Directory Traversal

08/08/2022

Aug 7, 2022

4 min read
Dir-Trav Web-Notes

Directory traversal

In this section, we'll explain what directory traversal is, describe how to carry out path traversal attacks and circumvent common obstacles, and spell out how to prevent path traversal vulnerabilities.

References

File Inclusion/Path traversal

Directory / Path Traversal

PayloadsAllTheThings/Directory Traversal at master · swisskyrepo/PayloadsAllTheThings

GitHub - vavkamil/awesome-bugbounty-tools: A curated list of various bug bounty tools

Laravel - OWASP Cheat Sheet Series

pentest-guide/Directory-Traversal-File-Include at master · Voorivex/pentest-guide

Writeups Bug Bounty hackerone

What is directory traversal?

Directory traversal (also known as file path traversal) is a web security vulnerability that allows an attacker to read arbitrary files on the server that is running an application. This might include application code and data, credentials for back-end systems, and sensitive operating system files. In some cases, an attacker might be able to write to arbitrary files on the server, allowing them to modify application data or behavior, and ultimately take full control of the server.

Reading arbitrary files via directory traversal

Images are loaded via some HTML like the following:

```
<img src="/loadImage?filename=218.png">
```

The `loadImage`URL takes a `filename`parameter and returns the contents of the specified file. The image files themselves are stored on disk in the location `/var/www/images/`

In the above case, the application reads from the following file path: `/var/www/images/218.png`

The application implements no defenses against directory traversal attacks, so an attacker can request the following URL to retrieve an arbitrary file from the server's filesystem:

`[https://insecure-website.com/loadImage?filename=../../../etc/passwd](https://insecure-website.com/loadImage?filename=../../../etc/passwd)` This causes the application to read from the following file path: `/var/www/images/../../../etc/passwd`

this make the attacker to read the passwd directory file

On Windows, both `../`and `..\`are valid directory traversal sequences, and an equivalent attack to retrieve a standard operating system file would be:

`https://insecure-website.com/loadImage?filename=..\..\..\windows\win.ini`

Lab: File path traversal, simple case Web Security Academy

Common Blocks to exploiting file path traversal vulnerabilities

If an application strips or blocks directory traversal sequences from the user-supplied filename, then it might be possible to bypass the defense using a variety of techniques.

You might be able to use an absolute path from the filesystem root, such as `filename=/etc/passwd`, to directly reference a file without using any traversal sequences.

Lab: File path traversal, traversal sequences blocked with absolute path bypass Web Security Academy

You might be able to use nested traversal sequences, such as `....//` or `....\/`, which will revert to simple traversal sequences when the inner sequence is stripped.

Lab: File path traversal, traversal sequences stripped non-recursively Web Security Academy

In some contexts, such as in a URL path or the `filename` parameter of a `multipart/form-data` request, web servers may strip any directory traversal sequences before passing your input to the application. You can sometimes bypass this kind of sanitization by URL encoding, or even double URL encoding, the `../` characters, resulting in `%2e%2e%2f` or `%252e%252e%252f` respectively. Various non-standard encodings, such as `..%c0%af` or `..%ef%bc%8f`, may also do the trick.

Lab: File path traversal, traversal sequences stripped with superfluous URL-decode Web Security Academy

If an application requires that the user-supplied filename must start with the expected base folder, such as `/var/www/images`, then it might be possible to include the required base folder followed by suitable

traversal sequences. For example:

```
filename=/var/www/images/../../../etc/passwd
```

Lab: File path traversal, validation of start of path Web Security Academy

If an application requires that the user-supplied filename must end with an expected file extension, such as `.png`, then it might be possible to use a null byte to effectively terminate the file path before the required extension. For example:

```
filename=../../../etc/passwd%00.png
```

Lab: File path traversal, validation of file extension with null byte bypass Web Security Academy

---

How to prevent a directory traversal attack

If it is considered unavoidable to pass user-supplied input to filesystem APIs, then two layers of defense should be used together to prevent attacks:

- The application should validate the user input before processing it. Ideally, the validation should compare against a whitelist of permitted values.

  If that isn't possible for the required functionality, then the validation should verify that the input contains only permitted content, such as purely alphanumeric characters.

- After validating the supplied input, the application should append the input to the base directory and use a platform filesystem API to canonicalize the path.

  It should verify that the canonicalized path starts with the expected base directory.

Below is an example of some simple Java code to validate the canonical path of a file based on user input:

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