Tryhackme | OWASP Broken Access Control | writeup

Sudarshan Patel · Follow
10 min read · Jan 8, 2024

Broken access controls are a type of security vulnerability that arises when an application or system fails to properly restrict access to sensitive data or functionality. This vulnerability allows attackers to gain unauthorized access to resources that should be restricted, such as user accounts, files, databases, or administrative functions. Broken access controls can occur due to a variety of factors, including poor design, configuration errors, or coding mistakes.

Objectives that the student will learn:

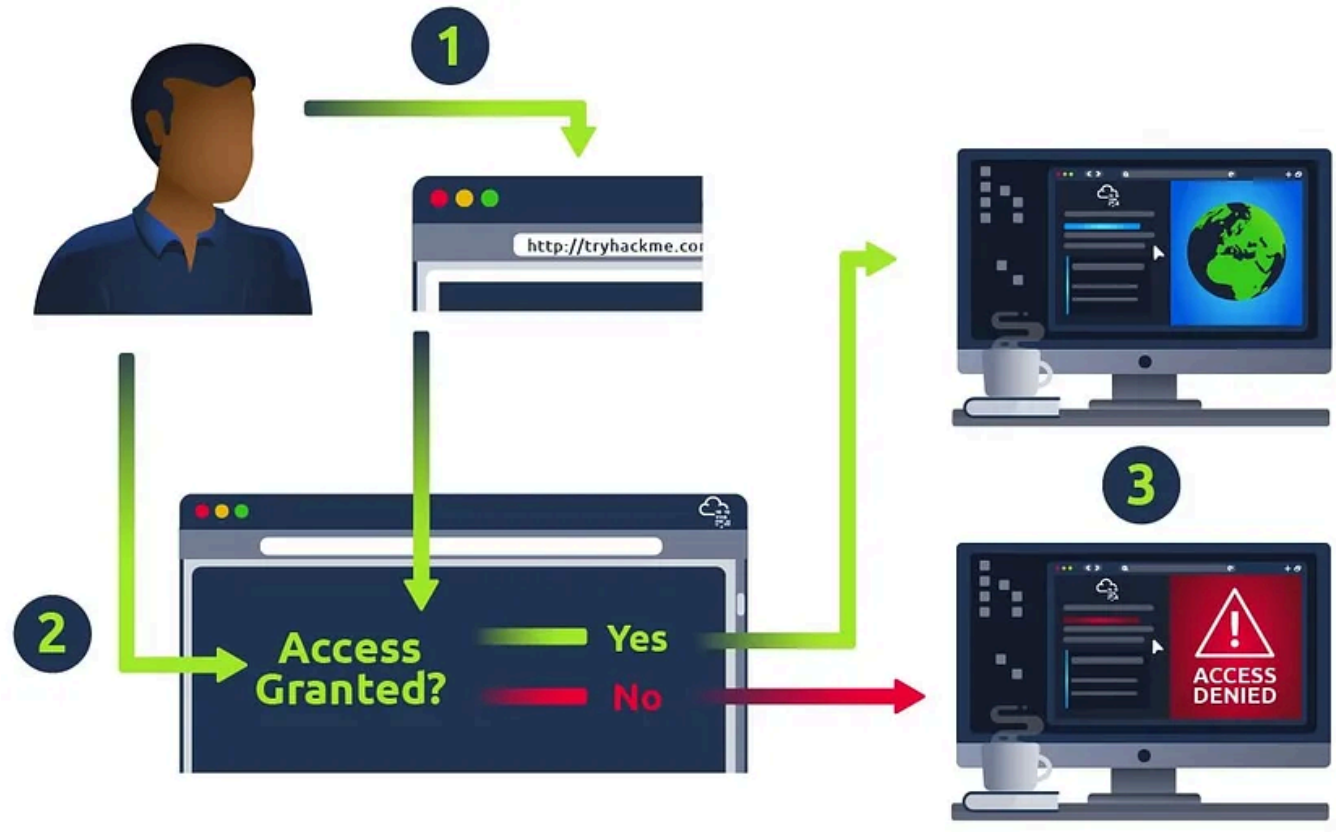
1. Understand what Broken Access Control is and its impact.
2. Identify Broken Access Control vulnerabilities in web applications.
3. Exploit these vulnerabilities in a controlled environment.
4. Understand and apply measures to mitigate and prevent these vulnerabilities.

Pre-requisites:

1. Basic understanding of JSON, web applications, and HTTP protocols.
2. Familiarity with scripting languages such as PHP and JavaScript.
3. Knowledge of web application security standards and frameworks such as OWASP Top 10.
4. Basic understanding and usage of a proxy tool like Burp Suite.

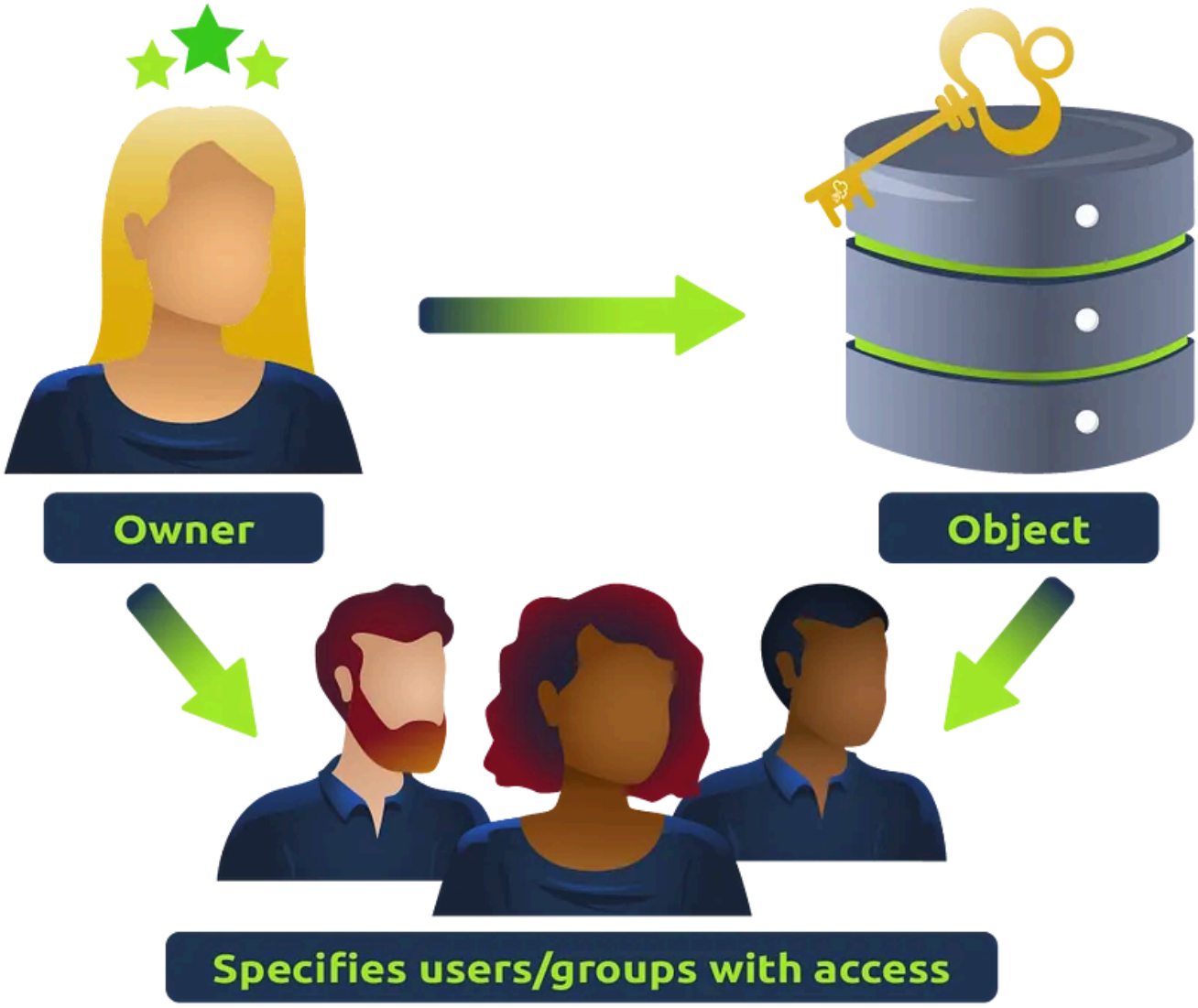
What is Access Control?

