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=

oss-site scripting (	58)
Learn about XSS to occurs. This mean	sanitized input to your application was detected. <u>bugs</u> [2] . To reproduce this vulnerability, follow the link below and note that a popup ns some user input to your application is not escaped or sanitized. If you have cing the exploit yourself, follow <u>these instructions</u>
http://	/escape/serverside/escapeHtml/js_singlequoted_string
Reproduction U	RL
http://	<pre>//escape/serverside/escapeHtml/js_singlequoted_string?q='-function()</pre>
{alert()}()-">%5c	:"> <script>alert()</script> <audio onerror%3dalert()="" src%3dx="">&lt;"-'-function()</audio>
<u>{alert()}()</u> [2]	
Stack trace	
Error	
at xssdetect	ted ( <anonymous>:5:67)</anonymous>
at	
http:// q=%27-functi	<pre>/escape/serverside/escapeHtml/js_singlequoted_string ion(){xssdetected(211701101200400000941911n)}()-</pre>
%22%3E%5c%22	2%3E%3CscrIpt%3Exssdetected(211701101200400000941911n)%3C/scr
%27-functior at	n(){xssdetected(211701101200400000941911n)}():3:35
	/escape/serverside/escapeHtml/js_singlequoted_string
http:// a=%27-functi	
q=%27-functi	ion(){xssdetected(211701101200400000941911n)}()- 2%3E%3CscrIpt%3Exssdetected(211701101200400000941911n)%3C/scr

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 <u>Cross-site scripting</u> (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.

## Xss Angular Callback (15)

Unescaped or unsanitized input to your application was detected. Learn about <u>XSS bugs</u> [2]. An XSS vulnerability in AngularJS modules can occur when a user-provided string is interpolated by Angular. Injecting user-provided values into an AngularJS interpolation is dangerous: for one thing, an attacker can inject arbitrary code into the page rendered by browsers; for another, an attacker can perform actions on behalf of the victim browser in the page's origin. To reproduce this potential vulnerability, follow the link below. This link will either directly open an alert dialog or inject the string "XSSDETECTED" to prove the attack can execute code. In the second case, you can open the developer tools of your browser and search for "XSSDETECTED" to find the exact position of the injection. If you have difficulty reproducing the exploit yourself, follow these instructions

	http://	/angular/an	gular_body_alt_s	symbols_raw/1.6.0
--	---------	-------------	------------------	-------------------

Reproduction URL
http:// /angular/angular_body_alt_symbols_raw/1.6.0?
q=%5B%5Bconstructor.constructor((5%7Cnumber).constructor.fromCharCode(97,108,101,114,116
()%5D%5D [2]
Stack trace
Error
at reportXss ( <anonymous>:4:73)</anonymous>
at Scope.x_ngfn_x ( <anonymous>:29:79)</anonymous>
at fn (eval at compile
(http://ajax.googleapis.com/ajax/libs/angularjs/1.6.0/angular.js:15152:1
<anonymous>:4:188)</anonymous>
at expressionInputWatch
(http://ajax.googleapis.com/ajax/libs/angularjs/1.6.0/angular.js:16153:3
at Scope.\$digest
<pre>(http://ajax.googleapis.com/ajax/libs/angularjs/1.6.0/angular.js:17792:34 at Scope.\$apply</pre>
(http://ajax.googleapis.com/ajax/libs/angularjs/1.6.0/angular.js:18066:24
at bootstrapApply
(http://ajax.googleapis.com/ajax/libs/angularjs/1.6.0/angular.js:1841:15)
at Object.invoke
(http://ajax.googleapis.com/ajax/libs/angularjs/1.6.0/angular.js:4839:19
at doBootstrap
(http://ajax.googleapis.com/ajax/libs/angularjs/1.6.0/angular.js:1839:14)
at angular.resumeBootstrap
(http://ajax.googleapis.com/ajax/libs/angularjs/1.6.0/angular.js:1867:12)

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=

With Rosetta Fl your server.	ash, an attacker can force your site to execute a malicious Flash file as if it originated on
response, for e supply an alpha	nerability occurs when the value of a request parameter is reflected at the beginning of the cample, in requests using JSONP. Under certain circumstances, an attacker may be able to numeric-only Flash file in the vulnerable parameter causing the browser to execute the t originated on the vulnerable server.
To help protect	JSONP endpoints, you can add /**/ to the start of the callback parameter. Learn more
http://	/flashinjection/callbackIsEchoedBack
URL of the vu	Inerable page
URL of the vu	Inerable page

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.

