

**St. Francis Institute of Technology**  
**(An Autonomous Institution)**  
AICTE Approved | Affiliated to University of Mumbai

A+ Grade by NAAC: CMPN, EXTC, INFT NBA Accredited: ISO 9001:2015 Certified

**Department of Information Technology**

A.Y. 2025-2026  
Class: BE-IT A/B, Semester: VII  
Subject: Secure Application Development Lab

Student Name: Charis Fernandes

Student Roll No: 26

**Experiment – 8: Study of Session Management for Web Applications**

**Aim:** To Study the session management for web applications

**Objectives:** The students will be able understand Session Management for the web applications

**Lab objective mapped:** ITL703.5 To apply Security at Session Layer Management

**Requirements:** Personal Computer, Windows operating system browser, Internet Connection

**Pre-Experiment Theory:**

**Session Management for Web Applications**

Session management is intended to provide a robust and cryptographically secure association between authenticated users and their sessions. Using session management combined with secure HTTP (HTTPS), you can enforce authorization checks and prevent common attacks such as replay attacks, request forging, and man-in-the-middle attacks.

A web server is free to forget about pages it has served in the past, so there is no explicit state. To tie state explicitly held on the server to a session in progress, an explicit session ID is stored somewhere in client-submitted data.

A session is a lasting connection between a web user (the browser) and the Uniface Web Application Server. It starts when the user starts to access a web application and ends when the browser is closed, when the session expiry time is exceeded, or when the user logs out (assuming a logout mechanism is provided). Uniface generates a unique session identifier (session ID) for each session, which can be accessed using the SESSION property of \$webinfo("WEBSERVERCONTEXT").

**Threats**

The session ID is unique and is therefore vulnerable to attacks in which the session ID is stolen and then used to mimic a user and gain further access to the application. The following topics describe some of the techniques used to steal session IDs,

- Session Fixation:

Session fixation is a web security threat in which a user's session ID is set to one known to an attacker. For example, an attacker can send a user an email with a link containing that particular session ID. In this case the attacker just has to wait until the user logs in.

- Session Sidejacking

Session sidejacking is a type of security threat in which an attacker hijacks a session by intercepting and reading network traffic between two parties to steal the session cookie.

Unsecured WIFI hotspots are particularly vulnerable to this type of interception, since anyone sharing the network will generally be able to read most of the network traffic between other nodes and the access point. Alternatively an attacker (with physical access) could simply steal the session key directly.

- Session replay

Using a session ID, an attacker can replay a web session to masquerade as a user and gain access to a web application

- Cross-Site Scripting

Cross-site scripting vulnerabilities enable an attacker to trick the user's browser into running code (such as JavaScript, Java, Flash, ActiveX, or VBScript) that is treated as trustworthy, thereby allowing the attacker to obtain a copy of the session ID or inject malicious code into web pages viewed by other users.

- Eavesdropping and Cookies Containing Sensitive Data

Eavesdropping is a web security threat in which intermediate servers or sniffer programs intercept and read HTTP data packets as they traverse intermediate servers en route to their destination.

Web applications use the Hypertext Transfer Protocol (HTTP) as a communication protocol between servers and clients. Because the HTTP requests and responses are sent in plain text, they can be easily read and analyzed. Moreover, the internet transfers data by traversing many intermediary computers and routers, increasing the number of nodes where data packets can be intercepted and read.

### Session Cookies:

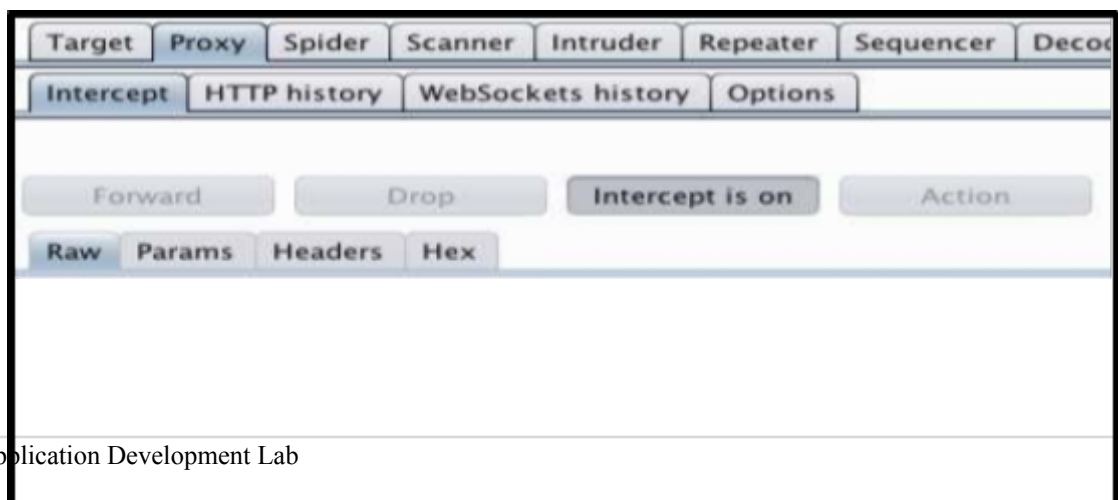
Session cookies track the user's behavior on the website and help websites identify users browsing through web pages of a website. The session ID is a unique, randomly generated number that stores the session cookies.

