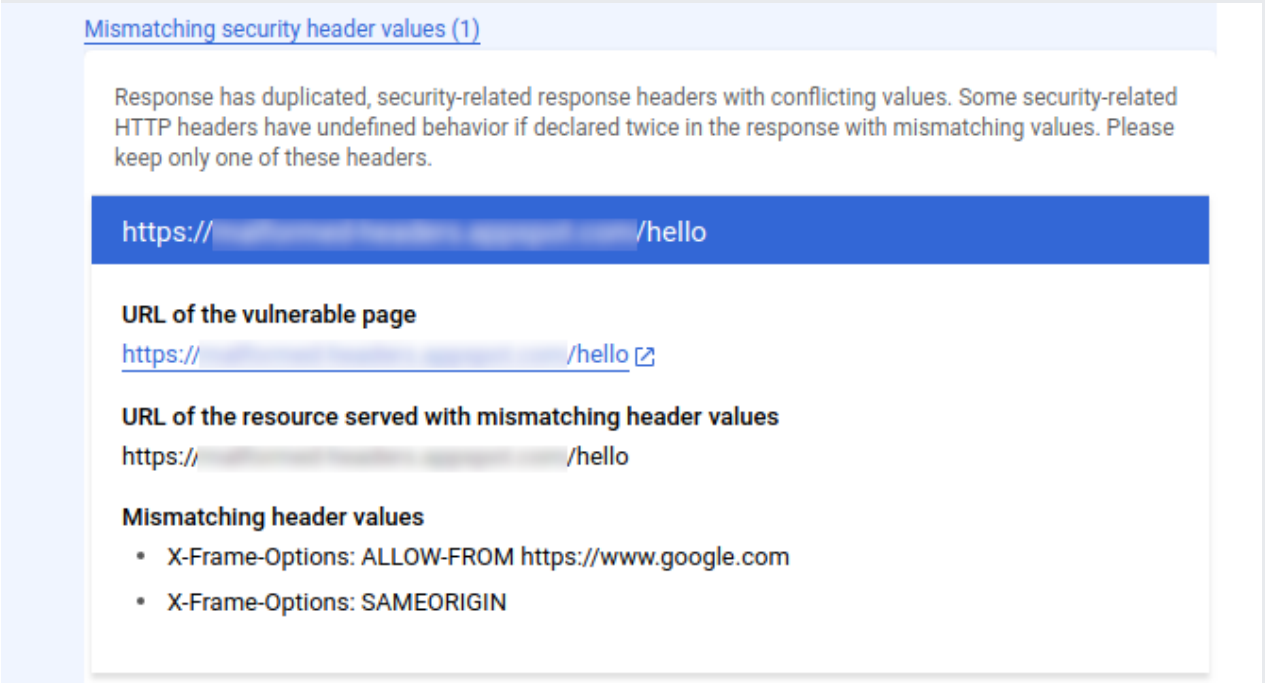
Following is an example of a misspelled security header name finding.



To reproduce this vulnerability, check for the misspelling in the network tab of your browser's developer tools.

Web Security Scanner might find that the response has duplicated, security-related response headers with conflicting values. Some security-related HTTP headers have undefined behavior if declared twice in the response with mismatching values.

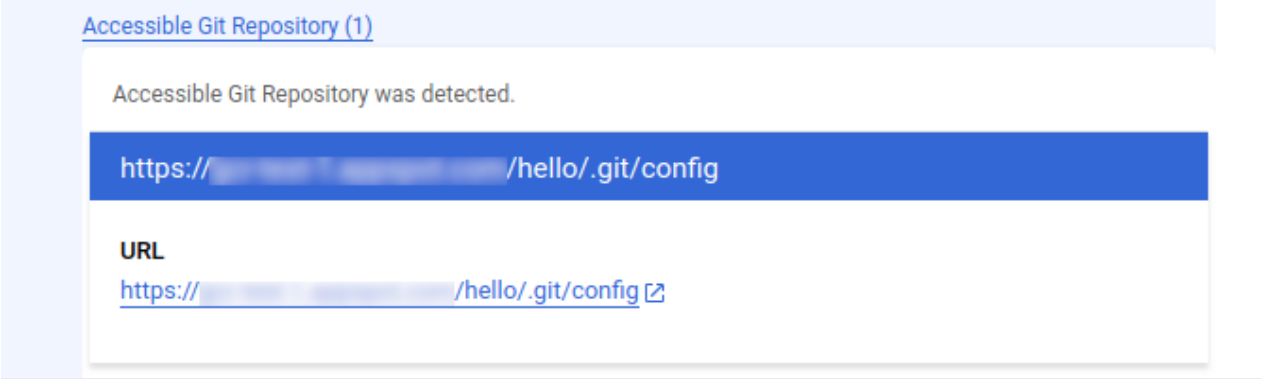
Following is an example of a mismatching security header values finding.



To fix this vulnerability, keep only one of these mismatching headers.

Web Security Scanner might find an accessible Git or SVN repository in the application. This can lead to configuration and source code leaks.

Following is an example of an accessible Git repository finding.



To reproduce the vulnerability, click the reproduction URL in the finding report.

When Web Security Scanner reports an issue, you need to verify the issue's location. Do this with a browser that has XSS protection turned off. It's best to use a separate test instance of Chrome, but you can use most modern browsers that allow you to disable XSS protection.

To disable XSS protection in Chrome:

- If you use Linux, invoke the Linux Chrome command as follows:

- If you use macOS, invoke the Chrome command as follows:

Content Security Policy (CSP) enforcement might still prevent the JavaScript code from running. This can make it more difficult to reproduce the XSS. If you experience this issue, check the browser log console for details about the CSP violation that occurred.

ant: Because your test browser instance has XSS and other safety measures disabled, do not use it for anything oth esting your own security issues.