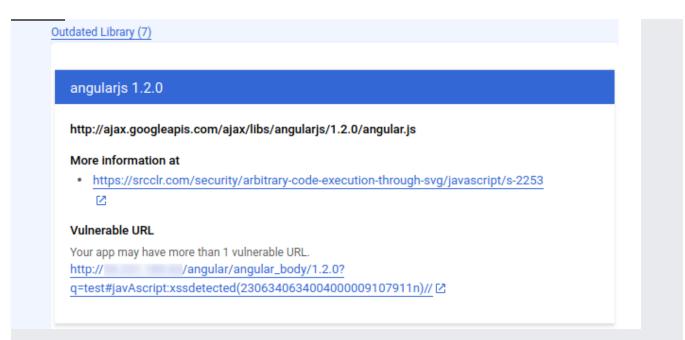
Following is an example of a mixed content alert in an HTTPS page attribute\_script, including an HTTP resource from http://irrelevant.google.com

	erved over HTTPS also loaded resources over HTTP (such as SCRIPT, IMAGE, or
could tamper with	TP. A man-in-the-middle attacker (such as someone on the same wireless network) the HTTP resource and gain full access to the website that loads the resource or to is taken by the user.
Learn more 🛛	
	bility, stop including the "http" protocol when loading resources embedded in the page. ources are also available over HTTPS: consider using a protocol-relative URL or
incipelity increase.	
https://	/mixedcontent/
	/mixedcontent/
https://	
https:// URL of the HTT	
https://	PS page
https:// URL of the HTT	
https:// URL of the HTT https://	PS page
https:// URL of the HTT https:// URL of the reso	PS page /mixedcontent/#javAscript:xssdetected(24233517042040000097911n)// [2] urce served over HTTP
https:// URL of the HTT https://	PS page /mixedcontent/#javAscript:xssdetected(24233517042040000097911n)// 🖸
https:// URL of the HTT https:// URL of the reso http://	PS page /mixedcontent/#javAscript:xssdetected(24233517042040000097911n)// [2] urce served over HTTP

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.

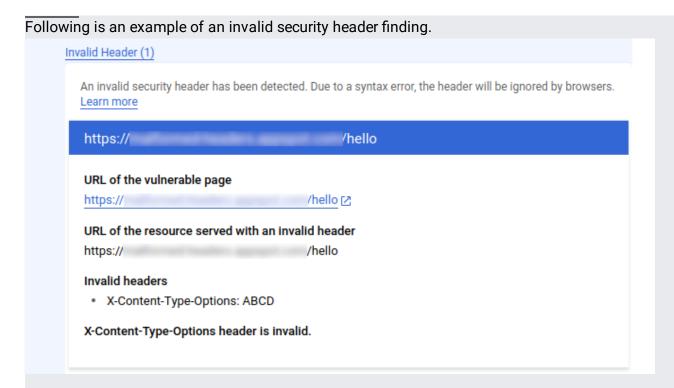
Following is an example of an invalid Content-Type header alert.