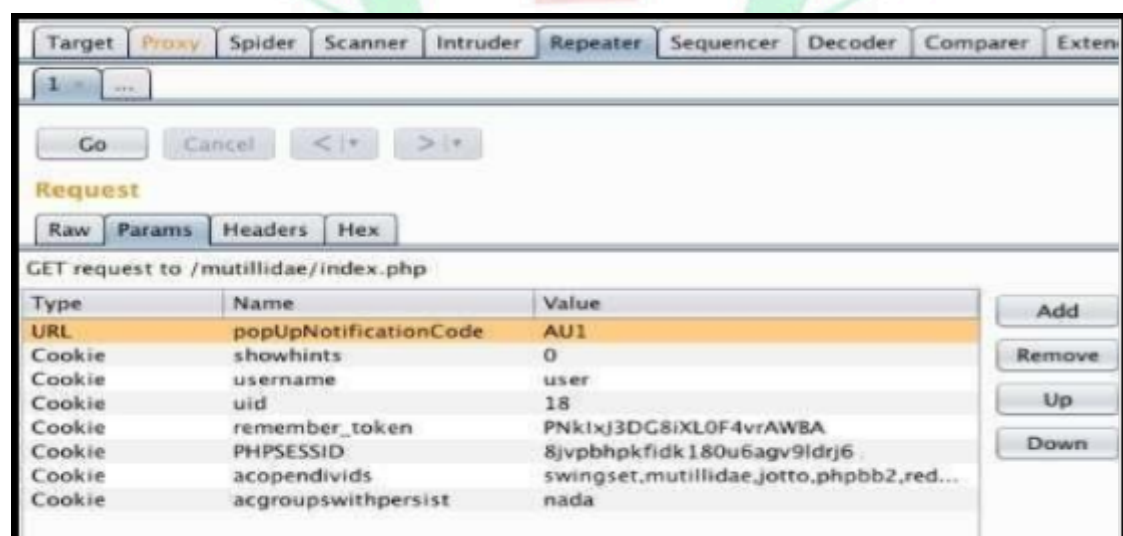
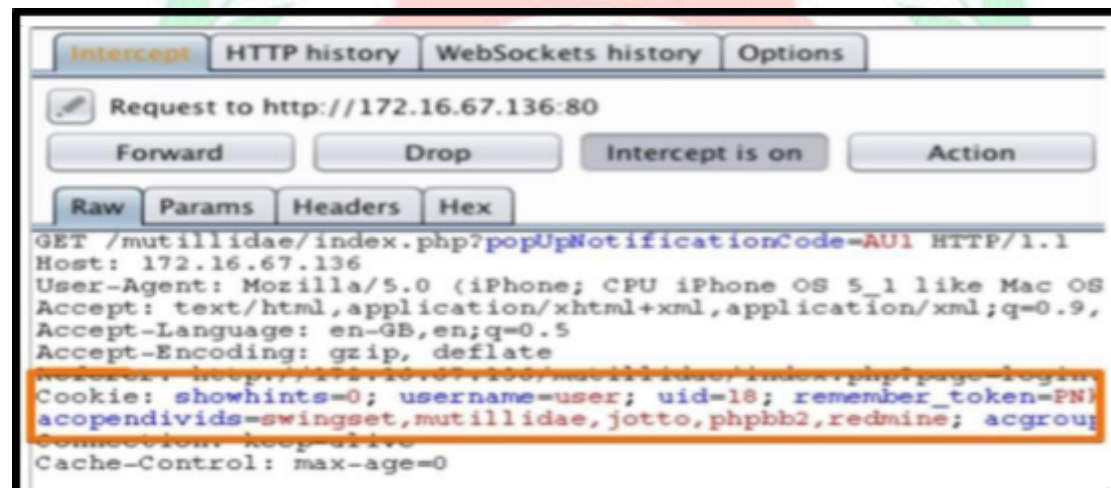
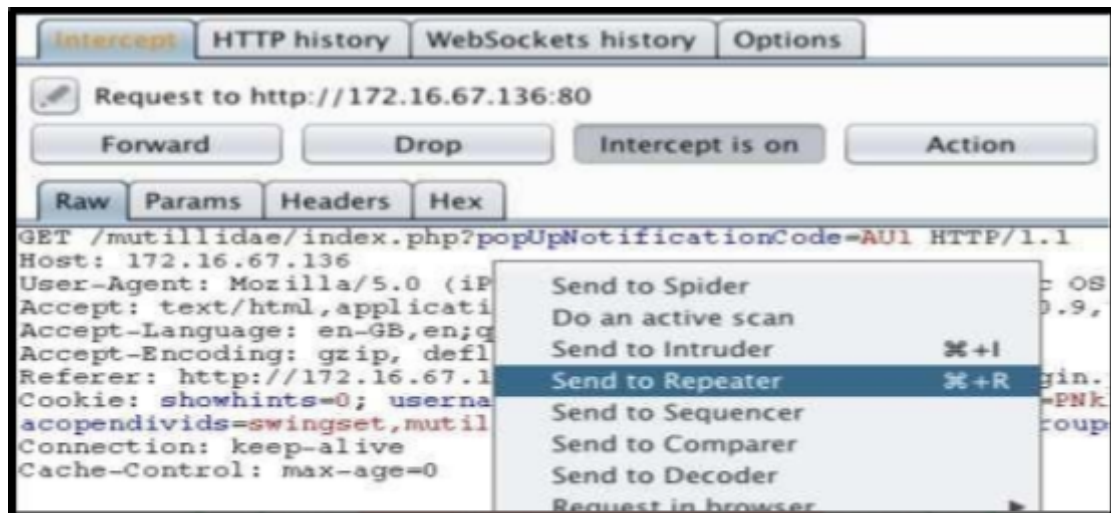
Cookies are vulnerable to man-in-the middle attacks if their secure attribute does not match the session connection (HTTP or HTTPS).

