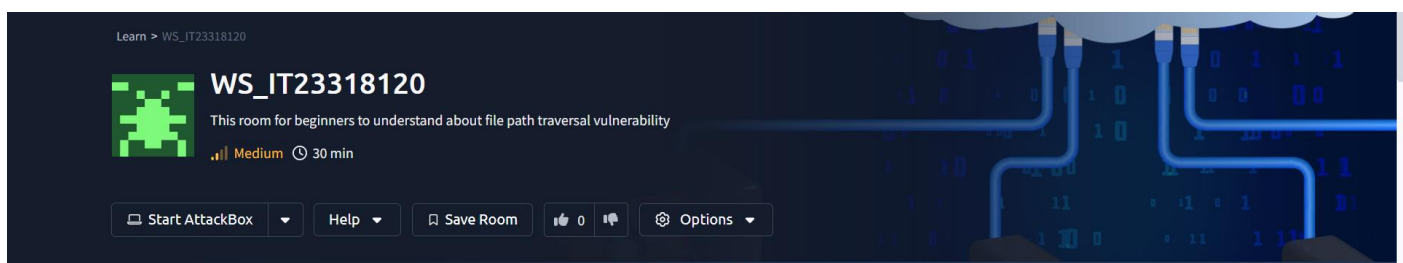


# Sri Lanka Institute of Information Technology



**Specialized in Cyber Security  
Year 2, Semester 2**

## **IE2062 – Web Security Report about TryHackMe Room**



**Sudent ID :** IT23318120  
**Name :** T.G.U Kaushalya

## Contents

About room.....	2
Introduction section .....	3
Task 01 .....	4
Task 02 .....	5
Task 03 .....	5
Task 04 .....	6
Task 05 .....	7
Task 06 .....	8
Task 07 .....	8
Conclusion .....	10

**About room****Title:** WS\_IT23318120**Description:**

This room provides learners with a hands-on introduction to path traversal attacks, a common web vulnerability that allows unauthorized access to sensitive files and directories by manipulating file paths. The room blends theoretical questions with practical VM-based exploitation to simulate real-world scenarios.

**Learning Objectives:**

By the end of this room, participants will:

- Identify vulnerable file input points in web applications.
- Understand how to construct path traversal payloads.
- Exploit basic and advanced path traversal vulnerabilities.
- Understand extension filtering and bypass techniques.
- Practice exploiting the vulnerability using CLI tools like curl.
- Implement mitigation strategies to defend against path traversal.

**Room code:** wsit23318120mH**Completion time:** 30 minutes**Difficulty:** medium**Type:** challenge

### Introduction section

This initial task introduces participants to the concept of Path Traversal, a vulnerability that occurs when user-controlled input is used to construct file paths without proper validation or sanitization.

Learners explore how attackers manipulate parameters such as file=, page=, or doc= in URLs to traverse the file system using patterns like ../, allowing access to restricted or sensitive files outside the intended directory.

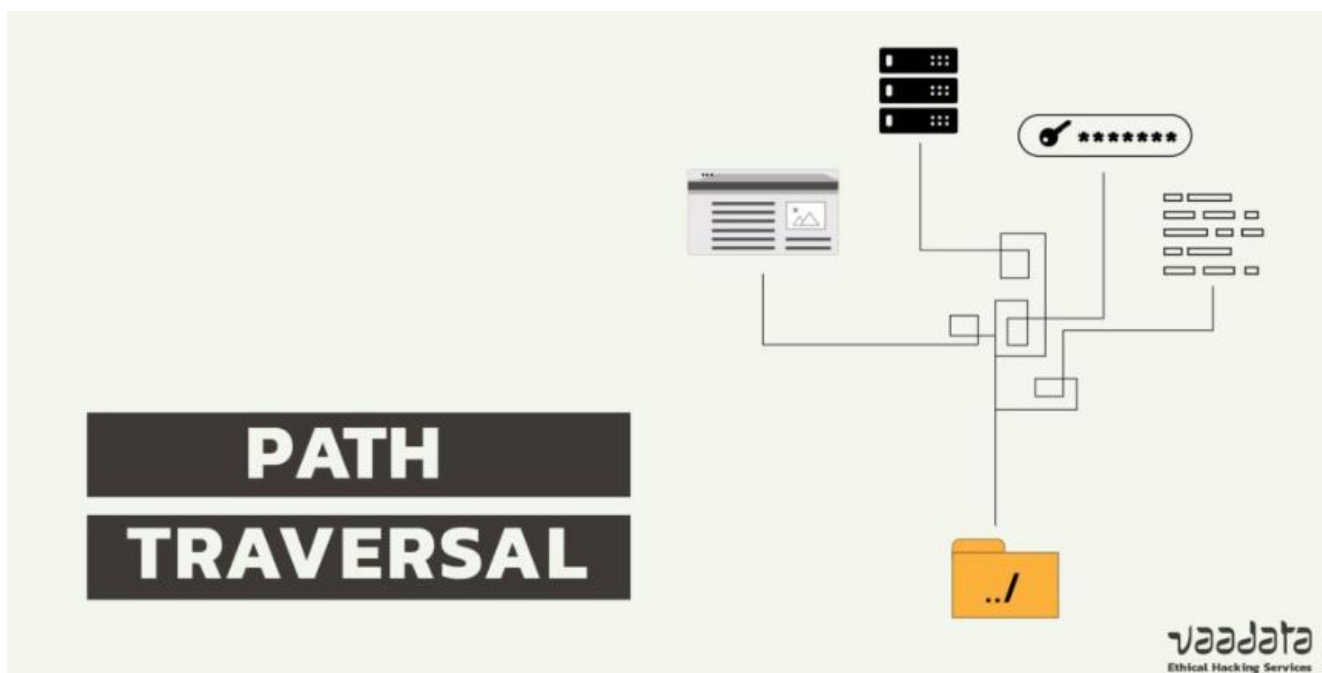
The objective is to:

- Identify potential path traversal points in URLs.
- Understand the danger of unsanitized file path parameters.
- Grasp the basic exploitation technique using real examples like:

<http://<target-ip>/view.php?file=report.txt>

### Questions will cover following factors

- Definition of Path Traversal: Recognizing it as a method for unauthorized file access.
- Exploitation Method: Using relative path sequences (e.g., ../) to move around the server's directory structure.
- Entry Point Indicators: Understanding that parameters like ?file= are common signs of potential vulnerabilities.

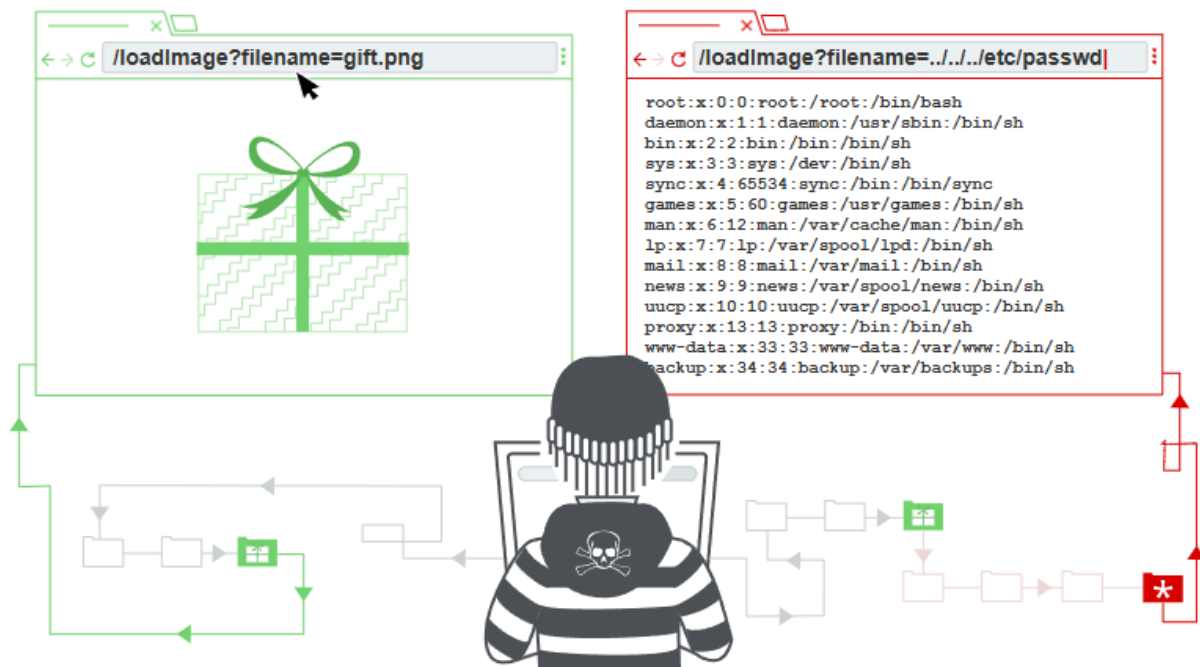


### Task 01

This section teaches learners how attackers use the `../` (dot-dot-slash) sequence in file paths to navigate upward in the directory structure. Many web applications allow file access via user-supplied parameters. If these inputs aren't properly sanitized, an attacker can manipulate them to escape the intended folder and access sensitive files on the server.

