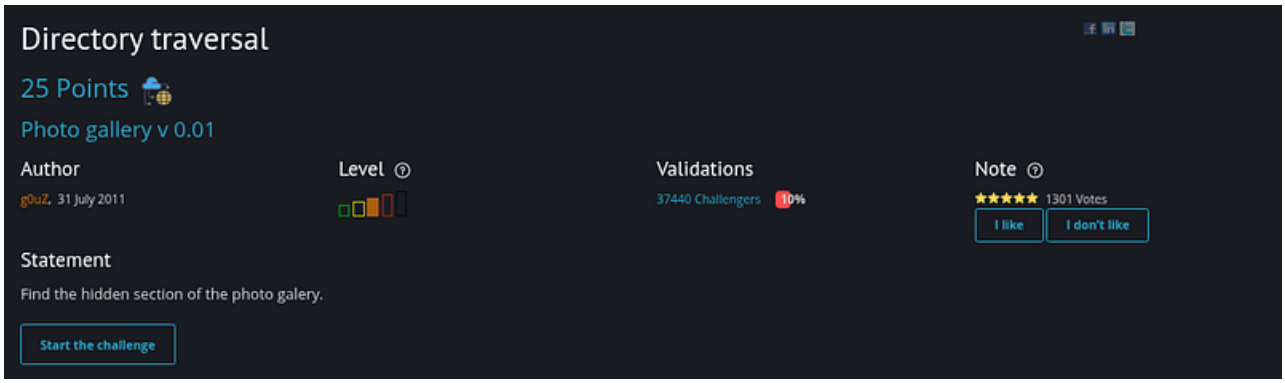
# Root-Me Write-up: Directory Traversal




## Initial Assessment

The challenge appeared to involve a **path traversal vulnerability**, which occurs when user-controlled input is not properly sanitized, allowing an attacker to access files or directories outside the intended scope by manipulating path parameters (e.g., ../).

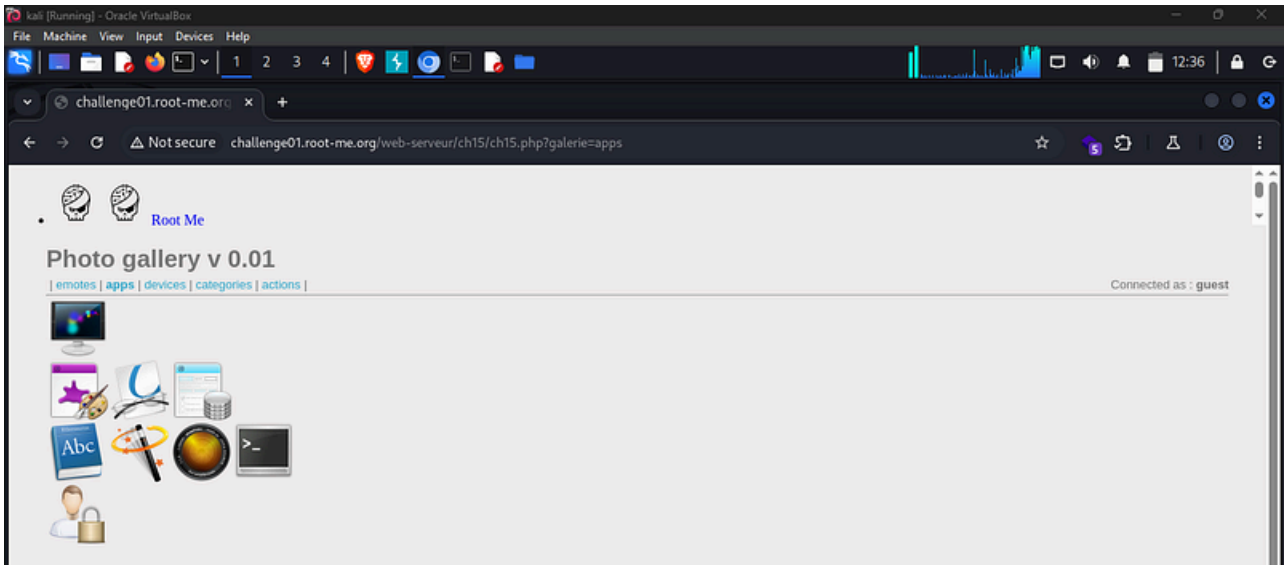
I began with basic test cases such as ../ and other typical traversal payloads to observe how the application handled directory navigation.

