CRASHTEST **SECURITY**

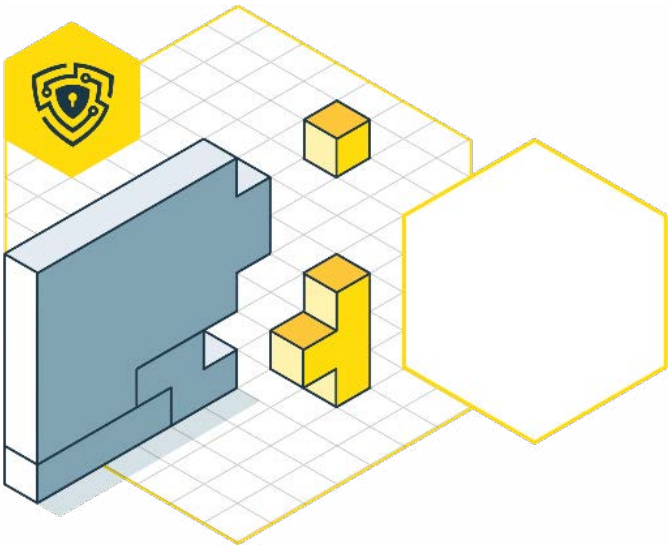
2022

GUIDE FOR PREVENTING XSS ATTACKS

WHAT ARE THE STEPS TO KEEP YOUR WEB APP OR API SAFE FROM SUCH VULNERABILITY

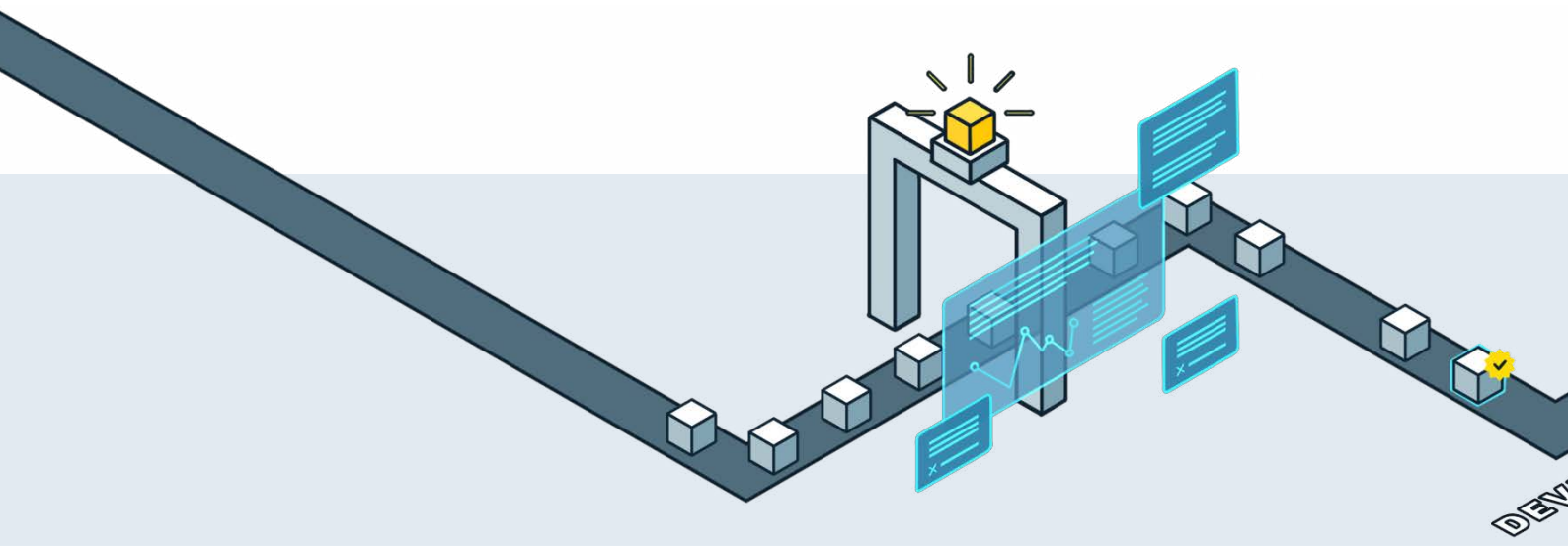
GUIDE FOR THE XSS PREVENTION

Table of Contents



INTRODUCTION TO THIS GUIDE

Injections are a common form of attack where the adversary leverages a security vulnerability in the application to supply it with untrusted inputs. Cross-site scripting (XSS) is one of the most commonly known injection attacks. The attacker manipulates the web application to return malicious scripts executed by the user's browser. In this guide, we discuss cross-site scripting attacks and how to prevent them in modern web applications.



WHAT IS XSS?

In a cross-site request attack, the hacker injects malicious code into the application server through a user input form. If the application does not validate user-supplied input, the malicious content is included in dynamic content forwarded to the browser for execution. While XSS vulnerabilities are present in most web programming frameworks, attacks mainly target **JavaScript** since it integrates with all modern browsers and is used by most dynamic websites.

XSS is a **client-side attack** since the malicious input is delivered to and executed by the user's browser. By compromising the user's interaction with the webserver, an attacker can perform a wide range of vicious tasks, including:

- **Stealing of cookies, session tokens, and other session information**
- **Installing malicious software on the user's machine**
- **Redirecting the victim to the attacker's malicious pages**
- **Theft of personal data/user identity theft**

These vulnerabilities typically occur on application elements that accept unsanitized user input, included in the web server's dynamic response.

TYPES OF XSS

Attackers can leverage the XSS application vulnerability to compromise an application using various tools and techniques. The most common forms of XSS attacks are:

STORED/PERSISTENT XSS