Access control is a security mechanism used to control which users or systems are allowed to access a particular resource or system. Access control is implemented in computer systems to ensure that only authorized users have access to resources, such as files, directories, databases, and web pages. The primary goal of access control is to protect sensitive data and ensure that it is only accessible to those who are authorized to access it.



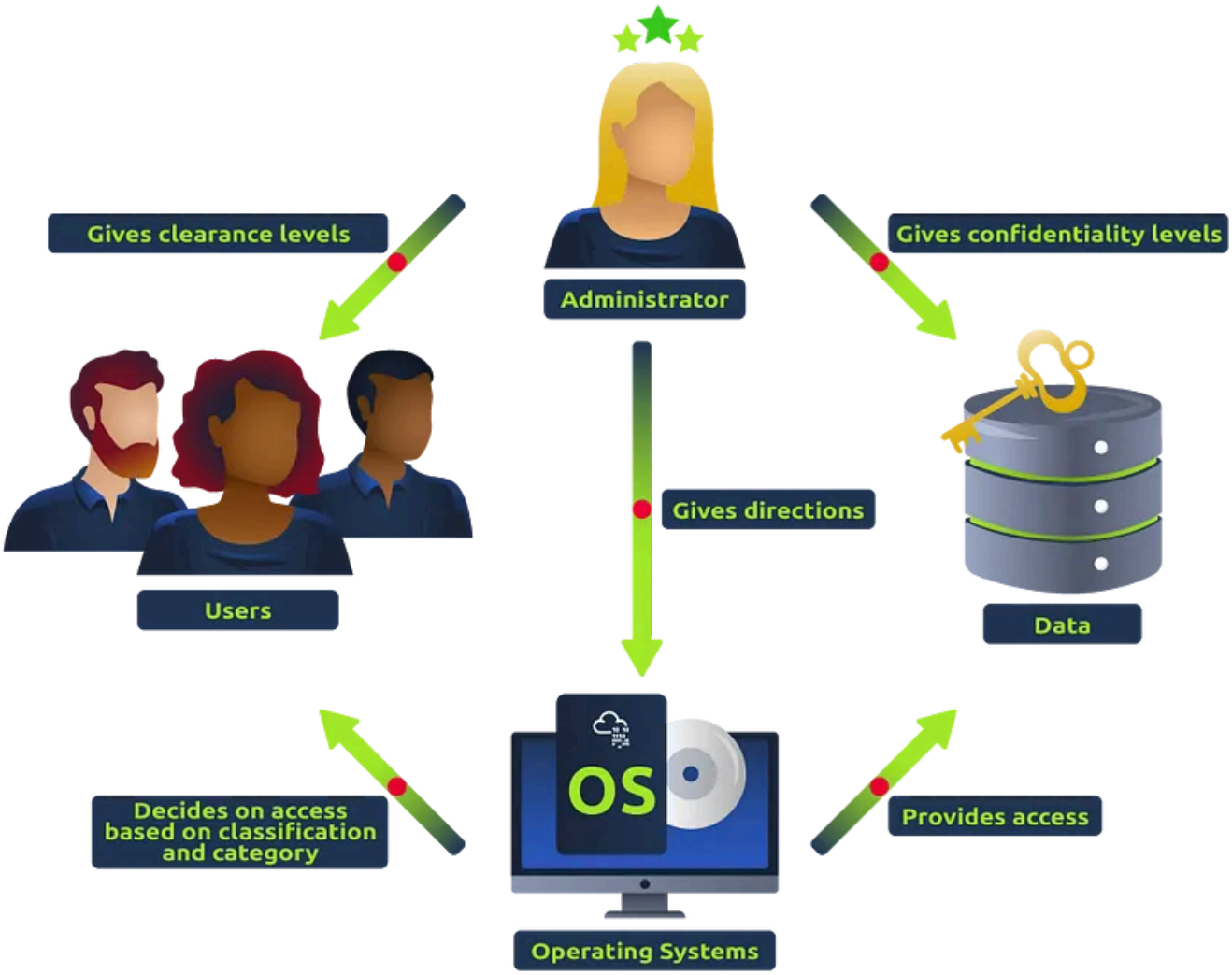
Access control can be implemented in different ways, depending on the type of resource being protected and the security requirements of the system. Some common access control mechanisms include:

Discretionary Access Control (DAC):



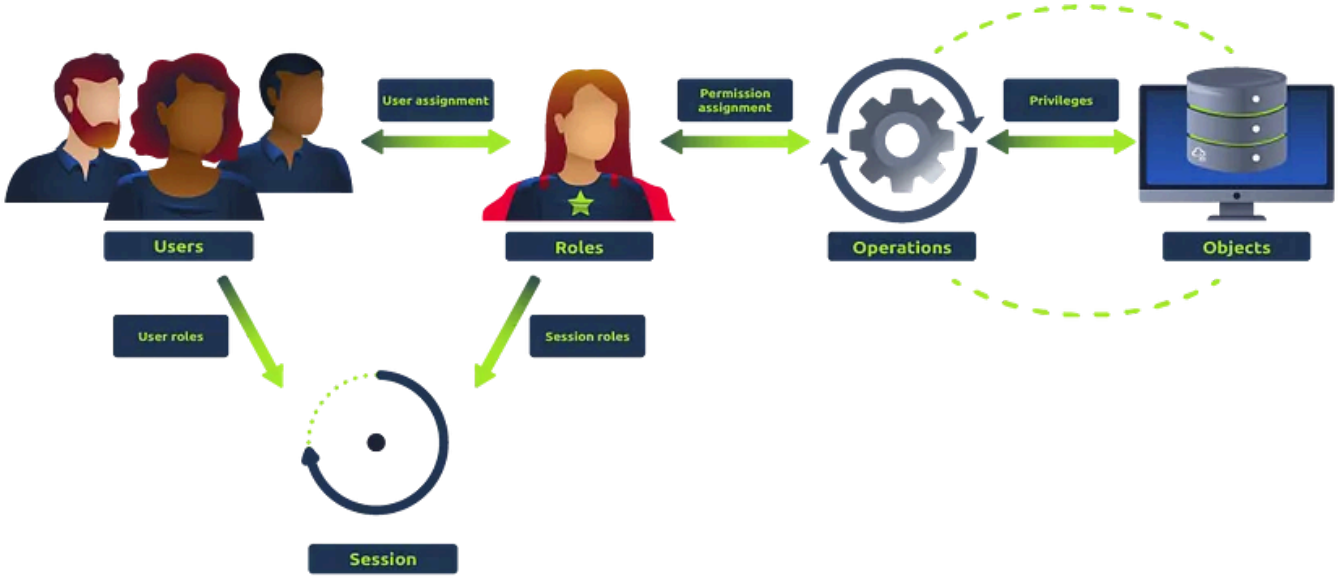
In this type of access control, the resource owner or administrator determines who is allowed to access a resource and what actions they are allowed to perform. DAC is commonly used in operating systems and file systems. In layman's terms, imagine a castle where the king can give keys to his advisors, allowing them to open any doors they like, whenever they want. That's DAC for you. It's the liberty to control access to your own resources. The one in charge, like the king of the castle, can hand out permissions to whomever they please, dictating who can come in and out.

Mandatory Access Control (MAC):



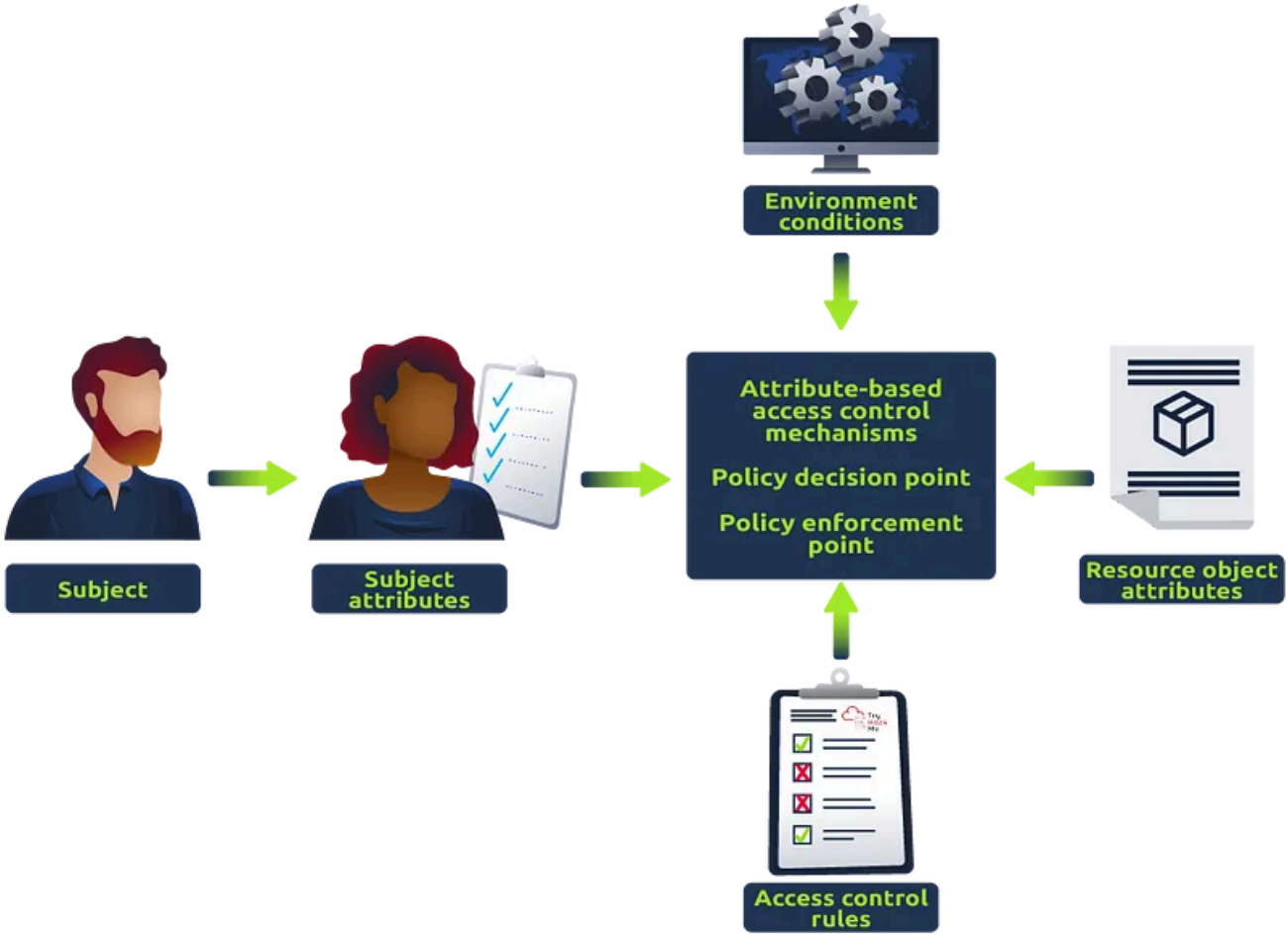
this type of access control, access to resources is determined by a set of predefined rules or policies that are enforced by the system. MAC is commonly used in highly secure environments, such as government and military systems. In layman's terms, picture a fort with an iron-clad security protocol. Only specific individuals with particular security clearances can access certain areas, and this is non-negotiable. The high commander sets the rules, and they are rigorously followed. That's how MAC works. It's like the stern security officer who allows no exceptions to the rule.

Role-Based Access Control (RBAC):



In this type of access control, users are assigned roles that define their level of access to resources. RBAC is commonly used in enterprise systems, where users have different levels of authority based on their job responsibilities. In layman's terms, imagine a modern corporation. You have your managers, your executives, your sales staff, etc. They each have different access to the building. Some can enter the boardroom, others can access the sales floor, and so on. That's the essence of RBAC — assigning access based on a person's role within an organization.

Attribute-Based Access Control (ABAC):



In this type of access control, access to resources is determined by a set of attributes, such as user role, time of day, location, and device. ABAC is commonly used in cloud environments and web applications. In layman's terms, think of a highly advanced sci-fi security system that scans individuals for certain attributes. Maybe it checks whether they're from a particular planet, whether they're carrying a specific device, or if they're trying to access a resource at a specific time. That's ABAC. It's like the smart, flexible security of the future.

Implementing access control can help prevent security breaches and unauthorized access to sensitive data. However, access control is not foolproof and can be vulnerable to various types of attacks, such as privilege escalation and broken access control vulnerabilities. Therefore, it is important to regularly review and test access control mechanisms to ensure that they are working as intended.

Broken Access Control:

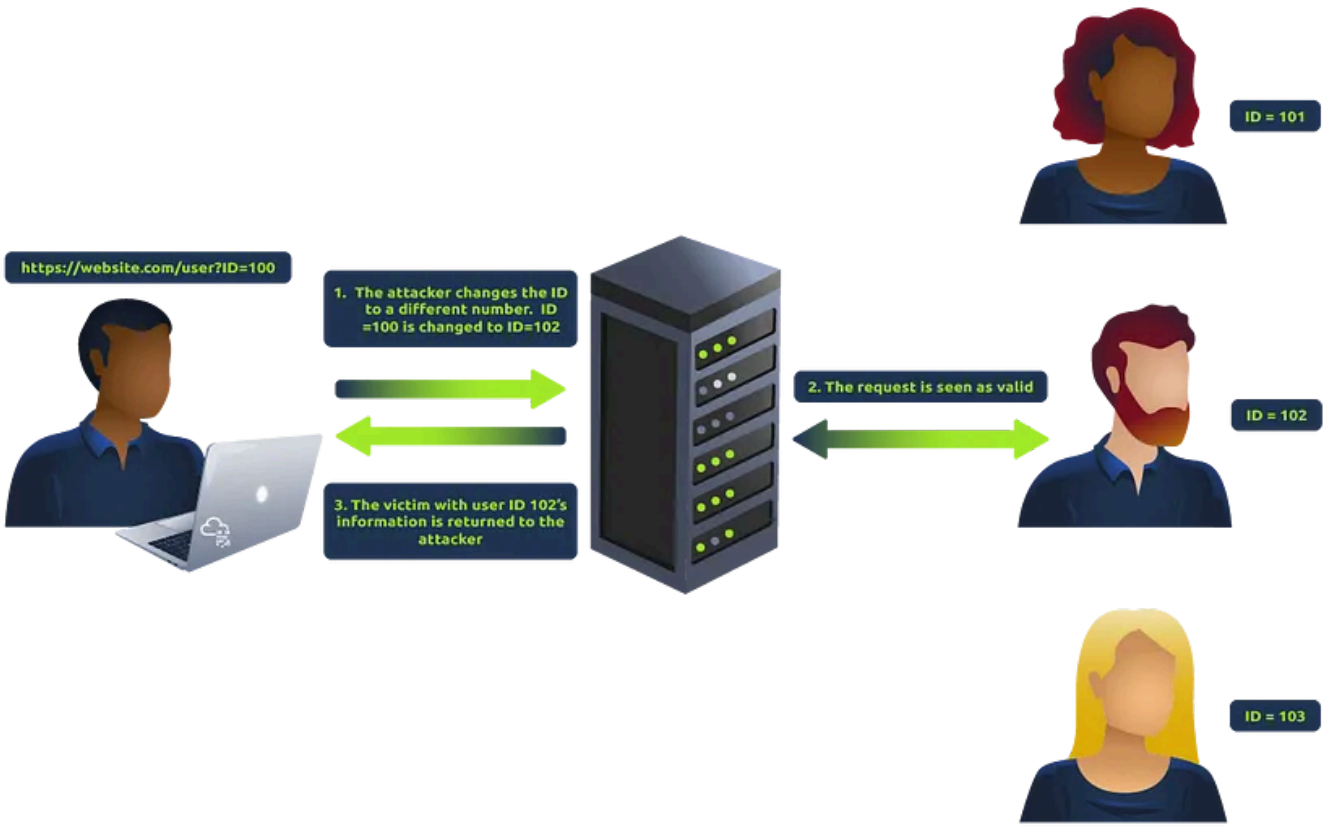
Broken access control vulnerabilities refer to situations where access control mechanisms fail to enforce proper restrictions on user access to resources or data. Here are some common exploits for broken access control and examples:

Horizontal privilege escalation occurs when an attacker can access resources or data belonging to other users with the same level of access. For example, a user might be able to access another user's account by changing the user ID in the URL.

