This page provides an overview of Web Security Scanner.

Web Security Scanner identifies security vulnerabilities in your App Engine, Google Kubernetes Engine (GKE), and Compute Engine web applications. It crawls your application, following all links within the scope of your starting URLs, and attempts to exercise as many user inputs and event handlers as possible. Currently, Web Security Scanner only supports public URLs and IPs that aren't behind a firewall.

The scanner currently detects the following:

- Cross-site scripting (XSS)
- Flash injection
- Mixed-content
- Clear text passwords
- Usage of insecure JavaScript libraries

Web Security Scanner currently supports the App Engine standard environment and App Engine flexible environments, Compute Engine instances, and GKE resources.

Web Security Scanner is designed to complement your existing secure design and development processes. To avoid distracting you with false positives, Web Security Scanner errs on the side of under reporting and doesn't display low confidence alerts. It does not replace a manual security review, and it does not guarantee that your application is free from security flaws. For more information on web security, see the OWASP Top Ten Project (https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project).

Learn more about Google Cloud Security (/security/)

Google may use aggregated and anonymized data to improve the performance of Web Security Scanner and to analy
ılnerability trends. Google will not share information on specific issues or the security state of a scanned website wit
arty.

Some important things to be aware of when using Web Security Scanner:

- Because Web Security Scanner is undergoing continual improvements, a future scan might report issues that are not reported by the current scan.
- Some features or sections of your application might not be tested.
- Web Security Scanner attempts to activate every control and input it finds.
- If you expose state-changing actions for which your test account has permission, Web Security Scanner is likely to activate them. This might lead to undesirable results.

For information about the Cloud Identity and Access Management (Cloud IAM) roles that are available for Web Security Scanner, see Access Control (/security-scanner/docs/access-control).

The security scan does not execute immediately. It is queued and then executes later, possibly hours later depending on system load. After the scan starts to execute, the time it takes will depend on the size of your application. Large applications with many URLs might take several hours to complete.

Web Security Scanner has filters in place that restrict scan targets to the specific App Engine instance for which the scan is created. Entering URLs for a different App Engine project or an outside domain will result in an error message.

tant: Attempting to subvert or in any way direct traffic to out-of-scope URLs is a violation of the acceptable use policy
d/terms/aup).

Within your project, the Web Security Scanner automatically *attempts* to avoid logout URLs and other generic locations that may adversely affect a scan. However, to be sure, you can use the scan settings to manually exclude URLs (/security-scanner/docs/excluded-urls).

Because Web Security Scanner populates fields, pushes buttons, clicks links, and so on, you should use it with caution. Web Security Scanner could potentially activate features that change the state of your data or system, with undesirable results.

For example:

- In a blog application that allows public comments, Web Security Scanner might post test strings as comments on all your blog articles.
- In an email sign-up page, Web Security Scanner might generate large numbers of test emails.

Following are some techniques that you can use, separately or in combination, to avoid unwanted outcomes:

1. **Run scans in a test environment.** Set up a test environment by creating a separate App Engine project and loading your application and data there. If you use the `gcloud` command-line tool (/sdk/gcloud/), you can specify the target project as a command-line option when you upload your app.
2. **Use a test account.** Create a user account that doesn't have access to sensitive data or harmful operations, and use it when scanning your app. Many applications present a special workflow during a user's first-time login, like accepting terms, creating a profile, and so on. Because of the different workflow, a test account for an initial user can have different scan results than an established user account. It's best to scan with an account that is in the normal user state, after the first-time flow is complete.
3. **Block individual UI elements** that you do not want activated by applying the CSS class `inq-no-click`. Event handlers that are attached to this element aren't activated during crawling and testing, regardless of whether they are inline JavaScript, or attached using `addEventListener`, or attached by setting the appropriate event handler property.
4. **Use backup data.** Consider making a backup of your data before scanning.
5. **Excluded URLs.** You can specify URL patterns that won't be crawled or tested. For information on syntax, see Excluding URLs (/security-scanner/docs/excluded-urls).

Before you scan, carefully audit your application for any feature that might affect data, users, or systems beyond the desired scope of your scan.

- Read about the details of scan results (/security-scanner/docs/scan-result-details).
- Using the Web Security Scanner (/security-scanner/docs/scanning).