In this type of attack, the hacker injects the malicious script into the application so that it is stored within the webserver. The Stored XSS vulnerability is more common and damaging since the script is only injected once and is included in all consequent HTTP responses. This vulnerability has a broader reach since it affects every user who logs in to the application. Attackers prefer to orchestrate this form of attack since they are self-contained and do not require the crafting of an external means to furnish client-side scripts.

REFLECTED XSS

In a Reflected XSS attack, the webserver immediately returns the user-supplied script within its HTTP response. This attack is not persistent because the webserver does not permanently store the vulnerable code contained within the input. The server reflects the external scripts to the client after processing the user's request. Reflected XSS, also known as **Type 2 XSS attacks**, are more common in forums where the hacker tricks an unsuspecting user into sending malicious code to the webserver using phishing attacks.

DOM-BASED XSS

In a DOM-based XSS attack, the malicious payload is executed when the attackers modify a website's Document Object Model (DOM) in the user's browser. While the HTTP response remains unaltered, the malicious modifications within the victim's browser make the client-side code run unexpected scripts. The attacker inputs vicious code within a user input form that supports dynamic code execution. Some most popular sources for DOM-based vulnerabilities are the **document's URL, location**, and **referrer**.

WHAT IS THE SEVERITY LEVEL OF AN ATTACK?

While it has been exploited and researched since the 21st century, the **Cross-site Scripting** vulnerability is still a choice attack vector due to its simplicity and effectiveness. The **Cross-site Scripting vulnerability** was positioned at **number#7** in the 2017 OWASP top 10 while it was consolidated under the **Injection vulnerability** and was ranked **number#3 in 2021**.

A hacker can deface a web application, redirect users to malicious websites, or steal their session cookie with a successful attack. Additionally, by obtaining user credentials of users with elevated privileges, hackers can achieve complete control of the web application.

HOW TO IDENTIFY XSS VULNERABILITIES WITH CRASHTEST SECURITY?