A cookie that contains sensitive data, such as authentication information, should have its secure attribute set to true, and it should only be created and used only within an HTTPS session. You should not rely on either the secure attribute alone or on the SSL connection alone to protect the data.

**Procedure:****Using burp suite to hack cookies and manipulate-sessions**

<https://portswigger.net/support/using-burp-to-hack-cookies-and-manipulate-sessions>

**Screen Shots:**





Go Cancel < >

Request

Raw Params Headers Hex

GET request to /mutillidae/index.php

Type	Name	Value
URL	popUpNotificationCode	AU1
Cookie	username	user
Cookie	uid	18
Cookie	PHPSESSID	8jvpbhpkfidk180u6agv9ldrj6

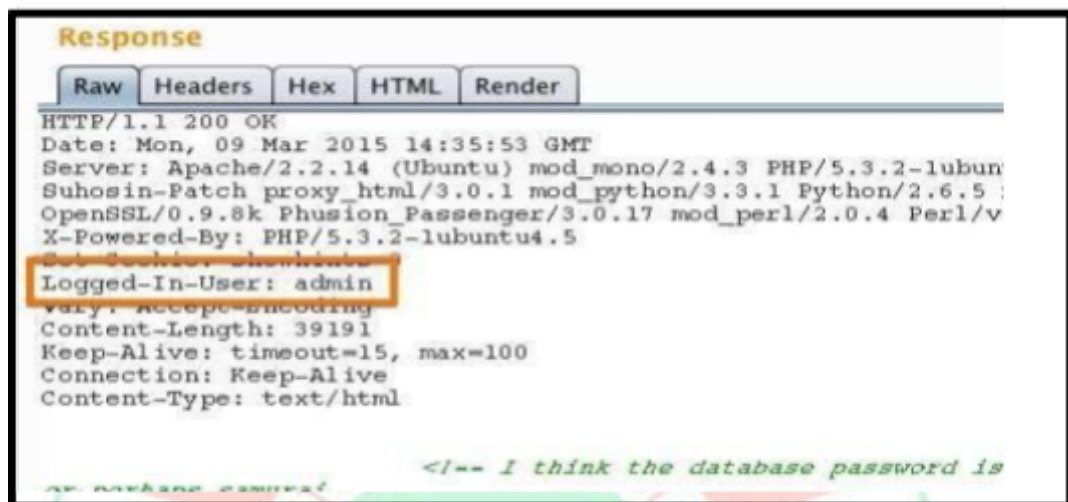
Add Remove Up Down

Request

Raw Params Headers Hex

GET request to /mutillidae/index.php

Type	Name	Value
URL	popUpNotificationCode	AU1
Cookie	username	user
Cookie	uid	1
Cookie	PHPSESSID	8jvpbhpkfidk180u6agv9ldrj6



```
Response
Raw Headers Hex HTML Render
HTTP/1.1 200 OK
Date: Mon, 09 Mar 2015 14:35:53 GMT
Server: Apache/2.2.14 (Ubuntu) mod_mono/2.4.3 PHP/5.3.2-lubun
Suhosin-Patch proxy_html/3.0.1 mod_python/3.3.1 Python/2.6.5
OpenSSL/0.9.8k Phusion Passenger/3.0.17 mod_perl/2.0.4 Perl/v
X-Powered-By: PHP/5.3.2-lubuntu4.5
Logged-In-User: admin
Vary: Accept-Encoding
Content-Length: 39191
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html

<!-- I think the database password is
```

### Post-Experiments Exercise:

Extended Theory: Nil

### Post Experimental Exercise:

#### Questions:

- Discuss the guidelines for secure session management.
- Explain the secure Session Management in HTTP along with its best practices.

#### Conclusion:

- Write what was performed in the experiment.
- Write the significance of the topic studied in the experiment.

#### References:

1. <https://snyk.io/blog/session-management-security/>
2. <https://www.geeksforgeeks.org/session-management-in-http/>
3. <https://www3.rocketsoftware.com/rocketd3/support/documentation/uniface/10/uniface/webapps/webSecurity/sensitiveCookies.html>

