

# PortSwigger

## 1) What is path traversal?

- Also called directory traversal. It lets an attacker cause the application to read (and sometimes write) arbitrary files on the server by manipulating file-path input. Typical targets: app code/data, credentials, and OS files.
- Risk escalation: reading sensitive files can leak secrets; writing files (if possible) can lead to tampering or full server takeover.

## 2) Reading arbitrary files via path traversal (example + mechanism)

- Typical vulnerable pattern: the app concatenates a user-provided filename to a base directory and passes the result to a filesystem API. Example: `/var/www/images/ + filename` → File read.
- Simple exploit: use `../` (on Unix) or `..\` (on Windows) to climb directories. Example request: `...?filename=../../etc/passwd` → the server may end up reading `/etc/passwd`.
- Absolute path trick: if traversal sequences are blocked, supplying an absolute path (e.g. `/etc/passwd`) may bypass defences that only look for `..` sequences.

## 3) Common obstacles and common bypass techniques

- PortSwigger lists many defences that apps implement and the usual bypasses to try:
- Defences that appear in the wild:
- Stripping or blocking `../` sequences.
- Requiring the filename to start with a specific base folder.
- Requiring the filename to end in a particular extension.
- Web server or framework-level sanitization (e.g., stripping in URL path or multipart filename).
- Bypasses and techniques to try:
- Nested / crafted traversal: `....//` or `....\` — if the app strips `..` but leaves `....//` the inner `..` may remain when processed.
- URL encoding / double-encoding: encode `../` as `%2e%2e%2f` or double-encode to `%252e%252e%252f` to defeat naive filters. Try non-standard encodings too (e.g. `..%c0%af`, `..%ef%bc%8f`).
- Include required base path in input: if the app forces the filename to start with `/var/www/images`, send `filename=/var/www/images/../../etc/passwd`.
- Null-byte termination (legacy / misconfigured): if the app appends a required extension (like `.png`), try `../../etc/passwd%00.png` to terminate the path early in languages/environments where a `%00` does terminate the string (note: largely mitigated in modern runtimes, but still worth checking where legacy behaviour exists).
- Try absolute paths: submit `/etc/passwd` or `C:\Windows\win.ini` when relative traversal is blocked.
- (PortSwigger also mentions Burp Intruder has a built-in payload list for path traversal fuzzing — useful for pen test automation.)

## 4) How to prevent a path traversal attack (recommended defences/pattern)

- PortSwigger recommends two layers of defence and some safer design choices:
- High-level principle:
- Avoid passing raw user input to filesystem APIs. Rework app flow so user-controlled values are not used directly to construct file paths (for example, use an ID that maps server-side to a filename, or serve files through an index).
- If you must accept filenames from users, use a defence-in-depth approach:
- Input validation (whitelist preferred): Compare user input against a whitelist of allowed values (best). If a whitelist is not possible, restrict to a narrow, safe character set (e.g., only alphanumeric characters, hyphens, underscores).
- Canonicalize and verify the resolved path: After validation, append the user input to the base directory and canonicalize the result (resolve symlinks, etc.) using platform filesystem APIs, then verify the canonical path starts with the expected base directory before using it. Example Java snippet from PortSwigger: