**What is Session Hijacking?**

The session hijacking is a type of web attack. It works based on the principle of computer sessions. The attack takes advantage of the active sessions. To know this in detail, we need to know what is a session. Let’s see what is a session and how the session works first.

**What is a session?**

The session refers to certain time period that communication of two computer systems or two parts of a single system takes place. When one logins to a password protected system, the session is used. The session will be valid up to the end of the communication. In some cases, such as in the above described case, the session is user-initiated. There is technology initiated sessions also. Various email clients use the sessions and these are examples for the sessions initiated by the technology. However, many of the active sessions will be hidden from the users. They will not know when a session starts and ends. The session is an important factor in the Internet communications.

Coming to the session hijacking, as we’ve seen earlier, the attacker uses the active session for implementing the attack. For most Internet communications, authentication will be needed. Authentication can be done in different methods. The most used method is the user is asked to enter a predefined username and password by the website. When the user enters this credentials, the system will check the same with the stored details. If the entered details match with the stored details, the system grants access to the particular user to the particular database or part of the website.

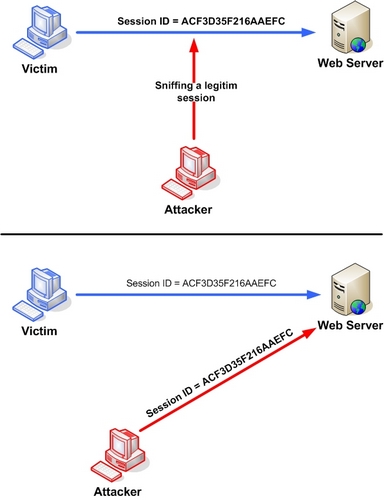
This type of authentication is done at the start of the communication and when the authentication completed successfully, then the session starts and remains active up to the end of the communication. The session hijacking attack takes place in such a fashion that when a session is active the attacker intrudes at the same time and takes advantage of the active session. This intrusion may or may not be detectable. Every session will be having a session id. This session id will be often stored in cookies or URLs. This attack is also called “Cookie Hijacking”. We can recognize a session hijacking attack by the behavior of the website that uses the current session. If the website does not respond in the expected way to the user inputs or if it completely stops working for unknown reason, it can be the result of a session hijacking attack.

**How Does Session Hijacking Works?**

As we know, the http communication uses many TCP connections and so that the server needs a method to recognize every user’s connections. The most used method is the authentication process and then the server sends a token to the client browser. This token is composed of a set of variable width and it could be used in different ways, like in the URL, in the header of http requisition as a cookie, in other part of the header of the http request or in the body of the http requisition. The attack compromises the session token by stealing or predicting a valid session token to gain unauthorized access to the web server. This compromising of session token can occurr in different ways. We are now going to see the two ways as session sniffing and cross-site script attack.

The Session Hijacking attack compromises the session token by stealing or predicting a valid session token to gain unauthorized access to the Web Server.

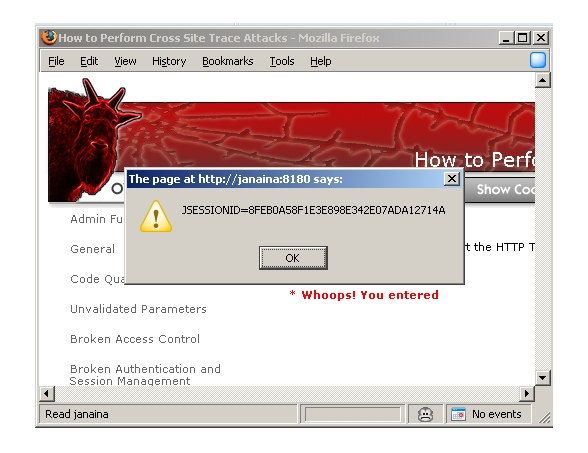
The session token could be compromised in different ways; the most common are:

* Predictable session token;
* Session Sniffing;
* Client-side attacks (XSS, malicious JavaScript Codes, Trojans, etc);
* [Man-in-the-middle attack](https://www.owasp.org/index.php/Man-in-the-middle_attack)
* [Man-in-the-browser attack](https://www.owasp.org/index.php/Man-in-the-browser_attack)
* **Example 1**
* **Session Sniffing**
* As we’ve seen earlier, there is a string called tokens. This is the session id of a valid session. The first step by the attacker is getting this session id. The attacker uses a sniffer to get the session id. When the session id is captured, the attacker uses this session id to gain unauthorized access to the web server.
* In the example, as we can see, first the attacker uses a sniffer to capture a valid token session called “Session ID”, then he uses the valid token session to gain unauthorized access to the Web Server.
* [](https://www.owasp.org/index.php/File:Session_Hijacking_3.JPG)
* Figure 2. Manipulating the token session executing the session hijacking attack.
* **Example 2**

**Cross-site script attack**

The cross-site script attack is a way to get the session id with the helping of running malicious code or script from the client side. In this attack, the attacker executes malicious scripts, also known as malicious payloads into a legitimate website or web application. By using this attack, the attacker does not target a victim directly, but the attacker could exploit a vulnerability in a website that the victim would visit and use the website to deliver malicious script to the victim’s browser.

* The attacker can compromise the session token by using malicious code or programs running at the client-side. The example shows how the attacker could use an XSS attack to steal the session token. If an attacker sends a crafted link to the victim with the malicious JavaScript, when the victim clicks on the link, the JavaScript will run and complete the instructions made by the attacker. The example in figure 3 uses an XSS attack to show the cookie value of the current session; using the same technique it's possible to create a specific JavaScript code that will send the cookie to the attacker.
* <SCRIPT>alert(document.cookie);</SCRIPT>

[](https://www.owasp.org/index.php/File:Code_Injection.JPG)

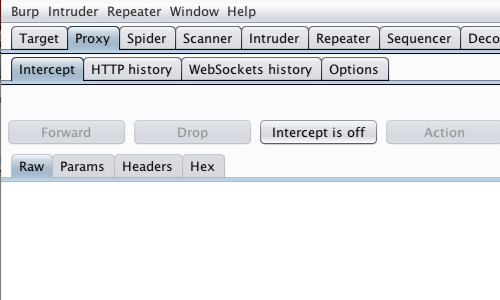
**How to prevent the Session Hijacking?**

As we’ve seen earlier, the method often used to steal session id is by installing a malicious code on the client website and then the cookie is stealing. The best way to prevent session hijacking is enabling the protection from the client side. It is recommended that taking preventive measures for the session hijacking on the client side. The users should have efficient antivirus, anti-malware software, and should keep the software up to date.

There is a technique that uses engines which fingerprints all requests of a session. In addition to tracking the IP address and SSL session id, the engines also track the http headers. Each change in the header adds penalty points to the session and the session gets terminated as soon as the points exceeds a certain limit. This limit can be configured. This is effective because when intrusion occurs, it will have a different http header order.

These are the recommended preventive measures to be taken from both the client and server sides in order to prevent the session hijacking attack

# **Using Burp to Hack Cookies and Manipulate Sessions**



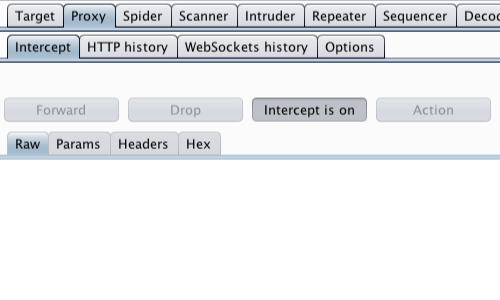
First, ensure that Burp is correctly[configured with your browser](https://support.portswigger.net/customer/portal/articles/1783055-Installing_Configuring%20your%20Browser.html).

With intercept turned off in the [Proxy](https://support.portswigger.net/customer/portal/articles/1783118-Proxy_Getting%20Started.html) "Intercept" tab, visit the login page of the application you are testing in your browser.



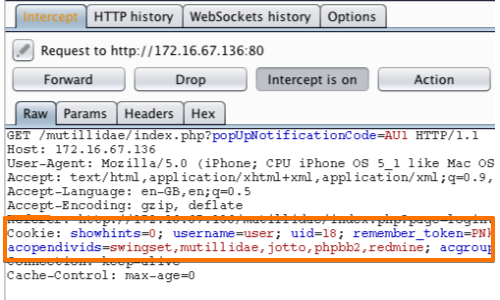
Log in to the application you are testing.

