

Web Ethics, Session Management and Session Fixation

Web Development and Security (ZEIT3119)

Week 10

Dr. Reza Rafeh

Revision

- MVC (Model-View-Controller)
 - Separation of Concerns
 - Reusability of Code
 - Testability
 - Flexibility
 - Improved Collaboration

Revision

- REST (Restful) API
 - REST is language independent
 - It is data driven i.e resources
 - It is stateless
 - API calls can be cached
 - Security: it Supports SLL &HTTPS
 - Requires fewer resources
 - Message formats: it Supports Plain text, HTML,JSON,XML.YAML & others
 - It uses HTTP status code, as it has standard Status code, such as 200,201
 - Rest is very light weight
 - Easy to call from JavaScript
 - It calls services via URL path
 - Performance is much better, less CPU intensive
 - For receiving data it uses XML or JSON

Ref: [What is REST API and its advantages? – Quora](#)

Outline

- Web Ethics and Safety
- Session Management
- Passing a parameter in Session Management
- Session Fixation
- Session Hijacking
- Cookies

Web Ethics and Safety

➤ Ethics

- Ethics and moral are terms that refer to the principles and how we decide to behave when interacting with other people
- Ethics are involved in every profession
- Business ethics is about how customers and employees are treated
- Similarly, Web ethics means acceptable behavior for using the internet
- Everyone is obliged to respect the rights and property of others on the internet

Web Ethics and Safety

CyberBullying

- It can be done by someone you know or a stranger
- It takes place over digital devices; cell phones, computers and tablets
- It is sending or sharing negative and harmful content about someone
- It can include sharing personal, private information about someone
- It can occur through SMS, emails, texts, gaming and social media apps

Safety

- Do not share personal information with anyone
- No photographs should be shared or sent to the strangers because they can misuse it.
- Always think and don't just link
- Make the passwords longer and stronger
- Do not use the same password

Web Ethics and Safety

Ethical rules for Computer users

- Do not use computers to harm other users.
- Do not use computers to steal others information.
- Do not access files without the permission of the owner.
- Do not copyright software without the author's permission.
- Always respect copyright laws and policies.
- Respect the privacy of others, just as you expect the same from others.
- Do not use other user's computer resources without their permission.
- Use the Internet ethically.
- Complain about illegal communication and activities, if found, communicate to Internet service Providers and local law enforcement authorities.
- Users are responsible for safeguarding their User ID and Passwords
- They should not write them on paper or anywhere else for remembrance.
- Users should not intentionally use the computers to retrieve or modify the information of others, which may include password information, files, etc..

Web Ethics and Safety

Ethical rules for Internet users

- *You should not use the Internet to harm other people.*
- *You should not interfere with other people's Internet work.*
- *You should not snoop around in other people's Internet files.*
- *You should not use the Internet to steal any data.*
- *You should not use the Internet to bear false witness.*
- *You should not copy or use proprietary software for which you have not paid (without permission).*
- *You should not use other people's Internet resources without authorization or proper compensation.*
- *You should think about the social consequences of the program you are writing or the system you are designing.*
- *You should always use the Internet in ways that ensure consideration and respect for your fellow humans.*

Web Ethics and Safety

Copyright

- It is a law that gives ownership to the content created by you.
- Copyright law grants several rights that an owner can have
 - Right to reproduce the work
 - Right to delete/edit the work
 - Right to make the work publicly
 - Right to distribute the copies
 - Right to perform the work

Session Management

- Session refers to the time spend by the user on the website
- Session management is the set of rules that governs between the user and the web based application
- Session management refers to the process of securely handling multiple requests to a web-based application or service from a single user or entity.
- Managing session means to maintain a stateful website
- Browsers and websites uses HTTP to communicate and HTTP is a stateless protocol where each command run independently without knowing the previous command
- HTTP is not responsible to maintain the parameters communicated between browsers and server
- HTTP does not have the procedure to select which parameters to pass via the Request packet

Session Management

- It is the responsibility of the application layer to define which set of parameters to be passed on every Request.
- To implement or introduce the concept of session, it is important to implement session management capabilities that can link both authentication and access control modules. These modules are available in web applications
- Web developers must define the set of parameters to pass as users browse a website, so that the displayed pages dynamically respond to the user's previous interaction with the website.
- There are two types of session management: cookie-based and URL re-writing. They can be used independently or together.
- A web administrator uses session management to track the frequency of visits to a website and movement within the site.

Why Sessions are necessary?

- A session is started when a user authenticates their identity using a password or another authentication protocol.
- A session allows you to share information across different pages of a single site that helps maintain state
- Once the user is authenticated, server knows that all the requests originate from the same user and server displays the user specific information
- Session management involves the sharing of secrets with authenticated users, secure cryptographic network communications are essential to maintain session management security.



Session Management and PHP

- The default setup of a PHP server is to use both Cookie propagation and URL propagation
- PHP checks whether the user has cookies enabled
- If the cookies are on, PHP uses Cookie propagation
- If cookies are off, PHP uses URL propagation

Session Management and PHP

➤ Cookie Propagation

- A cookie is stored on the users PC containing the session id.
- It is read in whenever `session_start()`; is called to initialize the session.
- Default behaviour is a cookie that expires when the browser is closed. Cookie properties can be modified with `session_set_cookie_params` if required.

```
session_set_cookie_params(30 * 60, "/", "test.com");
```

➤ URL Propagation

- The session id is propagated in the URL



```
(...some_folder/index.php?sid=26fe536a534d3c7cde4297abb45e275a)
```

- PHP provides a global constant to append the session id to any internal links, `SID`.
- e.g.

```
<a href="nextpage.php?<?=SID?>">Next page</a>
```

Session Management

Passing a parameter

1. Using a URL

- Typing the parameter in the URL browser directly
- Useful when we have to bookmark or forward the page with their parameters

2. Through Form

- Submitting the form through user input. The parameters are passed to web servers once submit button is clicked
- It is applicable where the display of the page depends on the users input, such as a search-result page or enrolment forms.