Vertical privilege escalation occurs when an attacker can access resources or data belonging to users with higher access levels. For example, a regular user can access administrative functions by manipulating a hidden form field or URL parameter.

Insufficient access control checks occur when access control checks are not performed correctly or consistently, allowing an attacker to bypass them. For example, an application might allow users to view sensitive data without verifying their proper permissions.

Insecure direct object references occur when an attacker can access a resource or data by exploiting a weakness in the application's access control mechanisms. For example, an application might use predictable or easily guessable identifiers for sensitive data, making it easier for an attacker to access. You may refer to this room in **Task #4** to learn more about this.



These exploits can be prevented by implementing strong access control mechanisms and regularly reviewing and testing them to ensure they are functioning as intended.

Let's get started with the task:

> *Question:*
>
> *What is IDOR?*

Answer: Insecure Direct Object Reference

An IDOR Vulnerability is a type of broken access control vulnerability.

> *Question:*
>
> *What occurs when an attacker can access resources or data belonging to other users with the same level of access?*

Answer: horizontal privilege escalation

Please refer to the horizontal privilege escalation section to get the answer to this question.

> *Question:*
>
> *What occurs when an attacker can access resources or data from users with higher access levels?*

Answer: vertical privilege escalation

Please refer to the vertical privilege escalation section to get the answer to this question.

> *Question:*
>
> *What is ABAC?*

Answer: attribute-based access control

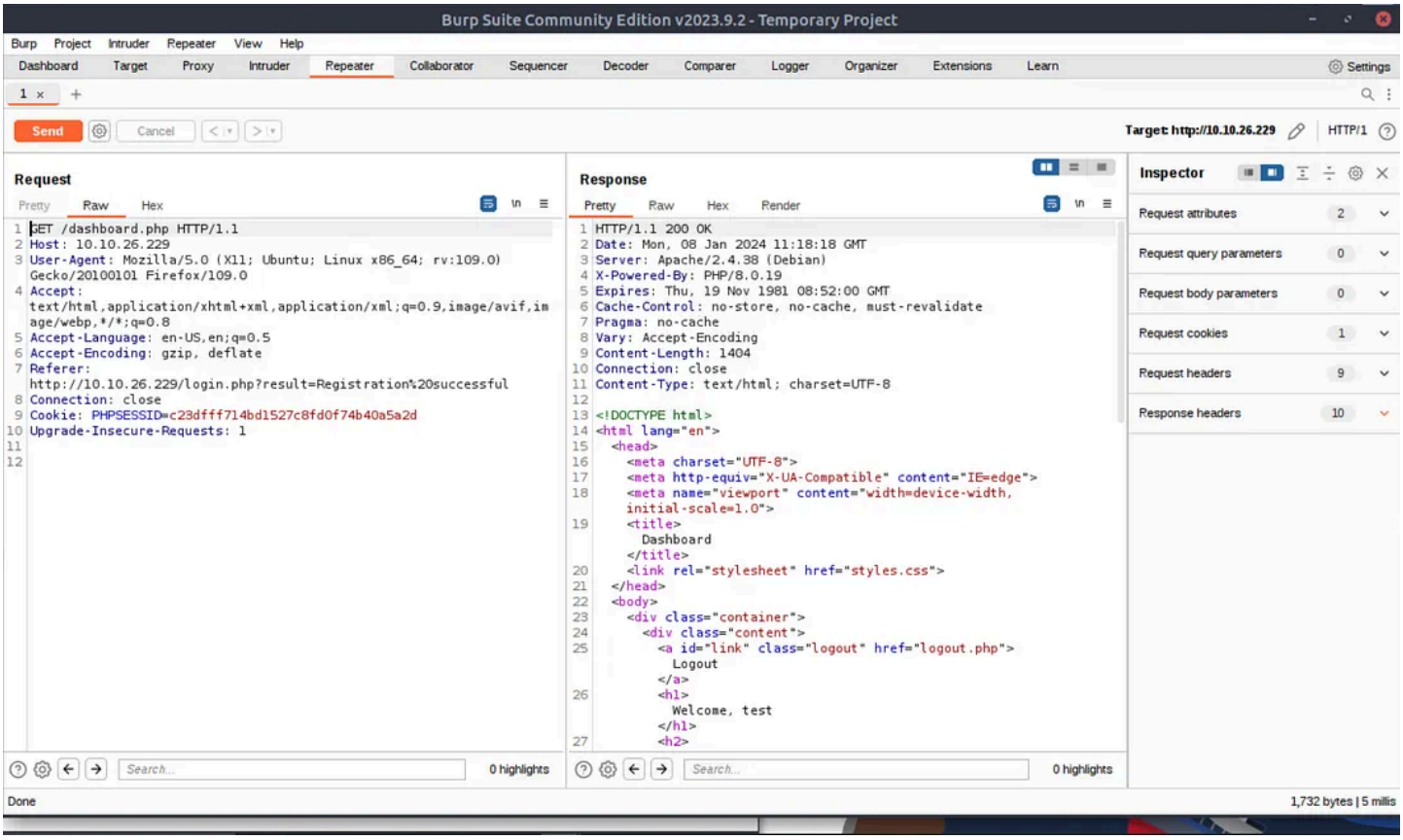
Please refer to the attribute-based access control section to get the answer to this question.

> *Question:*
>
> *What is RBAC?*

Answer: Role-based access control

Please refer to the role-based access control section to get the answer to this question.

