Web11: HTTP Cookies and some security issues

In this article, TipsMake.com learns about cookies and related security issues.

1. Introduction

A cookie, or HTTP cookie, web cookie, browser cookie is a small piece of data sent from the website and stored in the user's browser when they browse this website. Each time the user loads the website, the browser automatically sends a cookie to the web server to notify the website of the user's previous actions.

Cookies are designed to be a trusted mechanism, helping a website to remember status information (such as items in a shopping cart) or to store user activities (including clicks). a link, login, pages visited during the week, month or year, etc.).

Although cookies cannot carry viruses, nor can they install malware on your computer, the use of tracking cookies and especially third-party cookies is seen as a way to identify malicious users. personal information from the user's browsing history. Cookies can store passwords or content that users enter into html forms, such as credit card numbers or personal addresses.

When a user visits the website for the first time, a cookie is sent from the web server to the browser and stored in his computer. Then, when he returns to the website, the website will recognize him because of the information stored in the cookie.

Authentication cookies are a common method used to determine if a user is logged into a website. Without such a mechanism, it would be difficult for the website to know when to send personal information to the user, and force the user to perform multiple logins. Authentication cookies are very convenient, but create opportunities for hackers to read sensitive data in cookies, thereby performing actions that harm users.

2. HTTP Cookie Overview

2.1. History

The term cookie is derived from the term magic cookie, which is an immutable piece of data that is sent and received by a computer program. A programmer named Lou Montuli came up with the idea of ??using magic cookies in web communications in 1994. At that time, he was working for Netscape Communications, on an e-commerce application development project for MCI. Cookies have been used as a solution in building virtual shopping carts, helping MCI's servers not have to save the state of transactions, but instead, they are stored at the user's machine.

Together with John Giannandrea, Montuli wrote the first specification for Netscape cookies. Version 0.0beta of Mosaic Netscape, released on October 13, 1994, supported cookies. The first use of cookies (outside the lab)

checks whether the customer visiting the Netscape website has visited before or not. Montuli patented cookies in 1995. And Internet Explorer version 2 (released October 1995) has integrated use of cookies.

The introduction of cookies was not widespread during this time. In particular, cookies are accepted by default, the user is not notified of its presence. The public only became aware of the cookie after an article about it in the Financial Times was published on February 12, 1996. Since then, cookies have received a lot of attention, especially the private information contained in them. Cookies were discussed in two US Federal Trade Commission hearings in 1996 and 1997.

The first discussion of a formal specification for cookies began in April 1995. A special group within the Internet Technical Task Force (IETF) was formed to carry out this work. Finally, the specification was published by the group in February 1997. This specification defines third-party cookies as either deprecated for all, or at least not allowed. allowed by default.
In April 2011, the standard specification of cookies used in practice was published in RFC document 6265.

2.2. Terms

Session cookies

Session cookies, also called in-memory cookies or transient cookies, only exist in temporary memory when the user navigates to the website. If an expiration date or validity interval is not set at cookie creation, a session cookie will be set. Normally web browsers will automatically delete session cookies when the user closes the browser.

Persistent cookies

Persistent cookies prolong the user's session. If a persistent cookie is set to Max-Age for 1 year, within that period, the initial value set in the cookie will be sent to the server each time the user visits the website. It can be used to record important pieces of information, such as how the user first visited the website. For this reason, persistent cookies are also called tracking cookies.

Secure cookies

Secure cookie is a security attribute that is enabled when using HTTPS, which ensures that the cookie is always encrypted when it is transferred from the client to the server, preventing it from being eavesdropped. In addition, all cookies are subject to the browser's same-origin policy.

HttpOnly cookies

The HttpOnly attribute of the cookie is supported by most browsers. An HttpOnly session cookie will only be used during an HTTP (or HTTPS) request, thus limiting access by non-HTTP APIs such as Javascript. This restriction mitigates but does not eliminate cookie theft through Cross-site scripting (XSS) vulnerabilities.

Third-party cookies

A first-party cookie is a cookie belonging to the same domain (or of subdomains within the same domain) displayed in the browser's address bar. Third-party cookies (Third-party cookies) are cookies belonging to different domains that are displayed in the browser's address bar. Websites may contain content from third-party

domains (such as banner ads), from which the user's browsing history may be tracked. The privacy settings of most browsers block third-party tracking cookies.

For example, suppose a user visits the website example1.com. This website contains an ad from ad.foxytracking.com which, when downloaded, will also store the ad.foxytracking.com cookie. He then visits another website (example2.com), which also contains an ad from ad.foxytracking.com, and it also sets a cookie belonging to ad.foxytracking.com. Finally, both of these cookies will be sent to the advertiser when they load an ad or visit their website. Advertisers may use this cookie to build a user's browsing history on all websites that contain their ads.