Session Management

Passing a parameter

3. Through Hidden fields

- Another way of passing the parameters are through hidden fields. This is used to maintain sessions.
- They are visible on the HTML page but not on the browser.

Index.html

```
<!DOCTYPE html>
<html>
  <body>
    <form action="second.html">
      <input type="hidden" name="privilege"
value="admin">
      <button type="submit">Next</button>
    </form>
  </body>
</html>
```

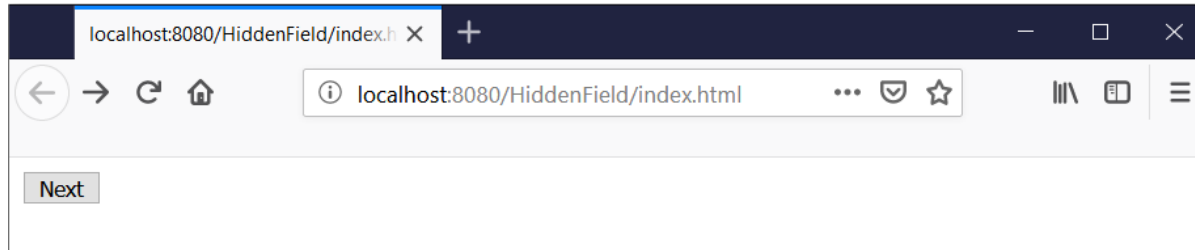
Second.html

```
<html>
  <body>
    Hidden parameters are passed, but not shown.
  </body>
</html>
```


Session Management

Passing a parameter

➤ Output

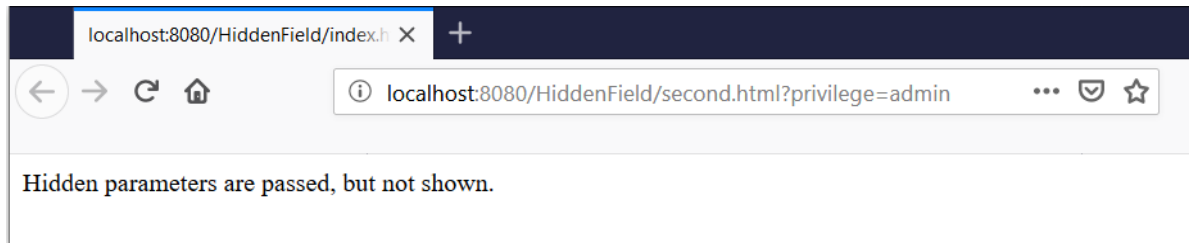


- The displayed page only shows the button, without the name-value pairs of the hidden input field. However, when you examine the HTML page that the browser receives, the hidden field is not hidden, i.e. it is communicated from the server to the browser. Witness this through the browser developer's tool, by pressing Ctrl + Shift + C on the browser.

Session Management

Passing a parameter

- When the “Next” button is pressed, the second.html page is displayed. Notice that the hidden parameters are passed.



- The above example demonstrates the nature of hidden fields. Although they are named “hidden”, they are not for security purposes.
- Another nature of hidden fields is that they are written as one input type for passing one parameter.
- Passing multiple parameters through hidden fields to maintain session is not scalable. Hidden fields are suitable to pass one (or a handful) message to another page.

Session Management and PHP

- To enable a session in PHP you have to use the function `session_start()`
- Once the session is started we can work with the `$_SESSION` variable like any other variable but there is one exception:
- Anything we leave in this variable at the end of our script will be available the next time we run this
- `isset()` directive is use to determine if a particular variable exists in the `$_SESSION` array. This will tell us if this is our first time with this session or if we're coming back to a session that has been previously established. If it's the first time then it will be initialized to 0
- The first time this will run it will give us output 1
- Refreshing the browser and it will print 2 and so on.

```
<?php
// Start a new session; this will create the
// $_SESSION variable
session_start();
// Initialize our persistent variable
if(isset($_SESSION['number']) == false)
{
    $_SESSION['number'] = 0;
}
// ...increment it...
$_SESSION['number'] += 1;
// ...and show the visitor what we've done.
echo $_SESSION['number'];
?>
```

Session Management and PHP

- In this example we gave a name to the session

`session_id('spectrum');`

- Anyone using this ID can access the session and session variables.
- Security reasons, it is a bad idea as it will allow everyone to share and potentially update the same values

```
<?php
// Give the session an ID
session_id('spectrum');
// Start the session using the ID provided
session_start();
if(isset($_SESSION['number']) == false)
{
    $_SESSION['number'] = 0;
}

$_SESSION['number'] += 1;
echo $_SESSION['number'];
?>
```

Session Management and PHP

- In this example we are initializing a session variable along with the cookie on the user's browser containing their current session ID.
- The browser will expire the cookie in 24 hours but before that we can cache the session ID on the workstation

```
<?php
// Look for cookie value from the user's browser.
// If it's set, we'll assume
// that the cookie holds the session ID to use.
if(isset($_COOKIE['spectrum']))
{
    session_id($_COOKIE['spectrum']);
}

session_start();
if(isset($_SESSION['number']) == false)
{
    $_SESSION['number'] = 0;

    // Initialize the cookie
    // expire in 86400 seconds (1 day)
    $expire = time() + 86400;

    setcookie('spectrum',session_id(),$expire);
}

$_SESSION['number'] += 1;
echo $_SESSION['number'];
?>
```

Session Management and PHP

Adding Security

- A further security is added to the session
- IP address of the user is stored in the session variable on the webserver
- Next time the PHP loads the session, if the IP address of the session does not match the user's IP address, we can destroy the current session or unset the current session and start a brand new session
- The drawback is when a user frequently visit the site from different IP addresses it will be considered as a hacker and will keep on resetting the session

```
<?php
if(isset($_COOKIE['spectrum']))
{
    session_id($_COOKIE['spectrum']);
}
session_start();
// Check to see if this session has an embedded IP
// address. If it does, we need to compare this to the
// current visitor's IP address and reset the session if they
// aren't the same.
if(isset($_SESSION['ipAddr']))
{
    if($_SESSION['ipAddr'] !=
    $_SERVER['REMOTE_ADDR'])
    // delete the session file on the server
    session_destroy();
    // Unload session variables in memory
    unset($_SESSION);
    session_start();// Start a new session
}
}
// Initialize our persistent variable and cookie
if(isset($_SESSION['number']) == false)
{
    $_SESSION['number'] = 0;
    $_SESSION['ipAddr'] = $_SERVER['REMOTE_ADDR'];
    // expire in 86400 seconds (1 day)
    $expire = time() + 86400;
    setcookie('spectrum',session_id(),$expire);
}
$_SESSION['number'] += 1;
echo $_SESSION['number'];
?>
```

Session Management in Express.js

- You need express-session and cookie-parser modules:

npm init -y

npm install express express-session cookie-parser

```
var cookieParser = require('cookie-parser');
var session = require('express-session');
const oneDay = 1000 * 60 * 60 * 24;
app.use(session({
  secret : 'ZEIT3119',
  cookie: { maxAge: oneDay },
  resave : true,

  saveUninitialized : true
}));
```

A unique string key used to authenticate a session

Enables the session to be stored back to the session store

Allows any uninitialized session (those created but not modified) to be sent to the store

Session Management in Express.js (Cont.)

- Setting session parameters:

```
req.session.user_id = data[count].id;  
req.session.user_role = data[count].role;
```

- Store the client IP:

```
var ip = req.headers['x-real-ip'] || req.connection.remoteAddress;  
req.session.ip = ip;
```

- Sending session parameters to pages:

```
res.render('index', { title: 'Express', session : req.session });
```

- Destroying a session

```
request.session.destroy();
```


Session Variable

Problem Dealing with dynamic Web Pages

- How do we maintain a user's identity across multiple pages?
- How do we pass data from page to page?

Session Variables

- They enable to track session information about the user through various pages on your site
- PHP sessions are like server-side cookie files. Each one stores variables that are unique to the user request that created it and ideally can be accessed only on subsequent requests from that user.
- Hackers try to turn this functionality into a vulnerability to gain access to resources. There are session attacks that you must attempt to counter.

Types of Session Attacks

- Session Fixation
- Session Hijacking

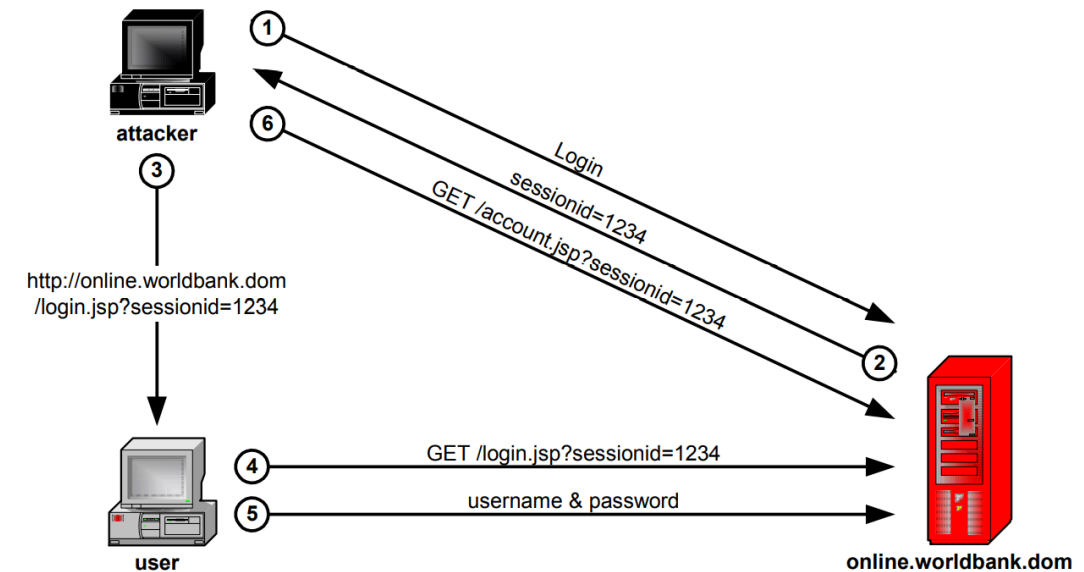
Session Fixation

- It is an attack that allows a hacker to hijack a valid user session
- Type of attacks on web application users where an attacker is able to trick a victim into using a Session ID which is previously known to them.
- When the victim makes use of the known Session ID in their requests to a vulnerable application, the attacker can exploit this vulnerability to make their own requests using the same Session ID – carrying out actions as if they were the legitimate owner of the Session.
- A typical scenario involves the attacker prompting their victim into clicking on a link which directs them to sign in, while also supplying a Session ID.
- It is a very basic attack
- PHP has a very good defense for this type of attack, in the built-in `session_regenerate_id()` function. This function generates a new session file for the user, gets rid of the old one, and issues a new session cookie if your site utilizes them.

Session Fixation

How it Works

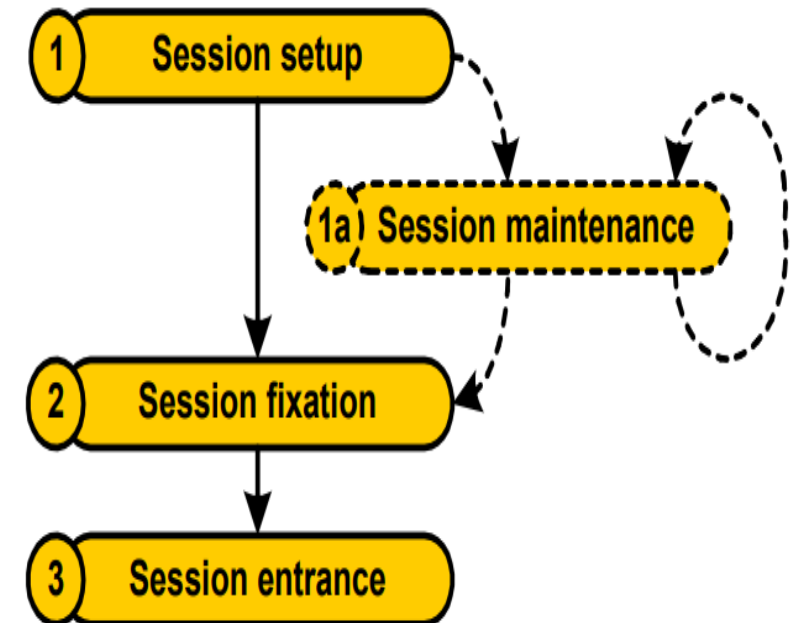
1. The attacker logs in the server as a legitimate user.
2. A session ID is issued to the attacker 1234
3. Attacker sends a hyperlink
<http://online.worldbank.dom/login.jsp?sessionid=1234> to the user, trying to attract the user by clicking on it
4. The user clicks on the link that opens the server login page on the web browser.
5. Upon receipt of the request for login.jsp?sessionid=1234, the web application has established that a session already exists for this user and a new one need not be created. Finally, the user provides his credentials to the login script .
6. The server grants him the access to the bank account. Knowing the Session ID, the attacker can also access the user's account via account.jsp?sessionid=1234
7. Since the session has already been fixed before the user logged in, we say that the user logged into the attacker's session.



Session Fixation

Process

- Session Fixation is a three-step Process
 1. **Session Setup:** The attacker sets up the “trap session” on the target server and obtain the session ID or selects arbitrary session ID to be used in the attack. To avoid session time out repeatedly the requests are sent
 2. **Session Fixation:** Attacker introduce the session ID to the user’s browser, thereby fixing the user session
 3. **Session Entrance:** Attacker waits for the user to login itself to the target server by using the previously fixed session ID and finally enters the user’s session



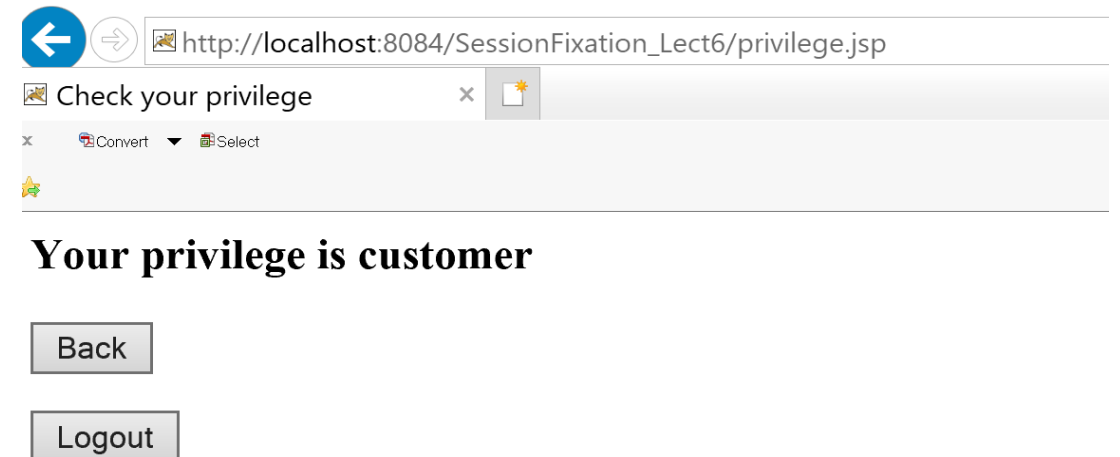
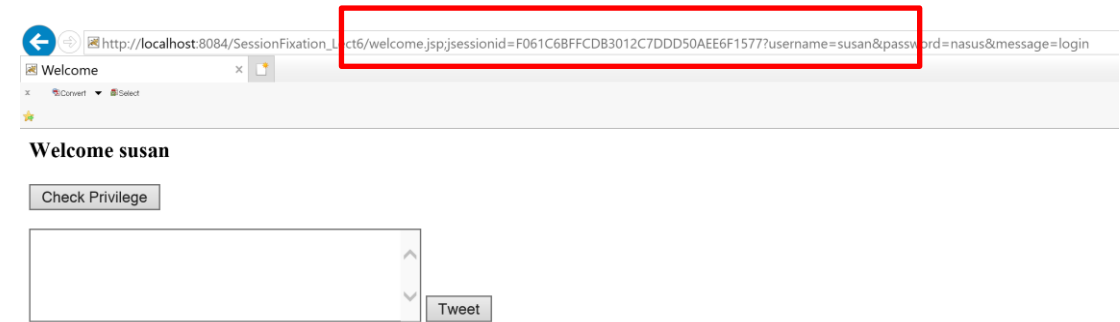
Session Fixation