## Enumeration

Next, I attempted directory enumeration using **Gobuster**, targeting commonly known directories. While this provided general context, the more useful insight came from manually inspecting the application.

Upon interacting with the gallery feature, I noticed that selecting different categories triggered requests of the following format:

/ch15.php?galerie=<category>




Based on the challenge hints, it was suggested that something important was hidden within the galerie parameter.

# Exploitation

I tested the following payload:

/ch15.php?galerie=/

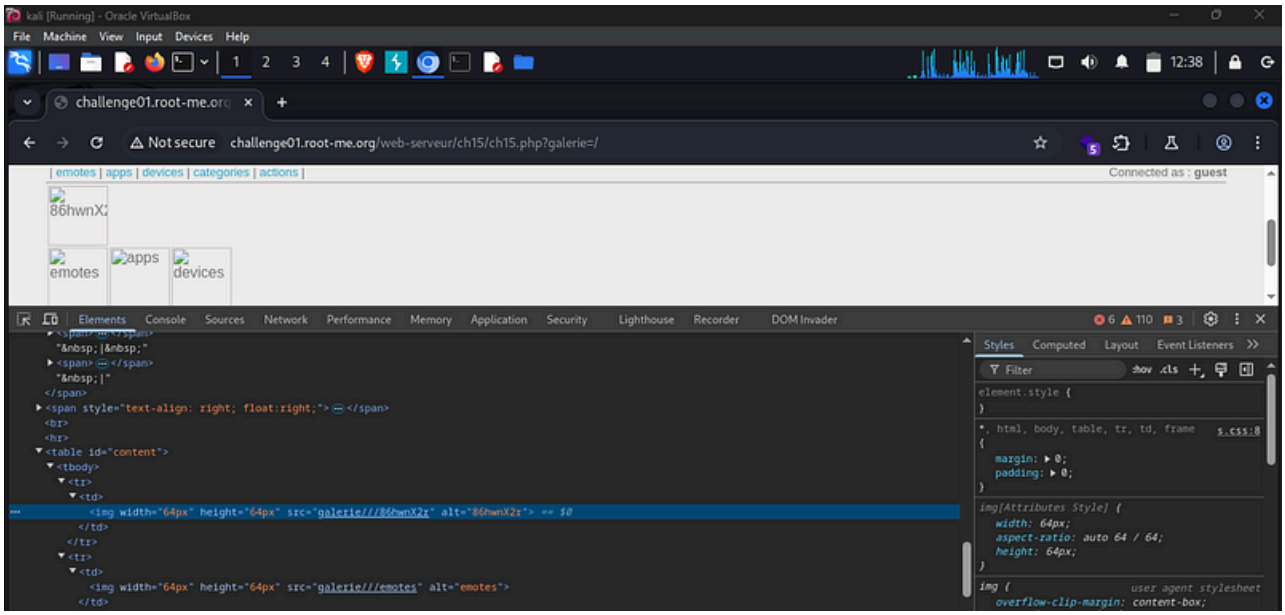
This effectively forced the application to reference another directory. As a result, a uniquely named image file was revealed:

86hwnX2r

The HTML source showed the following element:

<img width="64px" height="64px" src="galerie///86hwnX2r" alt="86hwnX2r">

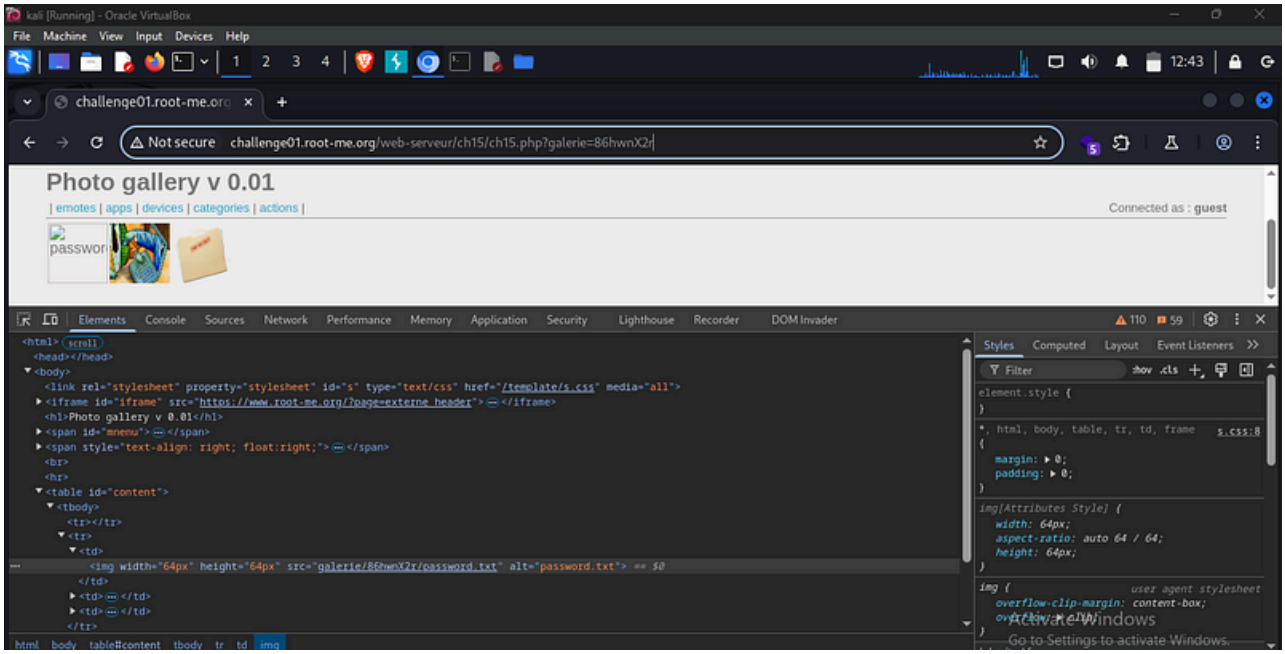
Using the browser's inspector, I examined the full path and then attempted to access it directly:




/ch15.php?galerie=86hwnX2r

This led to another discovery. The page referenced a text file:

<img width="64px" height="64px" src="galerie/86hwnX2r/password.txt" alt="password.txt">

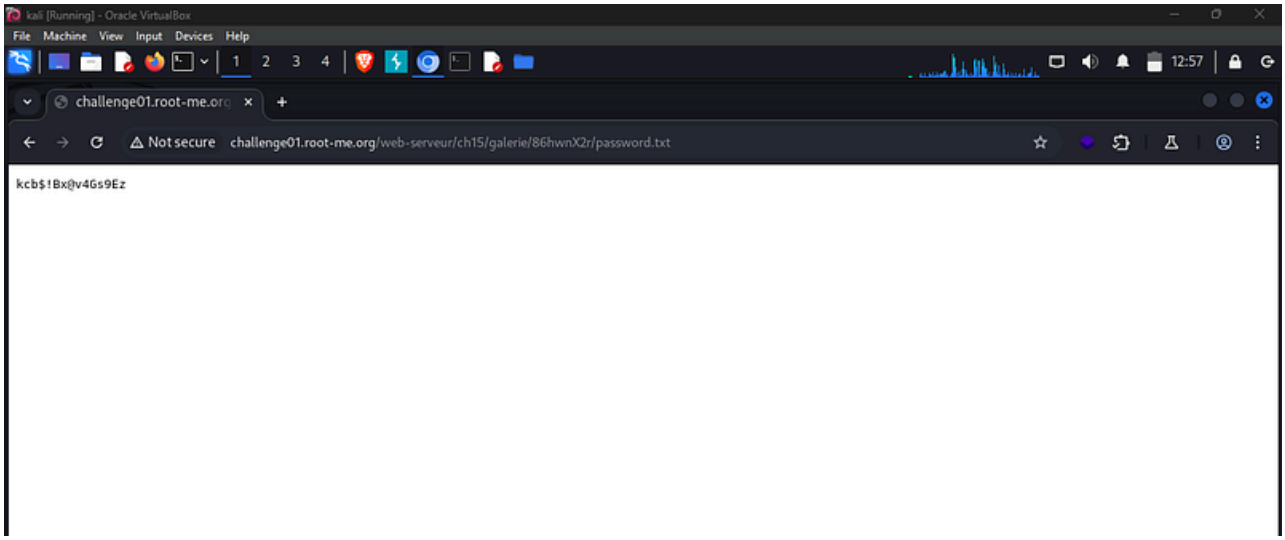



# Flag Retrieval

From the src attribute, the full file path could be inferred. I accessed it directly via:

/ch5/galerie/86hwnX2r/password.txt

This successfully revealed the flag.




# Flag

kcb$!Bx@v4Gs9Ez

By Alexander Sapo on December 30, 2025.

Canonical link

Exported from Medium on February 7, 2026.