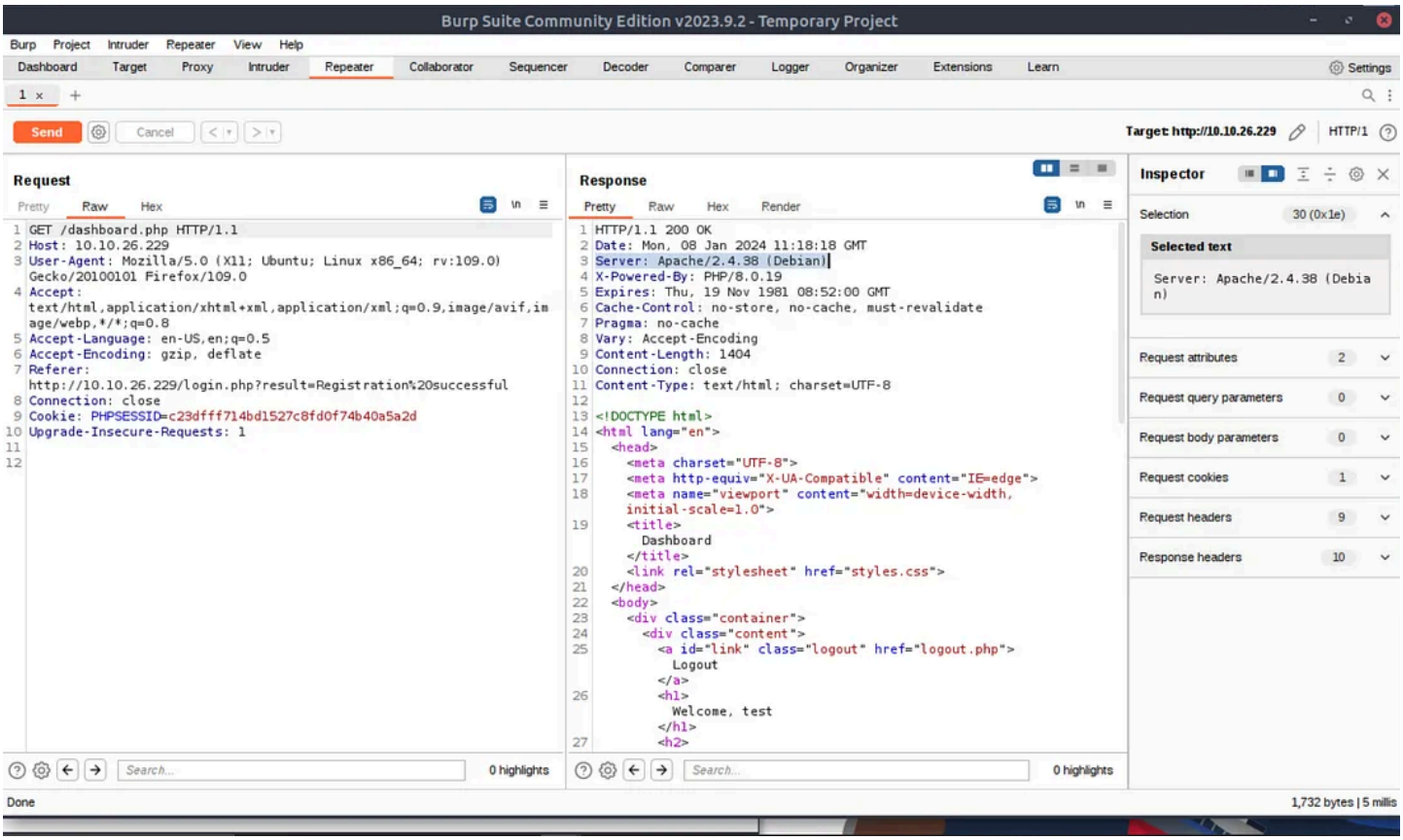
Please register an account by visiting the tryhackme machine-ip and intercept the login request. Sent the request to repeater. Please refer to the below screenshot.



> *Question:*
>
> *What is the type of server that is hosting the web application? This can be found in the response of the request in Burp Suite.*

Answer: apache

In the captured request, look at the response. We will get the type of server and its version number which is the answer to this question. Please refer to the below screenshot. The answer is highlighted in the screenshot.



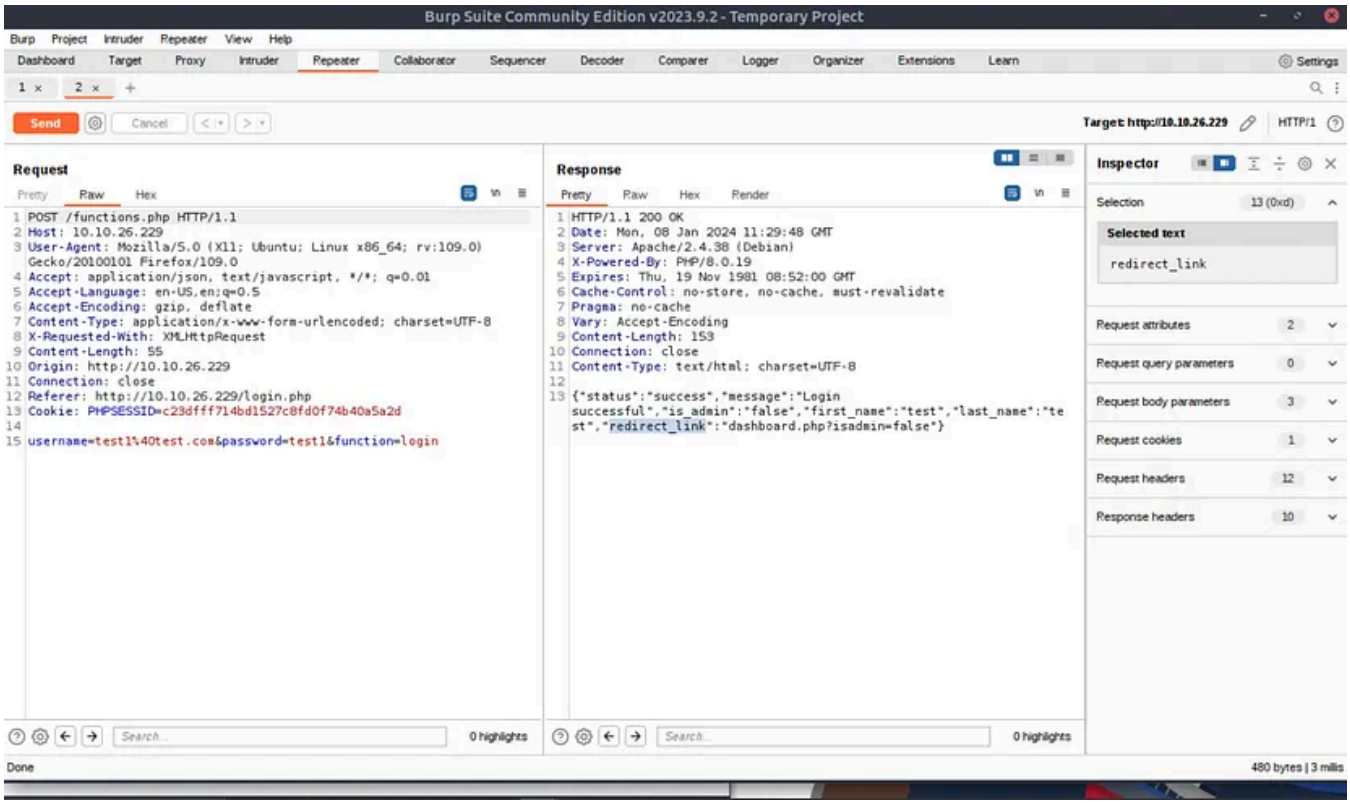
> *Question:*
>
> *What is the name of the parameter in the JSON response from the login request that contains a redirect link?*

Answer: redirect_link

Capture the login request and send it to the repeater.

Observe the response, where you will get the parameter name that contains a redirect link which is the answer to this question.

Please refer to the below screenshot.