- There are three different ways an attacker can choose to transport the **trap session ID** to the victim:
- **Using the URL:** User needs to enter the target server link sent by the attacker. It can easily be detected.
- **Using Session ID in a hidden field form:** The attacker needs to develop a look a like form of the original website. The form should also look like it is coming from a legitimate server. It is less practical version of fixation
- **Using Session ID in a cookie:** Most popular session ID transport mechanism. This provides the most convenient and effective and durable means of exploiting session fixation durability's. The attacker install the trap session ID cookie on the user's browser

Session Fixation

Example

- Session ID also appear on the URL once Susan logged in
- Susan checked her status and she is a privilege customer



Session Fixation

Susan Launch an Attack

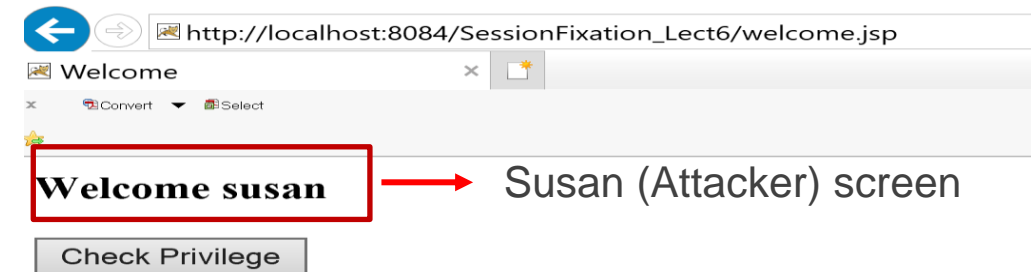
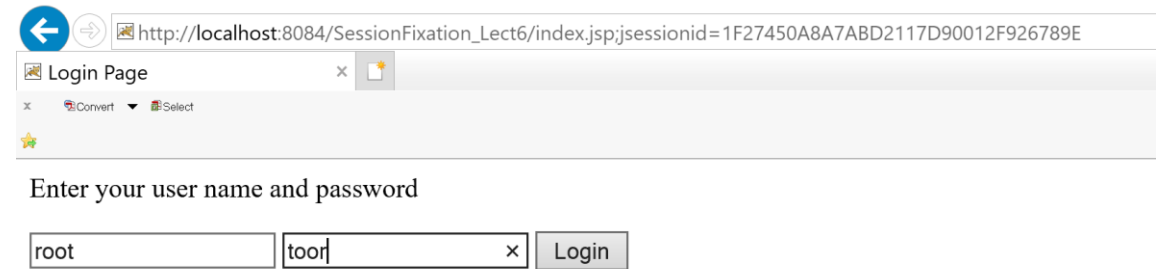
- Susan tailored the link adding in the session ID and sends to victim

http://localhost:8084/SessionFixation_Lect6/index.jsp;jsessionid=1F27450A8A7ABD2117D90012F926789E

- Victim clicked on the link sent by Susan and it logs in as admin with

username: root, password: toor

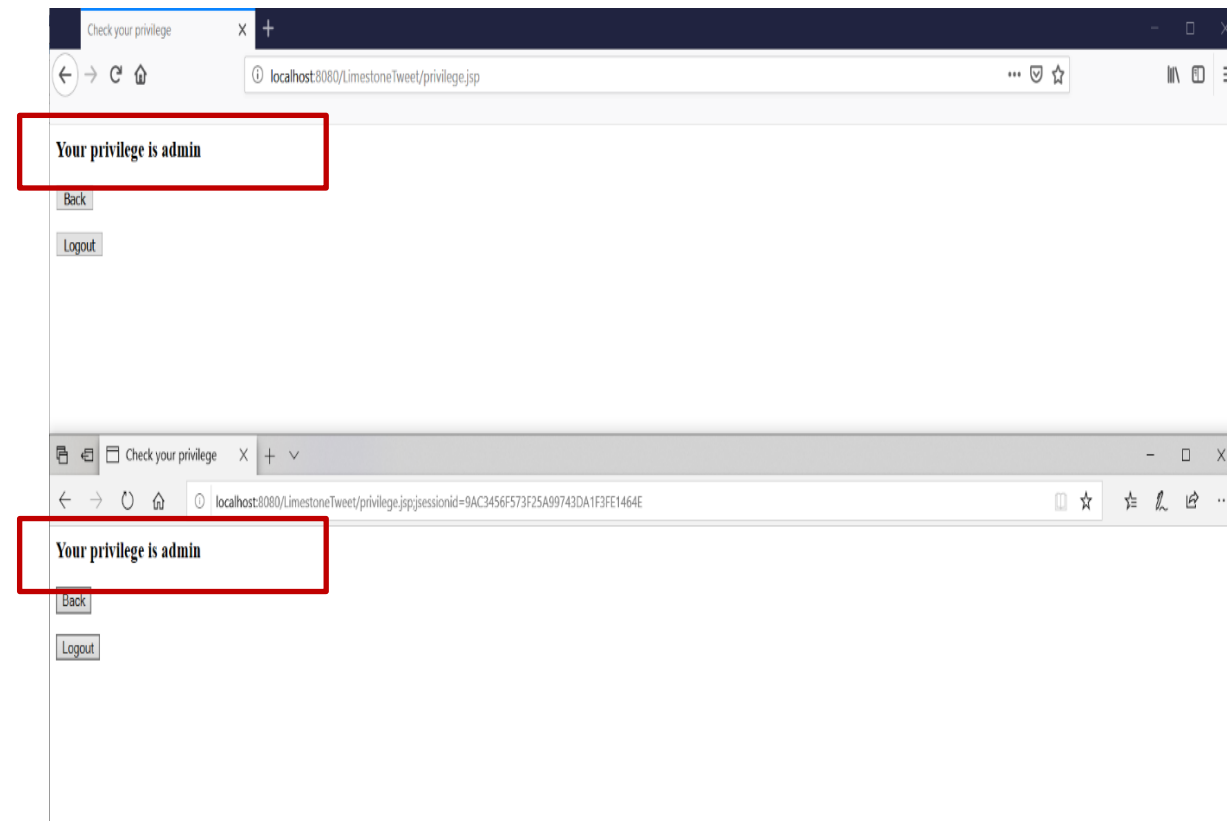
- Once the victim logs in it displays a welcome message to victim as well as to Susan (attacker)



