Path Traversal

> Vulnerability that allows an attacker to read arbitrary files on the server that is running an application.

Core Concepts

- Also known as **Directory Traversal**.
- Exploits insufficient security validation / sanitization of user-supplied input files names.
- Uses the **dot-dot-slash** (`../`) sequence to move up the directory tree.
- **Goal:** Access sensitive files outside the web root folder.
  - Application code and data.
  - Credentials for backend systems.
  - Sensitive operating system files.

Common Targets

- **Linux/Unix:**
  - `/etc/passwd` (User information)
  - `/etc/hosts` (Network mapping)
- **Windows:**
  - `C:\Windows\win.ini` (System configuration)
  - `C:\windows\system32\drivers\etc\hosts`

Types of Path Traversal Attacks

Simple Case (Relative Path)

- The application takes the filename and appends it to a directory path.
- The attacker injects `../` to step out of the intended folder and access the root. Example Payload:

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

Absolute Path Traversal

- Some applications allow the user to specify the full path from the filesystem root, bypassing the need for `../`. Example Payload:

```
/etc/passwd
```

HOW TO FIND PATH TRAVERSAL

Black-Box Testing Perspective

- **Map the application:** Identify all instances where the application appears to be retrieving data based on a filename or path parameter.
  - Query parameters: `?file=report.pdf`, `?image=design.png`, `?load=home`
  - API endpoints downloading files.
- **Fuzz the application:**
  - Submit known path traversal sequences (`../`, `..\`)
  - Submit absolute paths.
  - Observe error messages (e.g., "File not found" vs "Permission denied").

HOW TO EXPLOIT PATH TRAVERSAL (BYPASS TECHNIQUES)

If simple traversal is blocked, developers often implement weak filters that can be bypassed.

1. Traversal Sequences Stripped (Non-Recursive)

- The application removes `../` patterns, but only once.
- **Bypass:** Nested traversal sequences.
- Logic: `....//` becomes `../` after the filter strips the inner `../`. Example Payload:

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

2. URL Encoding / Double Encoding

- The web server may decode inputs, but the application filter might check the encoded version, or vice-versa.
- **Bypass:** Encode the `/` or `.` characters.

- URL Encoded: `%2e%2e%2f`
- Double URL Encoded: `%252e%252e%252f` Example Payload:

```
..%252f..%252f..%252fetc/passwd
```

3. Validation of Start of Path

- The application checks if the path starts with a specific folder (e.g., `/var/www/images`).
- **Bypass:** Include the required folder, then traverse back out. Example Payload:

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

4. Null Byte Injection (Legacy/Specific Languages)

- Some applications (C/C++, older PHP) treat `%00` as the end of a string.
- Used when the application forces an extension (e.g., appends `.jpg` to your input).
- **Bypass:** End the filename with `%00`. Example Payload:

```
../../../etc/passwd%00.jpg
```

(The system reads up to passwd, ignoring the .jpg)

Automated Exploitation Tools

- **DotDotPwn:** A fuzzer specifically for discovering traversal vulnerabilities.
- **Burp Suite:**
  - Scanner (Passive/Active)
  - Intruder (Fuzzing lists)
- **WFuzz:** Web application fuzzer using traversal wordlists.