A resource was loaded that o	doesn't match the response's Content-Type HTTP header.
may be able to bypass Chron	(especially JSON responses) with an incorrect Content-Type, an attacker ne's Site Isolation feature. Site Isolation is designed to protect users from g (UXSS) and speculative execution attacks including Spectre and
Learn more about Chrome Si	ite Isolation 🛯 .
Learn more about <u>Spectre ar</u>	nd Meltdown 🔁 .
To fix this vulnerability, pleas	se ensure that:
<ul> <li>JSON responses are s</li> </ul>	erved with the Content-Type header "application/json".
<ul> <li>Other sensitive respon</li> </ul>	ises are served with appropriate MIME types 🖸 .
<ul> <li>Serve content with the</li> </ul>	HTTP header "X-Content-Type-Options: nosniff".
https://	/invalidcontenttype/invalid
URL	
URL https://	/invalidcontenttype/invalid?
https://	/invalidcontenttype/invalid? Ascript:xssdetected(246742773020400000920911n)// 🖸
https:// invalid_content_type#javA	Ascript:xssdetected(246742773020400000920911n)// [2
https:// invalid_content_type#javA URL of the vulnerable pag	Ascript:xssdetected(246742773020400000920911n)// [2]
https:// invalid_content_type#javA URL of the vulnerable pag https://	Ascript:xssdetected(246742773020400000920911n)// 🗗 je /invalidcontenttype/invalid?
https:// invalid_content_type#javA URL of the vulnerable pag https://	Ascript:xssdetected(246742773020400000920911n)// [2]
https:// invalid_content_type#javA URL of the vulnerable pag https:// invalid_content_type#javA	Ascript:xssdetected(246742773020400000920911n)// 🗗 je /invalidcontenttype/invalid?
https:// invalid_content_type#javA URL of the vulnerable pag https:// invalid_content_type#javA	Ascript:xssdetected(246742773020400000920911n)// 🗗
https:// invalid_content_type#javA URL of the vulnerable pag https:// invalid_content_type#javA URL of the resource serve https://	Ascript:xssdetected(246742773020400000920911n)// [2] ge /invalidcontenttype/invalid? Ascript:xssdetected(246742773020400000920911n)// ed with an invalid Content-Type header /invalidcontenttype/invalid
https:// invalid_content_type#javA URL of the vulnerable pag https:// invalid_content_type#javA URL of the resource serve https:// Reported MIME type of th	Ascript:xssdetected(246742773020400000920911n)// [2] ge /invalidcontenttype/invalid? Ascript:xssdetected(246742773020400000920911n)// ed with an invalid Content-Type header /invalidcontenttype/invalid
https:// invalid_content_type#javA URL of the vulnerable pag https:// invalid_content_type#javA URL of the resource serve https:// Reported MIME type of th application/javascript	Ascript:xssdetected(246742773020400000920911n)// [2] ge /invalidcontenttype/invalid? Ascript:xssdetected(246742773020400000920911n)// ed with an invalid Content-Type header /invalidcontenttype/invalid he resource
https:// invalid_content_type#javA URL of the vulnerable pag https:// invalid_content_type#javA URL of the resource serve https:// Reported MIME type of th	Ascript:xssdetected(246742773020400000920911n)// [2] ge /invalidcontenttype/invalid? Ascript:xssdetected(246742773020400000920911n)// ed with an invalid Content-Type header /invalidcontenttype/invalid ne resource  the resource

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.



Valid headers are described in the following sections.

A <u>valid referrer policy</u> (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.

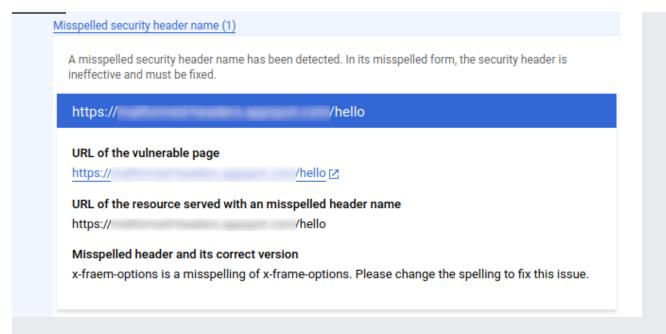
A valid X-XSS-Protection header must start with either 0 ("disable") or 1 ("enable"). Then, only if you enable the protection, you can add up to two options:

- 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.

ITTP headers have under eep only one of these he	l, security-related response headers with conflicting values. Some security-related fined behavior if declared twice in the response with mismatching values. Please eaders.
https://	/hello
URL of the vulnerable p	page
https://	/hello [2]
URL of the resource se	erved with mismatching header values
https://	/hello
Miemotohing hooder w	alues
Mismatching header va	
	ALLOW-FROM https://www.google.com

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.

Accessible Git Repository was detected.		
https://	/hello/.git/config	
URL		
https://	/hello/.git/config 🖸	

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.

t**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.