In 2014, there were a few websites that had set cookies that could be read by more than 100 third-party domains. On average, a website will be set about 10 cookies, with the maximum number of cookies being over 800.

Supercookie

Supercookies are cookies that originate from a top-level domain such as .com, or a Public Suffix, such as .co.uk. It is important that the supercookie is blocked by the browser due to some security issues. If unblocked, an attacker controlling a malicious website could set up a supercookie to impersonate user requests, sending requests to other websites that share the same Top-level domain or Public Suffix. For example, a supercookie originating from the domain .com could compromise requests to example.com, even if the cookie did not originate from example.com. It can be used to spoof logins or change user information.

Zombie cookies

Zombie cookies are cookies that are automatically regenerated after the user deletes them. This is done by a script that stores the content of the cookie in another location, such as the Flash content pool, the HTML5 bucket, or some other client-side mechanism.

2.3. Cookie structure

A cookie with a size of 4KB, consisting of 7 main components:

1. Name
2. Value
3. Expires (expiry date)
4. Path (the path to where the cookie is valid, '/' means the cookie is valid anywhere)
5. Domain
6. Secure
7. HttpOnly

The first two components (name and value) are required.

2.4. Use

Session Management

Cookies can be used to maintain data related to the user during various visits to the website. Cookies are the solution to creating a shopping cart, a virtual shopping cart that helps users save the items they choose while

browsing through the products.

Today's shopping cart application often stores the list of items in the cart in a server-side database instead of storing it in a client-side cookie. The web server will normally send a cookie containing the session identifier - session ID (which is unique). The web browser will return this session ID with each user's purchase request.

Allowing users to log into a website is another use of cookies. Normally, on first login, the web server sends the client a cookie containing the session ID. The user will submit their information and the web application will authenticate the session, then allow the user to use its services.

Cookies provide a fast and convenient client/server interaction mechanism. One of the advantages of a cookie is that it stores user information in a file located on the user's machine. This significantly reduces the storage space and processing time of the server.

Personalized

Cookies can be used to remember a user's personal information when they visit the website, to display more relevant content to that user each time he revisits the website.

A prominent example is amazon.com's book recommendation feature. When the user clicks on a book, amazon will give suggestions on the next books that the user should see. These recommendations are based on which books the user has previously browsed and which books they have chosen to purchase.

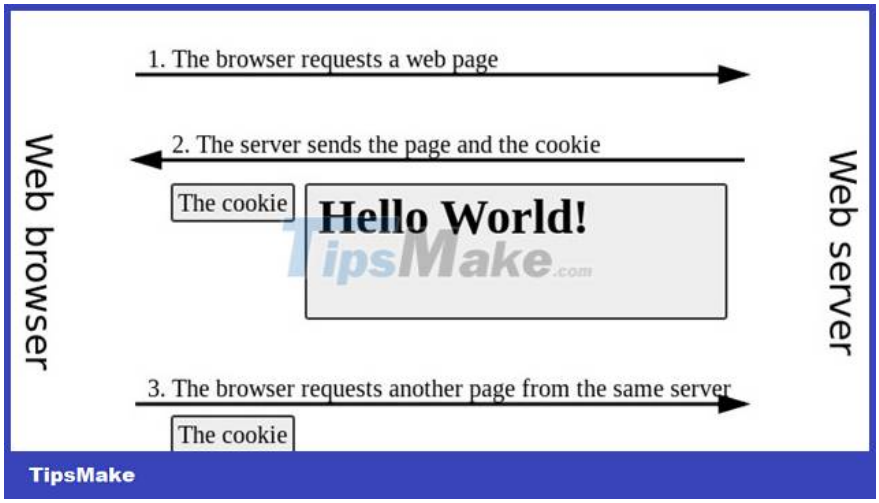
Another example is the search suggestion feature of Google Search. When you log in with your personal account and do a search, Google will make 'just for you' suggestions, and the results you want are always at the top of the list.

Follow

Tracking cookies can be used to track a user's browsing history. This can also be done by using the IP address of the computer sending the request to the site or by relying on the Referrer field of the HTTP request header, but cookies give more accuracy. This is done as follows:

1. If the user visits a website, but the request does not contain a cookie, the server assumes that this is the user's first visit to the site; The server will generate a random string and send it as a cookie back to the browser along with the requested web page.
2. From this point on, the cookie will be automatically sent by the browser to the server every time a new web page in this website is requested, the server will return the requested web page as usual, but the URL of that website and when Access will be saved in a log file. By analyzing the information stored in this log file, browsing history, frequently visited websites, browsing habits of people will be revealed.