You can log in using the credentials user:user.



Return to Burp.

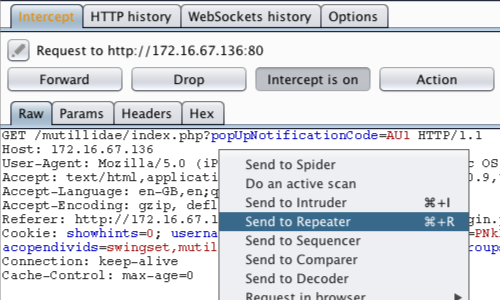
In the [Proxy](https://support.portswigger.net/customer/portal/articles/1783118-Proxy_Getting%20Started.html) "Intercept" tab, ensure "Intercept is on".



Refresh the page in your browser.

The request will be captured by Burp, it can be viewed in the [Proxy](https://support.portswigger.net/customer/portal/articles/1783118-Proxy_Getting%20Started.html) "Intercept" tab.

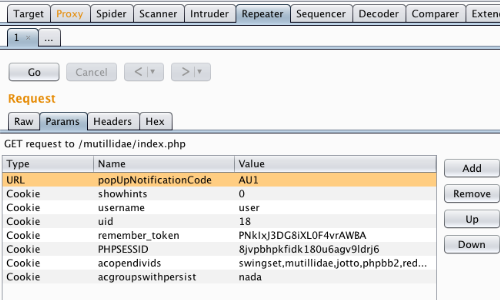
Cookies can be viewed in the cookie header.



We now need to investigate and edit each individual cookie.

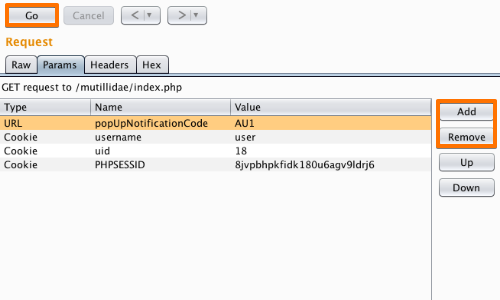
Right click anywhere on the request and click "Send to [Repeater](http://portswigger.net/burp/help/repeater.html)".

**Note:** You can also send requests to Repeater via the context menu in any location where HTTP requests are shown, such as the site map or Proxy history.



Go to the [Repeater](http://portswigger.net/burp/help/repeater.html) tab.

The cookies in the request can be edited easily in the "Params" tab.

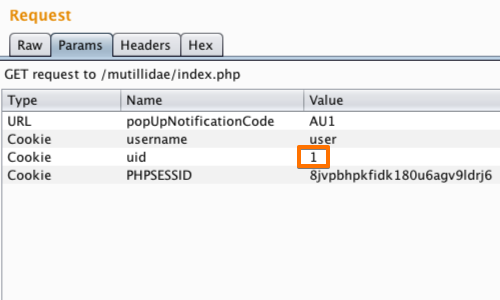


By removing cookies from the request we can ascertain the function of each cookie.

In this example, if the "username", "uid" and "PHPSESSID" cookies are removed, the session is ended and the user is logged out of the application.

We can use the [Repeater](http://portswigger.net/burp/help/repeater.html) to remove cookies and test the response from the server.

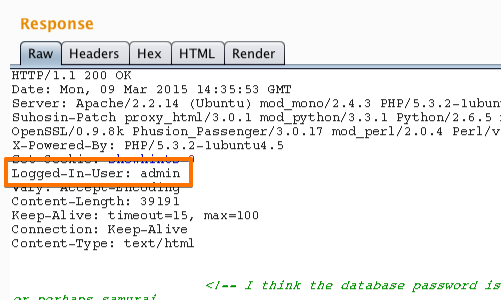
Remove and add cookies using the "Add" and "Remove" buttons and use the "Go" button to forward requests to the server.



Cookies can be edited in the Request "Params" table.

In this example we have altered the value of the "uid" cookie to 1.

Alter the value then click the "Go" button.



The response from the server can be viewed in the "Response" panel in Repeater.

The response shows that by altering the "uid" cookie we have logged in to the application as "admin".

We have used cookies to manipulate the session and access another account with elevated privileges.