Session Fixation

Susan Launch an Attack



- Once they check the privilege, victim and attacker both will gain the same privilege.
- Web application identifies any request based on session ID
- Any browser that passes such session ID can request all attributes that belong to that ID. In this case, the attribute is privilege=admin.
- When Susan and victim clicked the “Check Privilege” button, they are both identified as an admin.




Session Fixation Demo

localhost/lab7/menu.php

Menu

Item	Description	Price	Pic	Order
Burger	A juicy beef burger with all the fixings	10.99		<input type="text"/>
Pizza	A delicious pizza with your choice of toppings	12.99		<input type="text"/>

 \$0

A customer logged in

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

Cache Storage Filter Items

Cookies

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Last Accessed
connect.sid		localhost	/	Session	11	true	false	None	Sun, 11
PHPSESSID	dqv5m38a495dr12scgt5aksbfn	localhost	/	Session	35	false	false	None	Sun, 11

Indexed DB

Local Storage

PHPSESSID

Session Fixation Demo (Cont.)

localhost/lab7/signin.php/?PHPSESSID=dqv5m38a495dr12scgt5aksbfm

PHPSESSID

Sign In

Email:

jack@gmail.com

Admin User

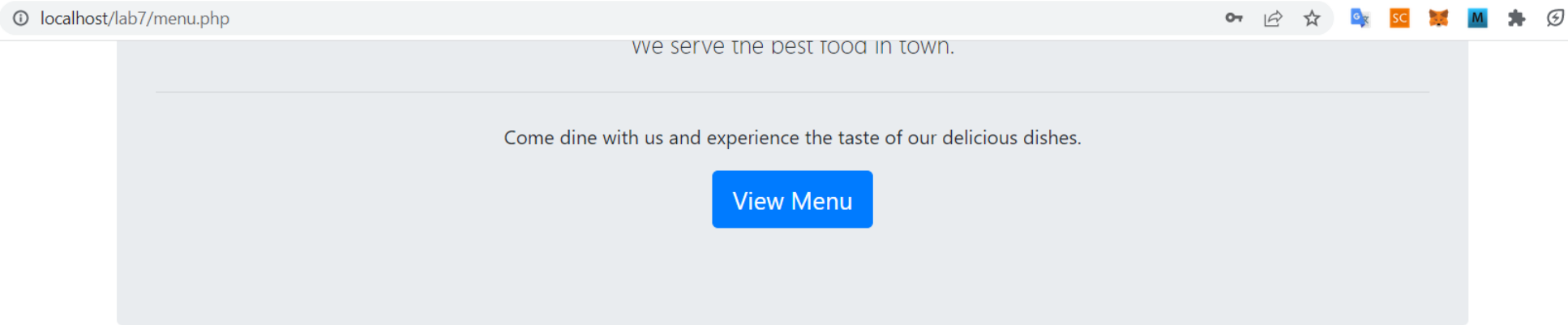
Password:

....



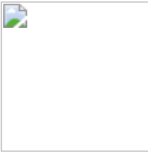

Don't have an account? [Sign up now.](#)

Sign in

Session Fixation Demo (Cont.)



Menu

Item	Description	Price	Pic	
Burger	A juicy beef burger with all the fixings	10.99		
Pizza	A delicious pizza with your choice of toppings	12.99		

Admin logged in

Description:

Price:

Select an image: No file chosen





Session Fixation Demo (Cont.)

localhost/lab7/menu.php



Customer refreshes the page and becomes admin! n

Menu

Item	Description	Price	Pic	
Burger	A juicy beef burger with all the fixings	10.99		
Pizza	A delicious pizza with your choice of toppings	12.99		

Description:

Price:

Select an image: No file selected.

Session Fixation Prevention

- The following code can be added in the login code and it can prevent session fixation attacks.
- This code ensures that any user who logs in is assigned a fresh random session identifier