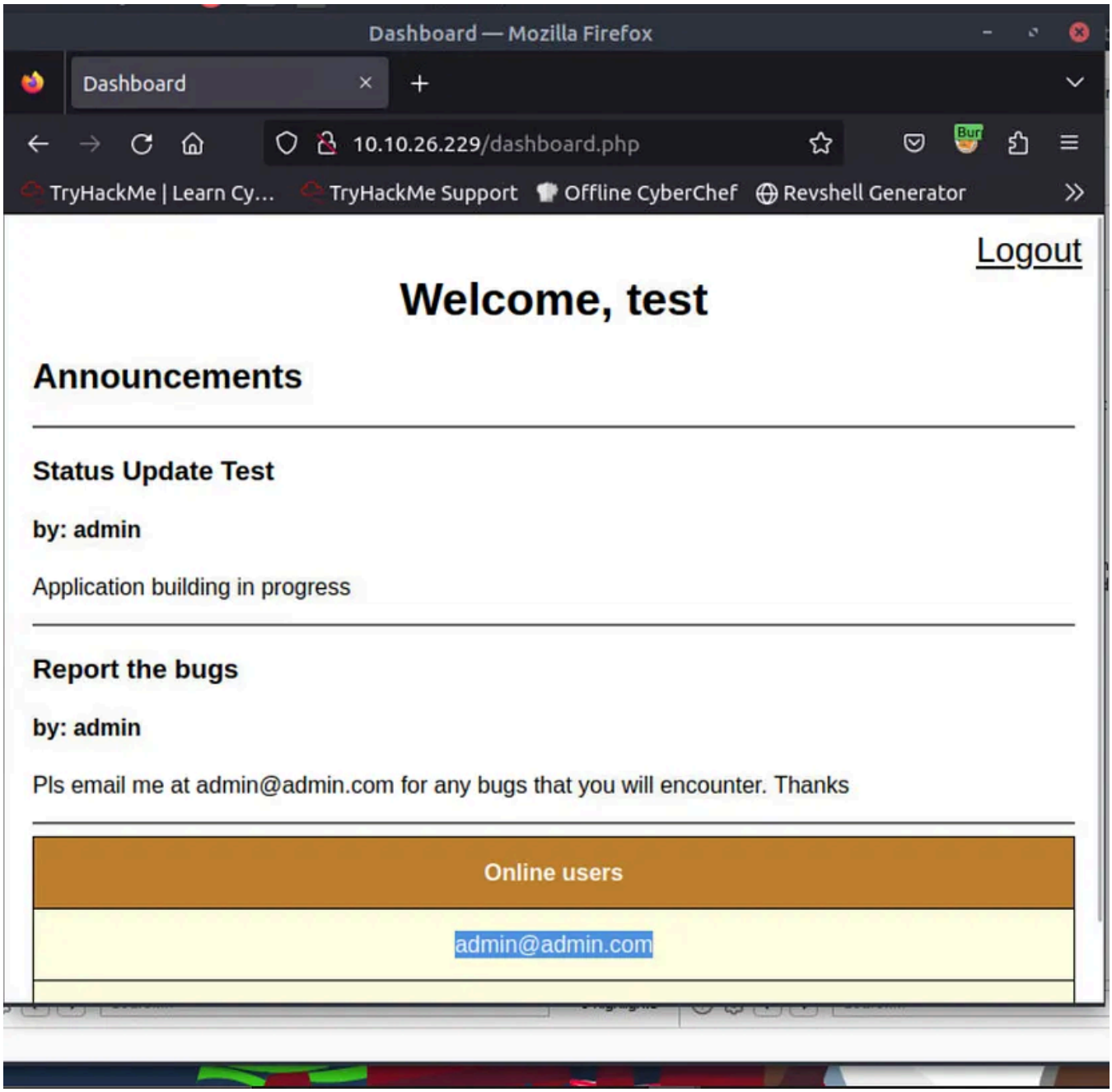
> *Question:*
>
> *What Burp Suite module allows us to capture requests and responses between ourselves and our target?*

Answer: proxy

> *Question:*
>
> *What is the admin's email that can be found in the online users' table?*

Answer: admin@admin.com

Please login to the dashboard. There will been the section below where you can get the answer to this question. Please refer to the screenshot below.



Now looking at the url of the redirect_link, paste the url directly in a new tab and look at the parameter passed in the url.

Next questions are based on the url parameter.

> *Question:*
>
> *What kind of privilege escalation happened after accessing admin.php?*

Answer: Vertical

Here, the attack type is vertical privilege escalation because we are escalating our privileges from user level to admin level.

> *Question:*
>
> *What parameter allows the attacker to access the admin page?*

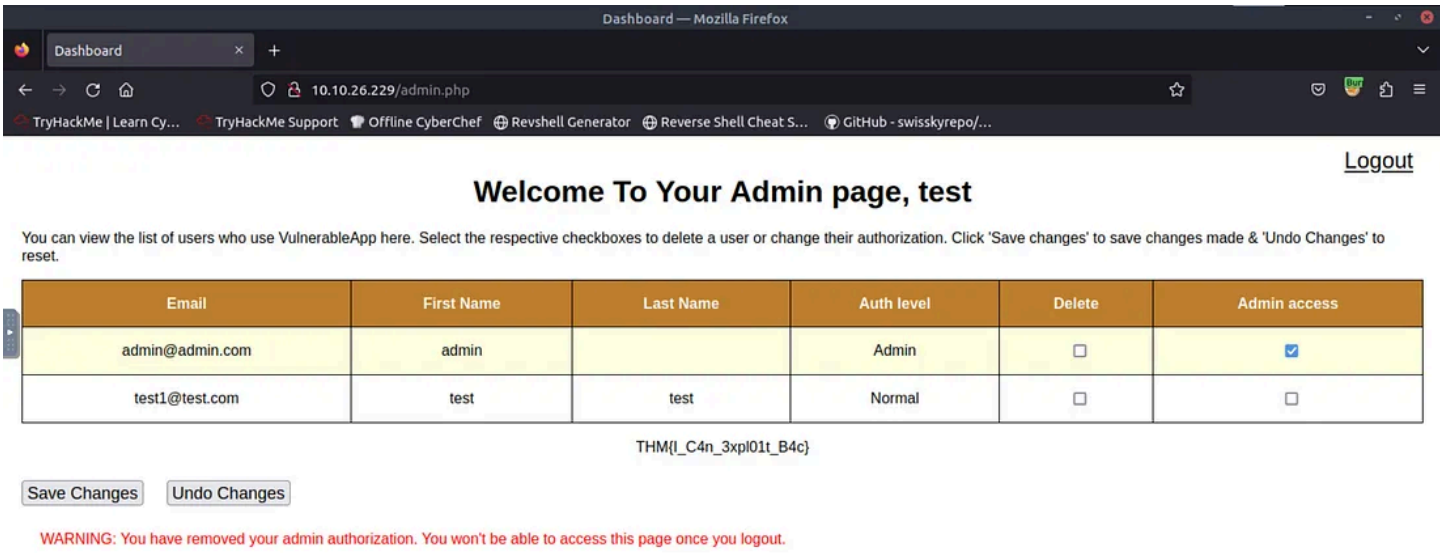
Answer: isadmin

The answer to this question lies in the url itself.

> *Question:*
>
> *What is the flag in the admin page?*

Answer: THM{I_C4n_3xpl01t_B4c}

Copy the url and modify the parameter value from “isadmin=false” to “isadmin=true” and forward the request. Please look at the below screenshot.



Mitigation

There are several steps that can be taken to mitigate the risk of broken access control vulnerabilities in PHP applications:

Implement Role-Based Access Control (RBAC): Role-based access control (RBAC) is a method of regulating access to computer or network resources based on the roles of individual users within an enterprise. By defining roles in an organization and assigning access rights to these roles, you can control what actions a user can

perform on a system. The provided code snippet illustrates how you can define roles (such as ‘admin’, ‘editor’, or ‘user’) and the permissions associated with them. The `hasPermission` function checks if a user of a certain role has a specified permission.

Sample Code

```php
// Define roles and permissions
$roles = [
    'admin' => ['create', 'read', 'update', 'delete'],
    'editor' => ['create', 'read', 'update'],
    'user' => ['read'],
];

// Check user permissions
function hasPermission($userRole, $requiredPermission) {
    global $roles;
    return in_array($requiredPermission, $roles[$userRole]);
}

// Example usage
if (hasPermission('admin', 'delete')) {
    // Allow delete operation
} else {
    // Deny delete operation
}
```

Use Parameterized Queries: Parameterized queries are a way to protect PHP applications from SQL Injection attacks, where malicious users could potentially gain unauthorized access to your database. By using placeholders instead of directly including user input into the SQL query, you can significantly reduce the risk of SQL Injection attacks. The provided example demonstrates how a query can be made secure using prepared statements, which separates SQL syntax from data and handles user input safely.

Sample Code

```
// Example of vulnerable query
 $username = $_POST['username'];
 $password = $_POST['password'];
 $query = "SELECT * FROM users WHERE username='$username' AND
password='$password'";

 // Example of secure query using prepared statements
 $username = $_POST['username'];
 $password = $_POST['password'];
 $stmt = $pdo->prepare("SELECT * FROM users WHERE username=? AND
password=?");
 $stmt->execute([$username, $password]);
 $user = $stmt->fetch();
```

Proper Session Management: Proper session management ensures that authenticated users have timely and appropriate access to resources, thereby reducing the risk of unauthorized access to sensitive information. Session management includes using secure cookies, setting session timeouts, and limiting the number of active sessions a user can have. The code snippet shows how to initialize a session, set session variables and check for session validity by looking at the last activity time.