#### Learning Objective:

Understand the functional purpose of `../` and how it's abused to bypass directory restrictions in insecure applications.

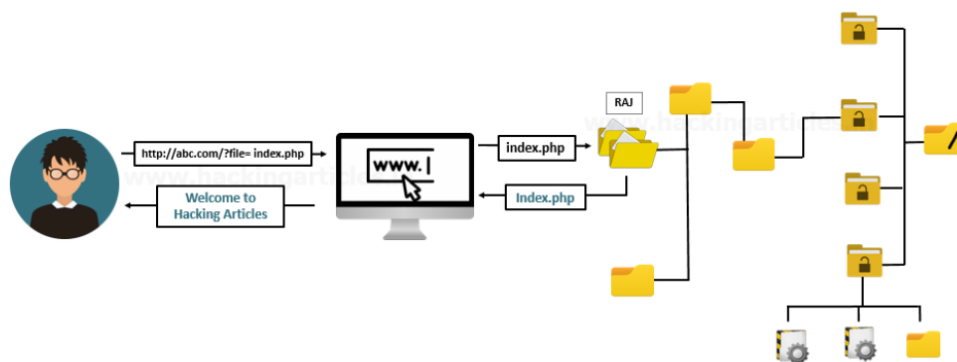


### Task 02

Some vulnerable applications automatically append a file extension (like .php) to user-supplied input. This is often done to restrict access only to certain file types. However, attackers can exploit this behavior using special characters like %00 (null byte) or by URL encoding traversal sequences to bypass such controls. The null byte (%00) is used to terminate strings early, so the server ignores the forced .php suffix. Similarly, attackers may URL-encode characters (e.g., %2F instead of /) to bypass weak filters or WAF rules.

#### Learning Objective:

Recognize techniques used to bypass extension restrictions and understand how null byte injection works in legacy or poorly configured servers.



### Task 03

In many real-world scenarios, a web application may be nested within multiple directories. If the attacker doesn't know the exact location of the target file (like /etc/passwd), they must **brute-force** the traversal depth by gradually adding more ../ sequences.

This trial-and-error approach helps the attacker determine how many directory levels they need to move up to escape the current directory and reach the desired file path.

#### Example :

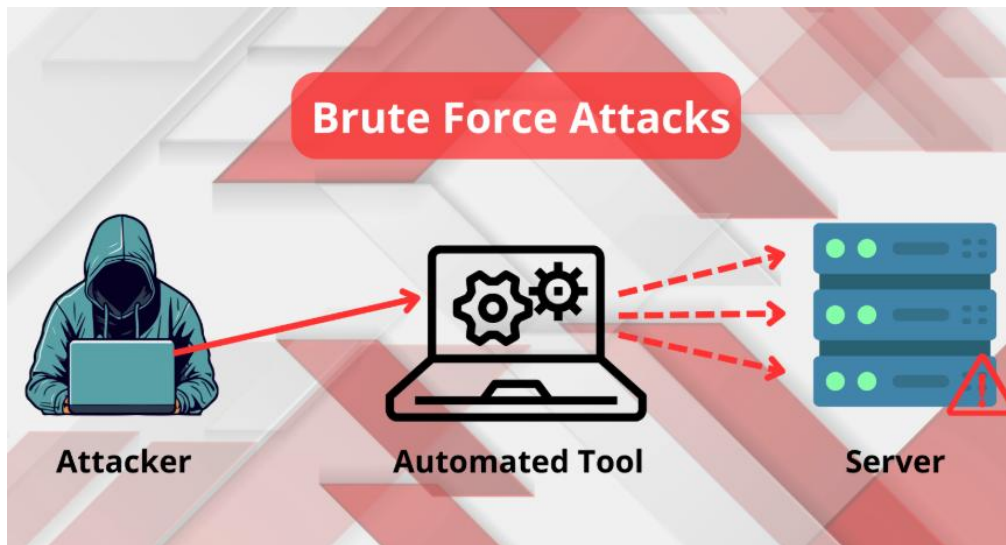
file=../etc/passwd

file=../../etc/passwd

file=../../../etc/passwd

#### Learning Objective

Understand why traversal depth matters and how attackers use it to reach protected directories when directory structure is unknown.



