Google Cloud

Guide to web application security scanning

So far, you've learned about how organizations can effectively protect their cloud environments and comply with regulations using vulnerability remediation and posture management. Remember, vulnerability remediation is the process of identifying, assessing, and resolving security vulnerabilities in a cloud environment. And, posture management is the continuous process of monitoring, assessing, and maintaining the security stance of an organization's cloud resources. You also explored ways to identify gaps in cloud resources using vulnerability scans. These practices are essential for organizations to operate securely in the ever-evolving cloud landscape. In this reading, you'll learn about how to use the Web Security Scanner to scan web applications for vulnerabilities.

Web Security Scanner

One feature of the Security Command Center (SCC) is the Web Security Scanner, which identifies security vulnerabilities in your cloud environment. Web Security Scanner works by crawling your applications. It follows all links within the scope of the specified starting URLs and attempts to exercise as many user inputs as possible.

Web Security Scanner supports vulnerability categories in the OWASP® Top Ten, which is a regularly updated report of critical security risks for web applications.

Note: To create and run custom scans with Web Security Scanner, you must enable Web Security Scanner in the Security Command Center.

Create and run a scan

Here's how to create a scan using the Google Cloud console:

1. In the Google Cloud console, click the **Navigation** menu.
2. Click **Security > Web Security Scanner**, and then click **+ New scan**.

When you create a scan, you must specify its settings in the **New scan** form. These settings include the following:

- **Name**: Enter a unique name for the new scan.
- **Starting URLs**: Enter the URL(s) that you'd like the scanner to begin its scan on. The Web Security Scanner will crawl each application's URL from the starting URL and follow all links within the scope of the starting URL.
- **Excluded URLs**: Enter the URL(s) that you'd like to exclude from scanning.

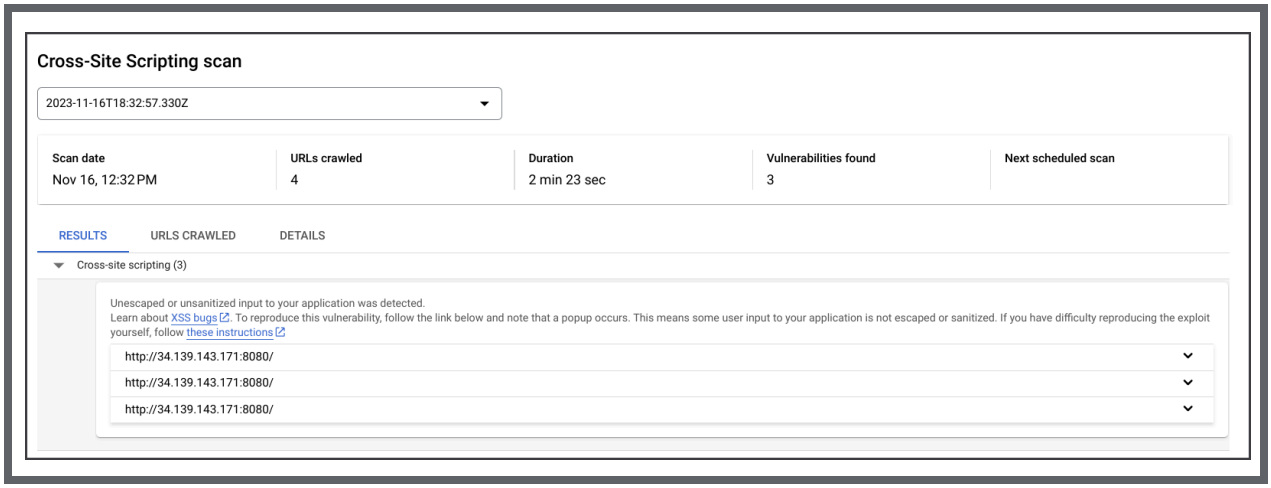
- **Authentication**: Specify whether the scanner authenticates to the web application when scanning for vulnerabilities.
- **Schedule**: Set up the scan to run daily, weekly, every two weeks, or every four weeks.
- **Export options**: Choose where to export the scan configurations and results.

To start the scan, click **Run Scan.** Once the scan begins to run, it is placed in a queue, which might delay the scan for several minutes.

Note: To complete the Web Security Scanner setup, you need the URL of an application that's been deployed. In the lab, you'll configure and deploy a simple Python application using Compute Engine.

Access scan results

Once the scan is complete, access the scan results in the Web Security Scanner page. The following screenshot displays the results of a Web Security Scanner scan. This example scan report reveals several cross-site scripting vulnerabilities on a web application due to unsanitized user input.



Cross-site scripting (XSS)

Cross-site scripting (XSS) is an injection attack that inserts code into a vulnerable website or web application. XSS attacks happen when data entered into a web application is included in dynamic content, and then that is sent to a web user without being validated for malicious content.

Here is an example of HTML code that can be injected into a web application's input. When the user's browser renders the page, it executes the script which appears as an alert message in the user's browser.

```
<script>alert('This is an XSS Injection')</script>
```

Web Security Scanner XSS injection testing simulates an injection attack by inputting a benign test string into user-editable fields, and then performing various user actions. Custom detectors observe the browser during this test to determine whether an injection was successful and assess its potential for exploitation.

Best practices for XSS mitigation

To address and prevent this type of vulnerability, web applications must sanitize user input before outputting it to the browser. This means that all user-supplied data needs to be validated.

For example, the following lines of HTML code are vulnerable to XSS because it simply trusts and accepts any user-supplied data without properly validating it.

```
output_string = input_string
```

You can use a HTML code snippet to address the XSS vulnerability. In the following code snippet, the user-supplied characters are replaced with their corresponding HTML escape sequences. This is done by taking the input string and replacing certain characters with their HTML escape codes based on the `html_escape_table`. Then, the result is stored in the variable `output_string` which ensures that user-supplied content is displayed correctly in HTML, preventing issues like code injection or unintended rendering of HTML entities. The resulting string `output_string` is safe to display in a web page.

```
output_string = "".join([html_escape_table.get(c, c) for c in
input_string])
```

Key takeaways

As a cloud security analyst, you'll validate the security posture of cloud environments and one way to do this is through vulnerability scans. Knowing how to create and scan web applications to identify and remediate vulnerabilities is crucial in maintaining an organization's security posture and protecting assets from threats.