Sample Code

```php
// Start session
session_start();

// Set session variables
$_SESSION['user_id'] = $user_id;
$_SESSION['last_activity'] = time();

// Check if session is still valid
if (isset($_SESSION['last_activity']) && (time() -
$_SESSION['last_activity'] > 1800)) {
    // Session has expired
    session_unset();
    session_destroy();
}
```

Use Secure Coding Practices: Secure coding practices involve methods to prevent the introduction of security vulnerabilities. Developers should sanitize and validate user input to prevent malicious data from causing harm and avoid using insecure functions or libraries. The given example shows how to sanitize user input using PHP's `filter_input` function and demonstrates how to securely hash a password using `password_hash` instead of an insecure function like `md5` .

Sample Code

```
// Validate user input
 $username = filter_input(INPUT_POST, 'username',
FILTER_SANITIZE_STRING);
 $password = filter_input(INPUT_POST, 'password',
FILTER_SANITIZE_STRING);
```


```
// Example of secure code using password_hash
$password = password_hash($password, PASSWORD_DEFAULT);
```

Conclusion

Broken access control is a security vulnerability that occurs when a system fails to properly enforce access controls, which can result in unauthorized users gaining access to sensitive information or performing actions they are not authorized to do.

Horizontal privilege escalation occurs when a user is able to access data or perform actions that they are not authorized to do within their own privilege level. This can be dangerous because it can allow an attacker who has already gained access to the system to move laterally through the network and access additional resources or sensitive data.

Vertical privilege escalation occurs when a user is able to gain access to data or perform actions that are reserved for users with higher privilege levels, such as system administrators. This can be even more dangerous because it can allow an attacker to gain full control of the system and potentially take over the entire network.

The impact of these types of privilege escalation can vary depending on the specific system and the level of access that is gained. However, in general, the consequences can include unauthorized access to sensitive information, data loss or theft, disruption of critical systems or services, and even complete network compromise. Therefore, it is important to implement strong access controls and regularly monitor for any signs of unauthorized access or activity.

Thank Your for reading !

Happy Hacking !

Author: Sudarshan Patel

Tryhackme | Tryhackme Walkthrough | Tryhackme Writeup

Follow

Written by Sudarshan Patel

81 Followers · 140 Following

Cyber Security Researcher && Bug Bounty Hunter

No responses yet

What are your thoughts?

Respond

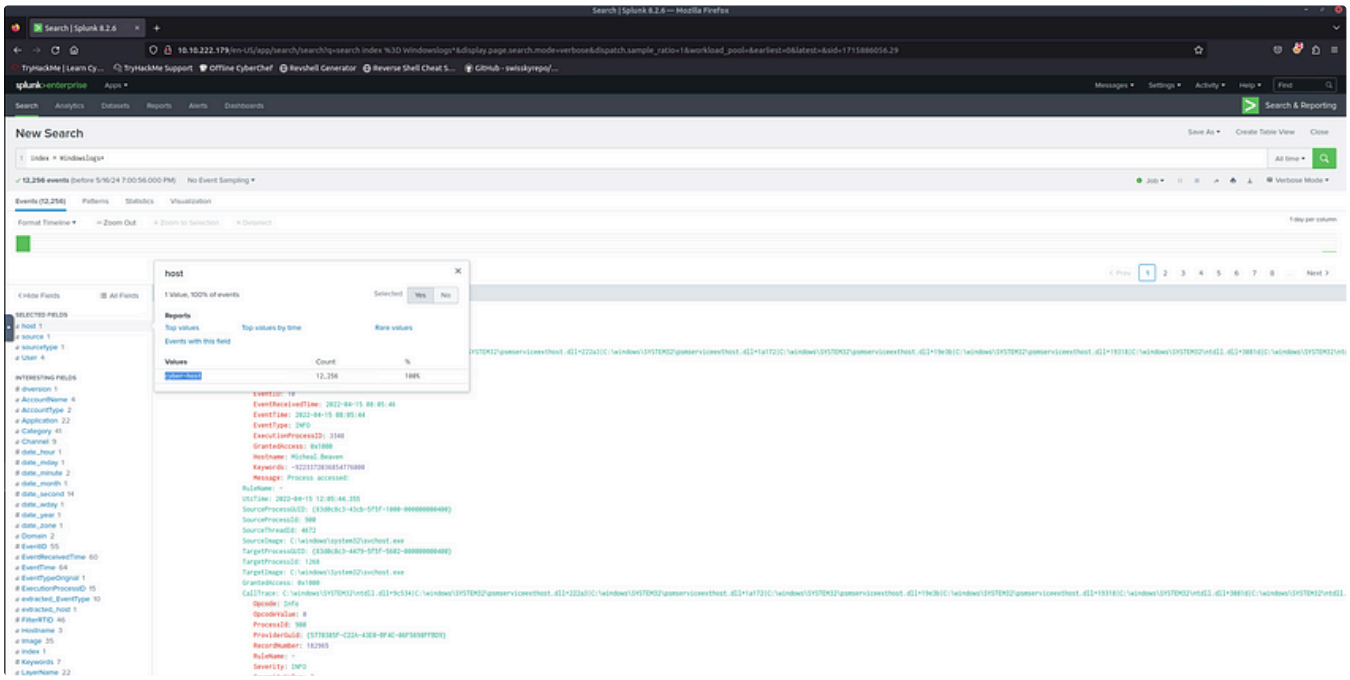
More from Sudarshan Patel

Sudarshan Patel

📌 👉Tryhackme | Advent of Cyber—2024 | Day 15 | Active Directory | Be it ever so heinous...

The Story

Dec 26, 2024



Sudarshan Patel

🖥️ 📊 Tryhackme | Splunk: Exploring SPL | Walkthrough 📈 📌

Learn and explore the basics of the Search Processing Language.

May 17, 2024 4 1

Sudarshan Patel

Tryhackme | OWASP Top 10–2021 | X—Server-Side Request Forgery (SSRF)

Ladies and gentlemen, a warm welcome to Day 10 of our exploration through the OWASP Top 10 2021! Your esteemed presence adds brilliance to...

Jan 4, 2024 10 1

Sudarshan Patel

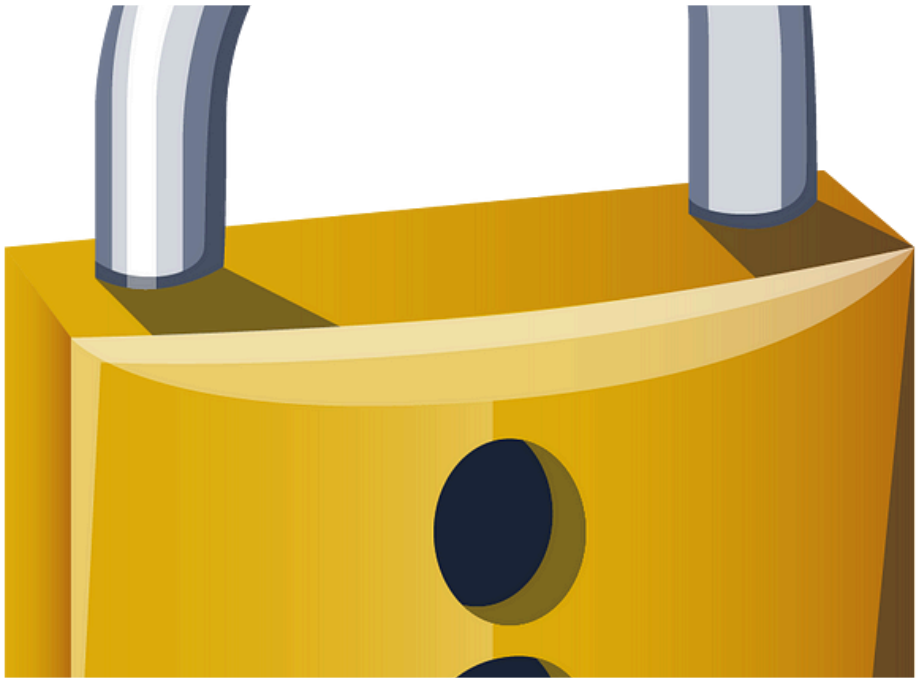
🚀 📌 Tryhackme | Advent Of Cyber — 2024 | Day 1: Maybe SOC-mas music, he thought, doesn't come from...