#### Task 04

A common cause of path traversal vulnerabilities is when developers directly include user-supplied input in file paths without validation. In PHP, functions like `file_get_contents()` or `include()` become dangerous when fed unsanitized input from `$_GET`, `$_POST`, or other request variables.

```
GNU nano 7.2          view.php
<?php
$file = $_GET['file'];
$content = file_get_contents($file);
echo "<pre>$content</pre>";
?>
```

In this example:

- The `$file` variable comes directly from the URL.
- There's no validation, sanitization, or restriction of the input.
- An attacker can exploit this to read arbitrary files from the system.

When this logic is extended with `include("pages/" . $file);`, the risk increases, especially if the attacker can control the filename or inject traversal sequences; potentially allowing execution of arbitrary PHP files.

## Task 05

To effectively apply the concepts of File Path Traversal, learners are encouraged to interact with a real vulnerable environment. This is made possible through a virtual machine (VM) that simulates the conditions needed for practical exploitation.

### Deployment Instructions:

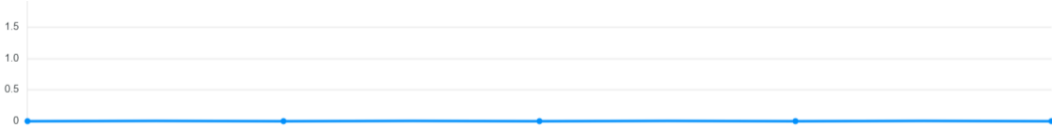
1. Click the "Start Machine" button at the top-right corner of this task to deploy the VM.
2. Once the machine is running, a dynamic IP address will be assigned and shown in the interface.
3. To begin attacking the machine, click "Start AttackBox" above the room name, this launches a browser-based Kali Linux environment directly connected to the target VM.

This setup enables safe and legal practice of path traversal attacks in a controlled TryHackMe environment.

**Task 6** ○ Task 05

To focus on learning about the File path traversal Vulnerability, please click on the **Start Machine** button located in the upper right hand corner of this task to deploy the virtual machine for this room.

After obtaining the machine's generated IP address, you can use our AttackBox by simply click on the **Start AttackBox** button located above the room name.



Target Machine Information		
<b>Title</b>	<b>Target IP Address</b>	<b>Expires</b>
VM latest	Shown in 0min 35s ⓘ	59min 33s

?
Add 1 hour
Terminate

**Task 1** ○ Introduction

**Task 2** ○ Task 01

**Task 3** ○ Task 02



**Task 06**

After launching the AttackBox and starting the target VM, learners are instructed to interact with the vulnerable web application through a browser.

**Objective:**

Use the browser (Firefox inside the AttackBox) to access:

<http://<generated IP>/view.php?file=notes.txt>

Then, apply the brute-force traversal technique (as explained in Task 03) to incrementally try deeper traversal levels and reveal hidden files like `/etc/passwd`.

By chaining enough `../` sequences, you will reach and expose sensitive server content that should be inaccessible from the browser — demonstrating the impact of improper input handling.





**Task 07**

Path traversal attacks can be effectively prevented by implementing a combination of secure coding practices and defensive mechanisms. Understanding these mitigation strategies is essential for building resilient web applications.

Top 3 Security Best Practices:

**1. Input Validation & Sanitization**

- Rigorously validate user input.
- Strip or reject suspicious sequences such as `../`, `%2e%2e/`, or null bytes (`\0`).
- Only allow expected, safe characters in file names.

Input Sanitization		Input Validation
Removes unsafe characters or encodes them to prevent harmful input		Checks if input meets specific criteria
Modifies the input to ensure safety before being processed or stored in a database		Ensures data complies with the expected format, like valid email addresses or numeric fields
Helps block malicious code to protect against XSS (Cross-Site Scripting) and SQL Injection		Helps prevent unexpected or erroneous data from being processed by verifying its type and integrity before use
Should be done before storing data in a database or displaying data to maintain security		Typically done before processing or saving input to make sure it adheres to rules and constraints

## 2. Whitelisting Allowed Files or Paths

- Create and enforce a whitelist of known-safe file names or directories.
- Block any input that doesn't match predefined, trusted entries.



## 3. Use Secure APIs & File Access Controls

- Avoid concatenating user input directly into file paths.
- Prefer APIs that resolve absolute paths safely.
- Implement server-level constraints like chroot jails, sandboxes, or file permission restrictions.

### Learning Objective

Understand the core defensive strategies for preventing path traversal and why whitelisting is particularly effective.

## Conclusion

This room effectively bridges **theory and practice**, providing cybersecurity learners with:

- Real-world examples of path traversal
- A safe, structured environment for attack simulation
- Reinforced learning through contextual MCQs

It serves as an ideal module for students, CTF learners, or junior pen testers seeking foundational web exploitation knowledge.



# THANK YOU 😊