2.5. Implementation



Cookies are data sent from the server to the browser. The browser will then send it back to the server without changing the contents, each time the user sends a request to the website. Cookies can also be set by a scripting language, such as Javascript.

Each web browser can store at least 300 cookies in a 4KB file, and at least 20 cookies per server or domain.

Set up cookies

Web server and browser communicate with each other via HTTP (HyperText Transfer Protocol). For example, to access the page http://www.example.org/index.html, the browser connects to the server by sending an HTTP Request of the form:

The server responds by sending the browser a simple text packet, called the HTTP Response. This packet may contain a single line containing the content of the cookie:

Set-Cookie is a field that instructs the browser to store a cookie and send it back to the server in the future whenever there is a request to the server (if the cookie is still expired). For example, the browser sends a request to http://www.example.org/spec.html by sending an HTTP Request of the form:

This is a request to another page on the same server. In this case, the server understands that this request is related to the previous request, and it responds by sending the browser the requested web page, possibly adding other cookie values.

The value of the cookie can be modified by the server by sending the browser the Set-Cookie: name=value field in the HTTP Response. The browser will then replace the old cookie value with this new value.

The value of a cookie can include any printable ASCII character except ',', ';' and whitespace. The name of the cookie must also not contain the '=' character, as that is the separator between the name and the value.
The term cookie crumb is sometimes used to refer to a cookie's name-value pair.

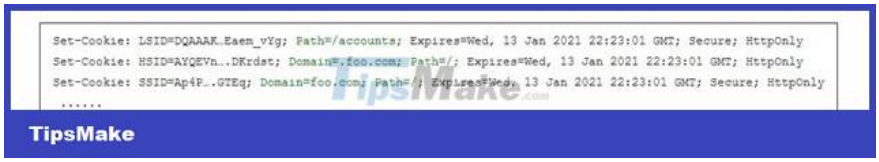
Cookies can also be set by Javascript or a similar scripting language. In Javascript, the document.cookie object is used to set cookies. The HttpOnly attribute is responsible for preventing bad scripts from reading the cookie content.

Attributes of cookies

Besides name-value pairs, the server can also set a number of other cookie properties: Domain, Path, Exirse, Max-Age, Secure, and HttpOnly. The browser will not send these attributes to the server, it only sends name-value pairs. These attributes are used by the browser to determine when to delete cookies, block cookies, or send cookies to the server.

Domain and path

Domain and path define the scope of the cookie. They allow the browser to determine when to send cookies to the server. If not specified, they assume that the domain and path of the object have been requested. However, there is a difference between a cookie set for foo.com without the domain attribute, and a cookie set with the domain attribute foo.com. In the first case, the cookie will only be sent when there is a request to foo.com. In the latter case, the cookie will be sent to all subdomains of foo.com. The following is an example of a Set-Cookie directive from a website after a user logs in, from a request to docs.foo.com:



The first cookie LSID has no domain attribute and has a path of /accounts. The browser will only send cookies when the requested page is contained in docs.foo.com/accounts. The other two cookies, the HSID and the SSID, are sent back to the server if there is a request to any of the subdomains of foo.com.
Cookies can also be set only for the top domain and its sub domain. Setting cookies on www.foo.com from www.bar.com will not be allowed for security reasons.

Expires and Max-Age

The Expires attribute tells the browser when to delete the cookie. Dates in Expires look like this: 'Wdy, DD Mon YYYY HH:MM:SS GMT'. Max-Age is also used to notify the expiration date of cookies. Let's consider the following example:

The first cookie is set to expire at 15-Jan-2013, it will be used by the browser when it expires. The second cookie made_write_conn does not expire, and is used as a session cookie, which is deleted when the browser is closed. The third cookie, reg_fb_gate, has expire time in the past, it will be deleted immediately.

Secure and HttpOnly

The Secure and HttpOnly properties are null; instead, their presence indicates that the Secure and HttpOnly sizes apply.