McSkidy tapped keys with a confident grin,

Dec 2, 2024 10

See all from Sudarshan Patel

Recommended from Medium



Huy Phu

TryHackMeOAuth Vulnerabilities

URL

Nov 10, 2024

Trnty

TryHackMe | Introduction To Honeypots Walkthrough

A guided room covering the deployment of honeypots and analysis of botnet activities

Sep 7, 2024 10

Lists

Staff picks

796 stories · 1558 saves

Stories to Help You Level-Up at Work

19 stories · 912 saves

Self-Improvement 101

20 stories · 3191 saves

Productivity 101

20 stories · 2704 saves

Huy Phu

TryHackMe—Attacking and Defending AWS: AWS Cloud 101

Introduction

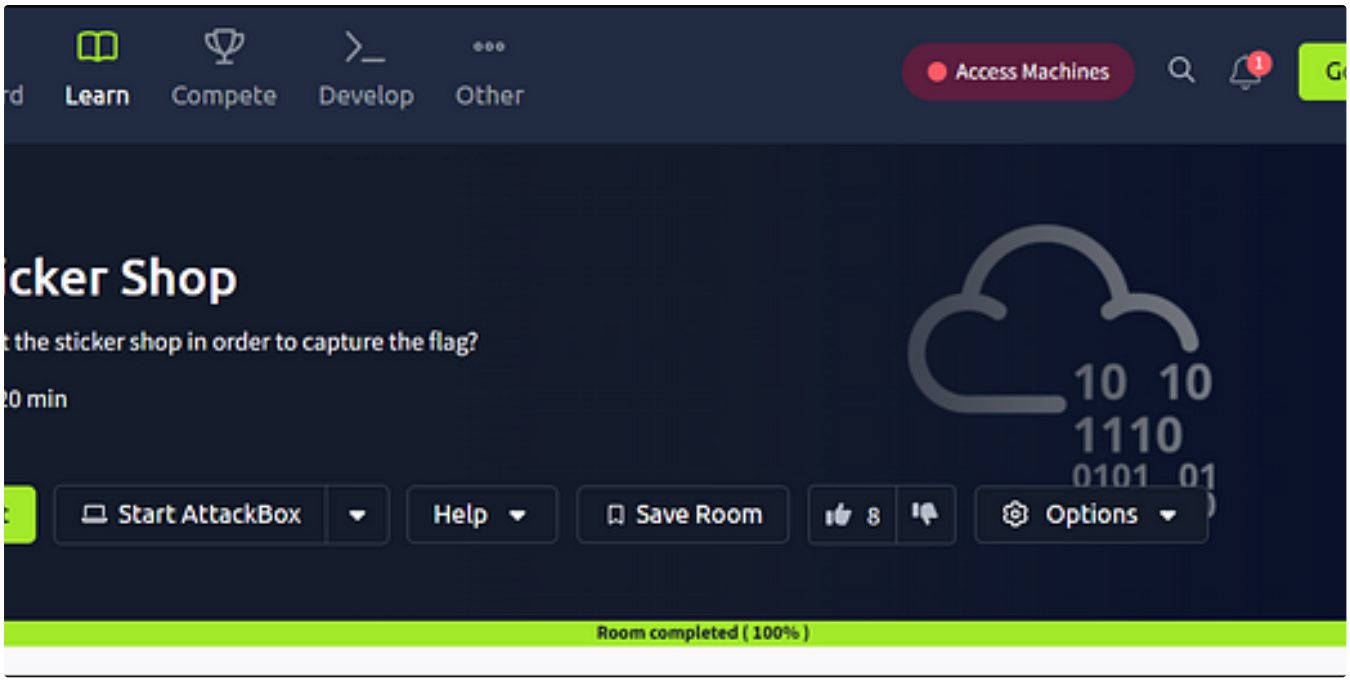
Aug 8, 2024 2 1

Shivamsharma

Broken Access Control

OWASP TOP 10 > Broken Access Control

Aug 23, 2024 16

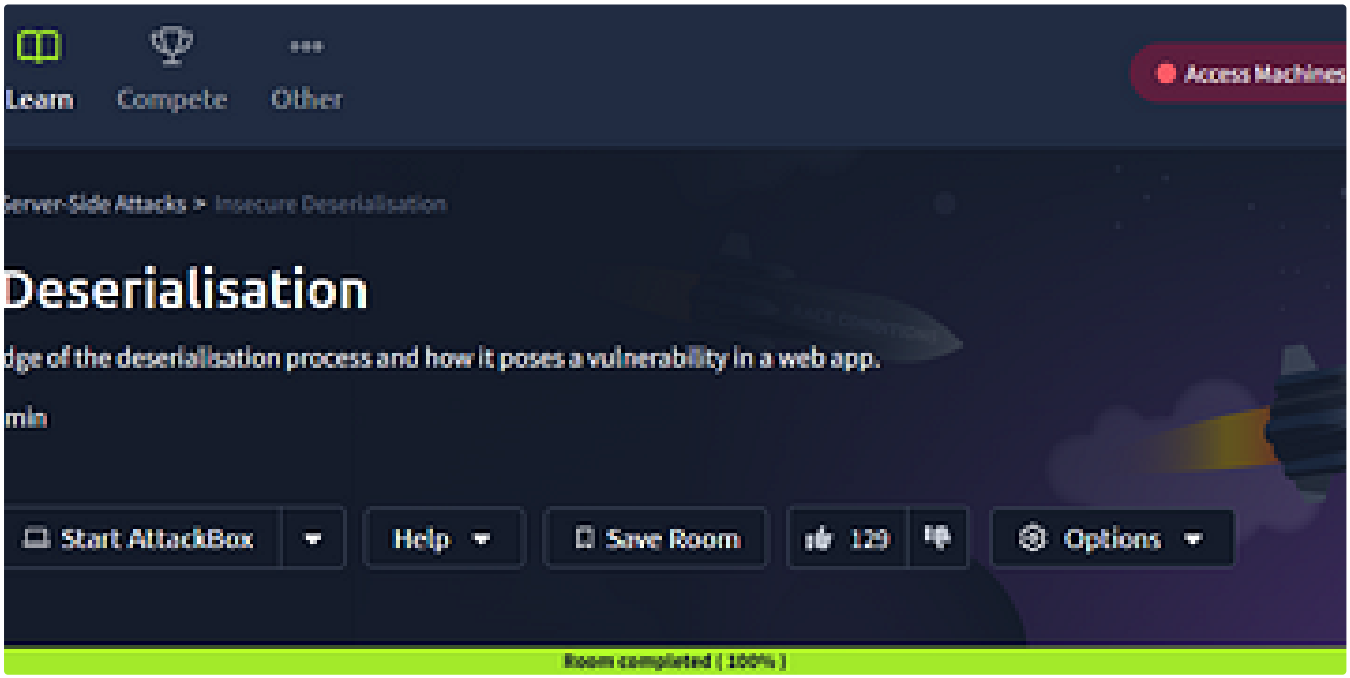


Shakhawat Hossain - 0xShakhawat

The Sticker Shop | TryHackMe | Walkthrough

How I Solved The Sticker Shop CTF: Exploiting Blind XSS to Capture the Flag. This writeup walks you through the steps of exploiting a Blind...

Nov 30, 2024 112 4



RosanaFSS

TryHackMe: Insecure Deserialisation

Web Application Pentesting learning path > Advanced Server-Side Attacks > Insecure Deserialisation: Get in-depth knowledge of the...

Nov 21, 2024 77