Crashtest's Security's XSS vulnerability scanner offers a quick and reliable approach to testing for existing XSS security gaps within websites. The scanner combines **DAST** and **SAST** techniques to help developers and security professionals examine each data entry and exit point. Crashtest Security assumes entry points as those interfaces that accept user-controlled data. In contrast, exit points are components of the application where this data might be included in the server's response.

The scanner automatically submits arbitrary values at entry points, analyzing whether the data has been altered at the exit point. Once links between entry and exit points are mapped, the platform tests these links for XSS vulnerabilities using predefined security rules.

Crashtest Security Suite automates XSS testing while swiftly integrating results into your tech stack. The scanner outputs comprehensive CSV, XML, and PDF reports that can be shared across teams for enterprise-wide awareness about the state and impact of XSS vulnerabilities.

XSS PREVENTION TECHNIQUES

Some ways to prevent XSS attacks in modern web applications include:

PERFORM PROPER INPUT VALIDATION

Developers should ensure that user-controlled input interfaces don't accept code that may result in the execution of malicious scripts in user browsers. Each input form should include an allowlist that restricts user input to admissible values. In areas where custom user content is required, the developers should avoid HTML inputs. The developers should follow these and other additional measures to ensure the web server does not accept data from untrusted sources.

AVOID UNTRUSTED URLS

Attackers can leverage unsafe URLs that execute JavaScript code to send malicious payloads within the protocol JavaScript: in DOM URL locations. Web development teams should ensure their URL parsers only accept trusted transfer schemes like HTTPS.

IMPLEMENT A CONTENT SECURITY POLICY

The Content Security Policy (CSP) is a client-side prevention technique that helps detect and prevent injection attacks like XSS. Using a CSP, developers define the domains that the browser can be viewed as trusted sources by when executing the server's response. This enables the browser to ignore scripts in source files originating from domains not listed in the CSP, thereby mitigating an XSS vulnerability.

ENFORCE PROPER OUTPUT ENCODING

Output encoding is a common practice of converting untrusted dynamic user input into secure formats where the browser displays them as data without executing them as code. Each programming language has an entity encoding scheme that should be implemented as part of secure coding best practices by web developers to ensure proper escaping.

ADOPTING BEST PRACTICES FOR XSS PREVENTION

Some framework-specific XSS prevention best practices include:

PHP

n a PHP application, HTML special characters function [***htmlspecialchars()***] helps convert dynamic special characters into HTML entities. This function is recommended to be used for escaping special characters to ensure they are not executed as coded input by the browser.

PHP also includes multiple libraries that can be used to prevent XSS attacks. These include **HTML purifiers, PHP Anti-XSS**, and **htmLawed**.

JAVASCRIPT

There are multiple ways to prevent XSS vulnerabilities in Javascript event handler attributes and blocks. Developers must configure the application to accept inline scripts stored in quoted data values. With the above configuration, any scripts outside the quoted context are treated as unsafe inputs that should not be executed in-browser.

The syntax for safe Javascript would be similar to:

<script>alerts('...ENCODED UNTRUSTED DATA…')</script>

The data can also be part of a quoted expression, as shown:

<script>x='...ENCODED TRUSTED DATA…'</script>

Javascript can also be encoded inside a quoted event handler, as shown:

***<div onmouseover="x='...ENCODE UNTRUSTED DATA BEFORE PUTTING HERE...'" ***

CRASHTEST SECURITY'S VULNERABILITY SCANNER

The Crashtest Security Suite includes several tools to help organizations detect and mitigate cross-site scripting attack vectors. The XSS scanner automatically examines all HTML input and Javascript exit points to enable a quick and safe application security assessment. Crashtest Security's vulnerability detection tool works across multiple layers of the tech stack to help QA and security teams analyze the exact XSS risks to which they are exposed.

To know more about how Crashtest Security vulnerability scanner can help you avoid the risk of being hacked through XSS security gaps, try a **free, 2 weeks** trial here.

Start 2-Week Trial for Free

CRASHTEST **SECURITY**

WWW.CRASHTEST-SECURITY.COM