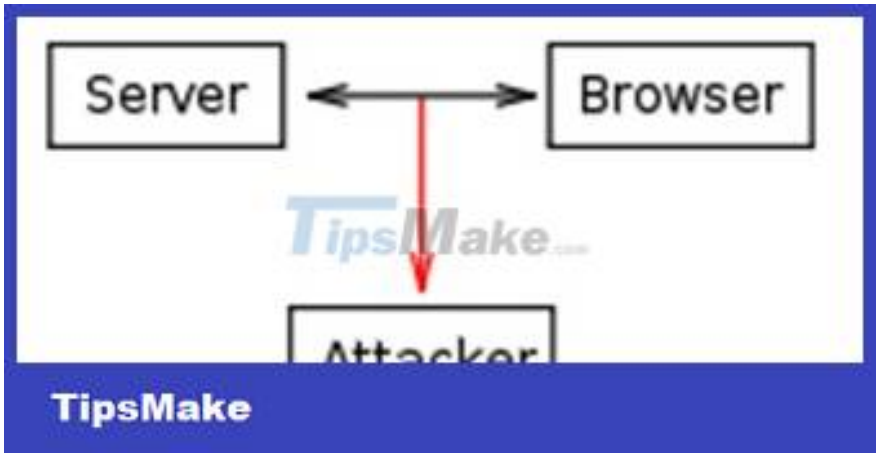
The Secure attribute holds the cookie transmission during an encrypted connection. If the web server sets a cookie with the secure attribute from a non-secure connection, the cookie can still be intercepted by a man-in-the-middle attacker (man-in-the-middle attack).

The HttpOnly attribute instructs the browser not to expose the cookie over a connection other than HTTP (or HTTPS), such as Javascript, and thus is difficult to obtain by exploiting the Cross-Site Scripting (XSS) vulnerability. .

3. Some security issues when using cookies

If a website uses session IDs to identify a user's session, an attacker could possibly steal cookies to impersonate the user. Here are some common cookie theft scenarios:

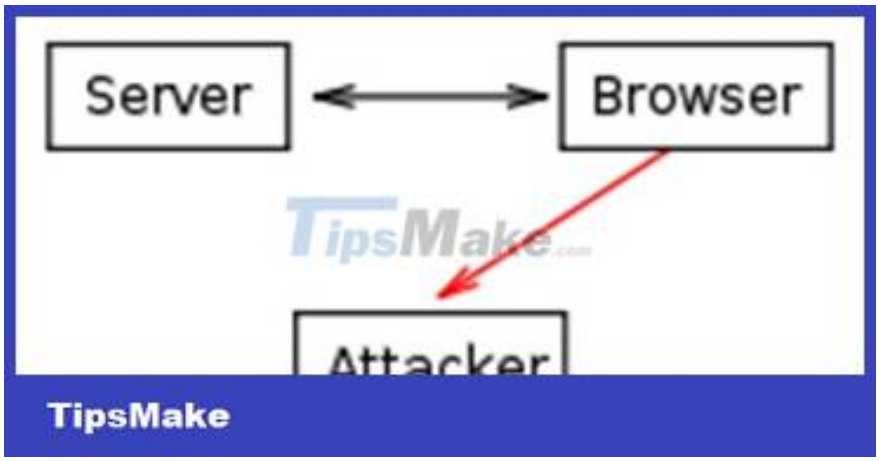
eavesdropping



Traffic in a network can be intercepted and read by a third person (not the receiver and the sender). This traffic includes cookies. If the transmission is not encrypted, an attacker can read the sensitive information contained in the cookie. And taking advantage of this information, attackers will impersonate users to perform malicious actions, such as banking transactions.

This problem can be solved by using a secure protocol between the user's computer and the server - the HTTPS protocol. The server can use the Secure flag when setting cookies. The cookie will then be sent only through an encrypted channel, such as an SSL connection.

Cross-site scripting



Scripting languages, such as Javascript, are allowed to read cookie values ??and send them to arbitrary servers.

Assuming a website has a Cross-site scripting error, hackers can insert malicious code, such as the following:

When the victim clicks on the above link, the browser will execute the script in the onclick: attribute, sending the victim's cookie to the attacker.com server.

Cross-site scripting is one of the most common vulnerabilities of websites (ranked third in the top 10 most common vulnerabilities in 2013 - according to OWASP).

The risk of this vulnerability can be reduced by setting the HttpOnly flag for cookies. Then the cookie will not be accessed by scripting languages.

Cross-site request forgery

CSRF (Cross-site request forgery) or one-click attack, is a method of exploiting website vulnerabilities in which unauthorized commands are executed by victims - users who are authorized by the website without their knowledge. .

CSRF will trick the victim's browser into sending http requests to web applications. In case the victim's session has not expired, the above requests will be made with the victim's authentication rights.

For example, victim Bob is transacting at some bank X's website. Fred is the attacker, this guy knows how the website of bank X works when he wants to transfer money from account A to account B as follows:

http://bank.example.com/withdraw?account=accountA&amount=100&for=accountB

He will send Bob a malicious route.

If bank X stores user credentials in cookies, and Bob clicks on the above link, Bob's money will be transferred to Fred.

To limit this risk, the Expires parameter can be set for cookies.

4. Conclusion

Cookies are used in most web applications today. It also contains potential risks, which have a significant impact on users. Therefore, on the side of developers, we need to understand cookies well and know how to set the necessary parameters so that the application can be more secure against hacker attacks.