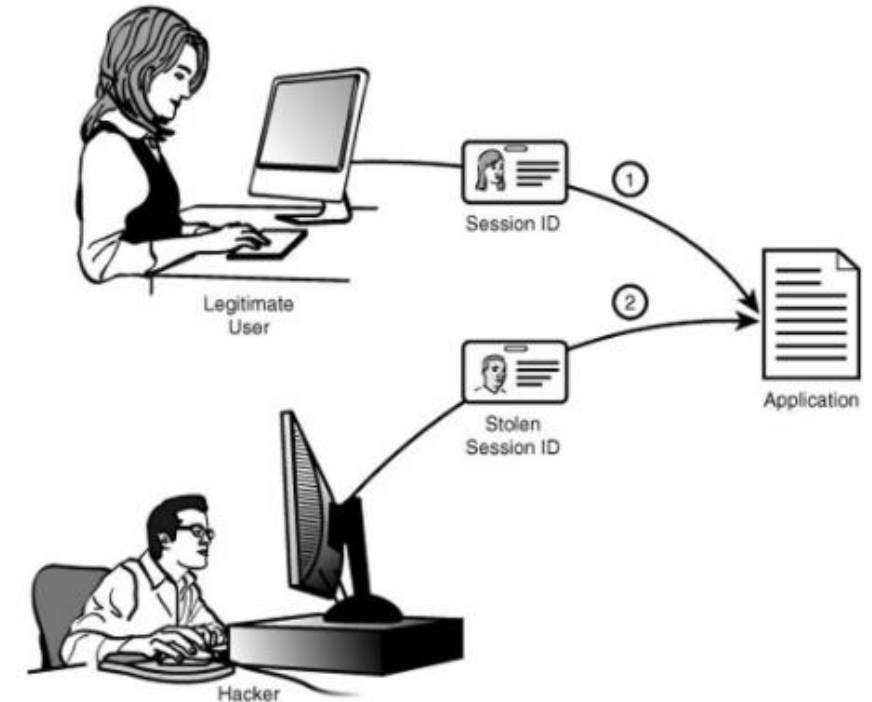
```
<?php  
  
session_start();  
  
if (!isset($_SESSION['initiated'])) {  
    session_regenerate_id();  
    $_SESSION['initiated'] = TRUE;  
}  
  
?>
```

Mitigation Technique

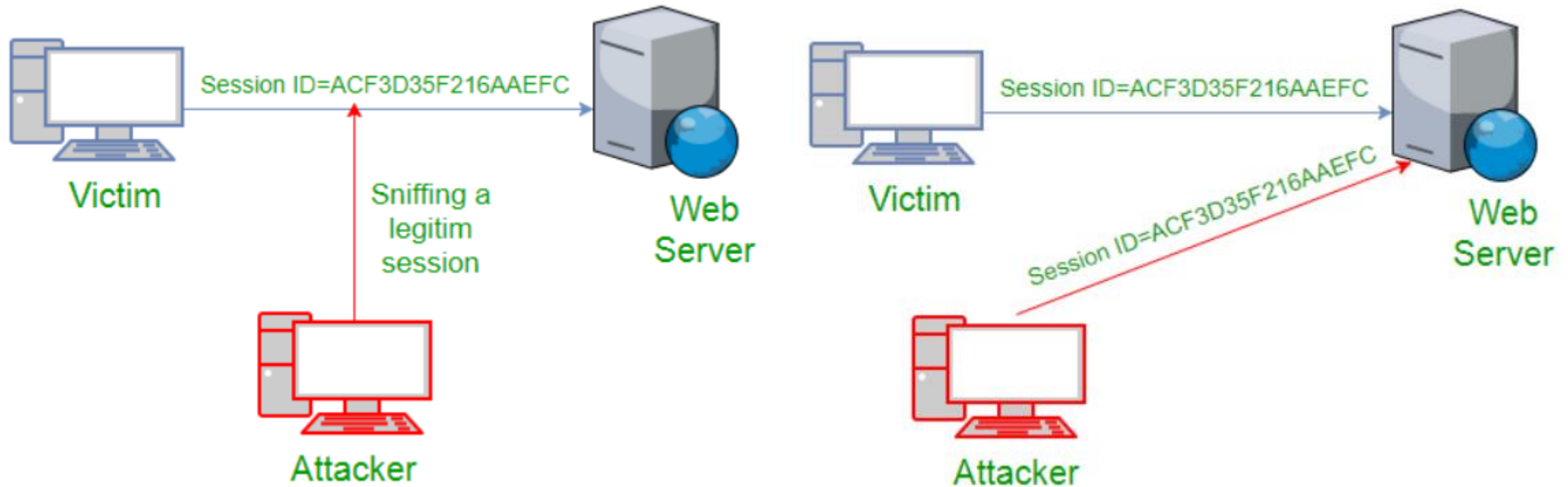
- **Prevent logins to a chosen session:** Web applications must deny any session ID's that are presented by the web browser at the login
- **Preventing Attacker from avoiding a valid session ID:** Session IDs should be assigned once the user has been authenticated.
- **Restricting the session ID usage:** Stolen session ID can be the cause of the session fixation. Sessions should be deleted on the server side and on the browser
- **User should logout:** It will not only destroy the current session but also destroy the previous sessions

Session Hijacking

- In a hijacking attack, the malicious user tries to access victim's site utilizing a valid session ID
- Attacker attacks the user's browser after victim logs in to the target server.
- There are a number of steps we can take to defend against a session hijacking.
- User agent verification
 - Identifies the user's identity using [HTTP_USER_AGENT](#)
- IP address verification
 - Store the users' IP when it first generates the session, and then on every page load and verify that IP address
- Secondary token
 - Two points of verification are set up.
 - Creating a token and Session ID
 - Token is stored in session ID



Session Hijacking (Cont.)



Session Hijacking Techniques

➤ Cross Site Scripting(XSS Attack)

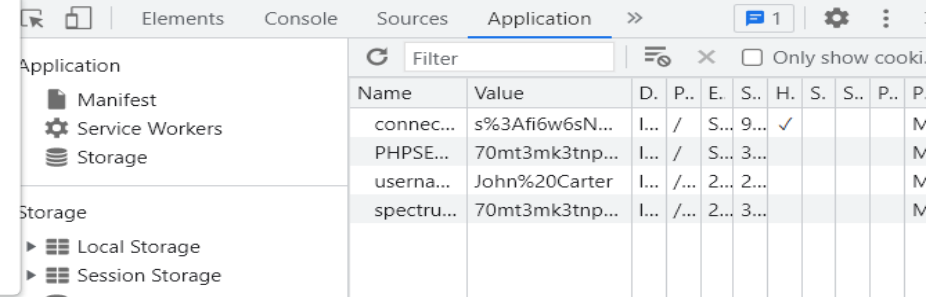
```
<!DOCTYPE html>
<html>
  <head>
    <SCRIPT type="text/javascript">
      var adr = '../attacker.php?victim_cookie=' + escape(document.cookie);
      alert(adr);
    </SCRIPT>
```

localhost/lab10/sessionhijacking.html

localhost says

../attacker.php?
victim_cookie=spectrum%3D70mt3mk3tnp7kb0230octd74hp%3B%20usernam
e%3DJohn%2520Carter%3B%20PHPSESSID%3D70mt3mk3tnp7kb0230octd74hp

OK



Name	Value	D.	P.	E.	S.	H.	S.	S.	P.	P.
conne...	s%3Afi6w6sN...	I...	/	S...	9...	✓				M
PHPSE...	70mt3mk3tnp...	I...	/	S...	3...					M
userna...	John%20Carter	I...	/...	2...	2...					M
spectru...	70mt3mk3tnp...	I...	/...	2...	3...					M

➤ IP Spoofing

➤ Blind Attack

Session Fixation Vs Session Hijacking

Session Fixation	Session Hijacking
Attacker attacks the user's browser before he logs in to the target server.	Attacker attacks the user's browser after he logs in to the target server.
Attacker gains one-time, temporary or long-term access to the user's session(s).	Attacker usually gains one-time access to the user's session and has to repeat the attack in order to gain access to another one.
Can require the attacker to maintain the trap session until the user logs into it.	Requires no session maintenance.
Involves communication link, target web server, all hosts in target server's domain, user's DNS server	Involves communication link, target web server

Session Fixation Vs Session Hijacking

Session Fixation	Session Hijacking
<p>Attack:</p> <ul style="list-style-type: none">➤ Tricking the user to log in through a malicious hyperlink or a malicious login form➤ Breaking into any host in the target server's domain➤ Adding a domain cookie-issuing server to the target server's domain in the user's DNS server➤ Network traffic modification	<p>Attack:</p> <ul style="list-style-type: none">➤ Obtaining the session ID in the HTTP referrer header sent to another web server➤ Network traffic sniffing (only works with an unencrypted link to the target server)

Cookies

- It is a text file with small amount of data e.g.,
 - a username or a password
 - time of visiting a website
 - items added in the cart
- It is just a name-value pair and is stored at user's computer.
- Cookies are created at the server side
- **HTTP cookies:**
 - They are used to identify the user and improve the web browsing
 - First time a web browser connects with a server , there are no cookies.
 - When a server responds it includes a *Set-Cookie:* header that defines a cookie
 - In the future whenever the browser connects with the same server, it includes a *Cookie:* header containing the name and value, which the server can use to connect related requests.
 - A server can define multiple cookies with different names, but browsers limit the number of cookies per server (around 50).

```
setcookie(name, value, expire, path, domain, secure);
```

Cookies

- **Session Cookie:**
- Cookies are used by the server to implement *sessions*.
- A session cookie only lasts for the duration of users using the website.
- A web browser normally deletes session cookies when it quits.
- A session cookie expires if the user does not access the website for a period chosen by the server (idle timeout). If someone comes and uses our computer, they would not be able to see anything on the sites that use session cookies, because they need to enter the username and password again.
- Typically, the cookie for an application contains an identifier for a session.
- Domain for cookie: server, port (optional), URL prefix (optional). The cookie is only included in requests matching its domain.
- The signature of creating the cookie is:
- Cookie (string name, string value)
 - `Cookie c = new Cookie ("lastpage", "3")`
 - A cookie c is created with the parameter lastpage that has a string value 3

Cookies- Example

Setting a Cookie and verifying whether a cookie is set or not

```
<?php
// Setting a cookie

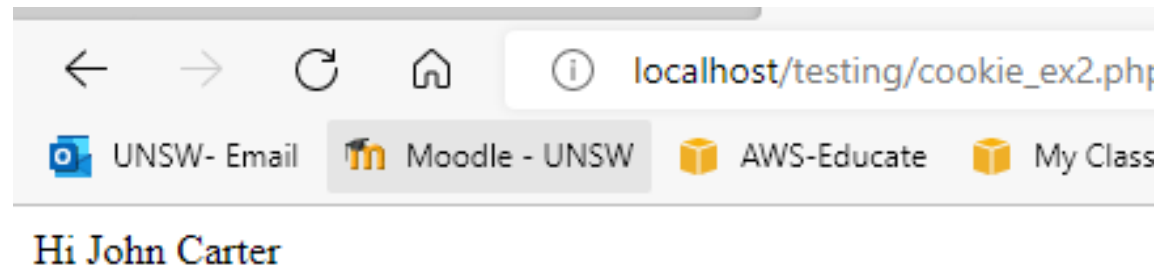
setcookie("username", "John Carter", time()+30*24*60*60);

if(isset($_COOKIE["username"])){
    echo "Hi " . $_COOKIE["username"];
} else{
    echo "Welcome Guest!";
}

?>
```

Deleting a Cookie

```
<?php // Deleting a cookie
setcookie("username", "", time()-3600);
?>
```



Difference between Cookie and Session

Cookies

- ☐ Cookies are stored on client side
- ☐ Cookies can only store strings.
- ☐ Cookies can be set to a long lifespan.

Sessions

- ☐ Sessions are stored on server side
- ☐ Sessions can store objects.
- ☐ When users close their browser, they also lost the session.

Final Note

- Please attend the labs this week and discuss Project 2 requirements with your lab demonstrators