[Th 2] Information Disclosure and Directory traversal

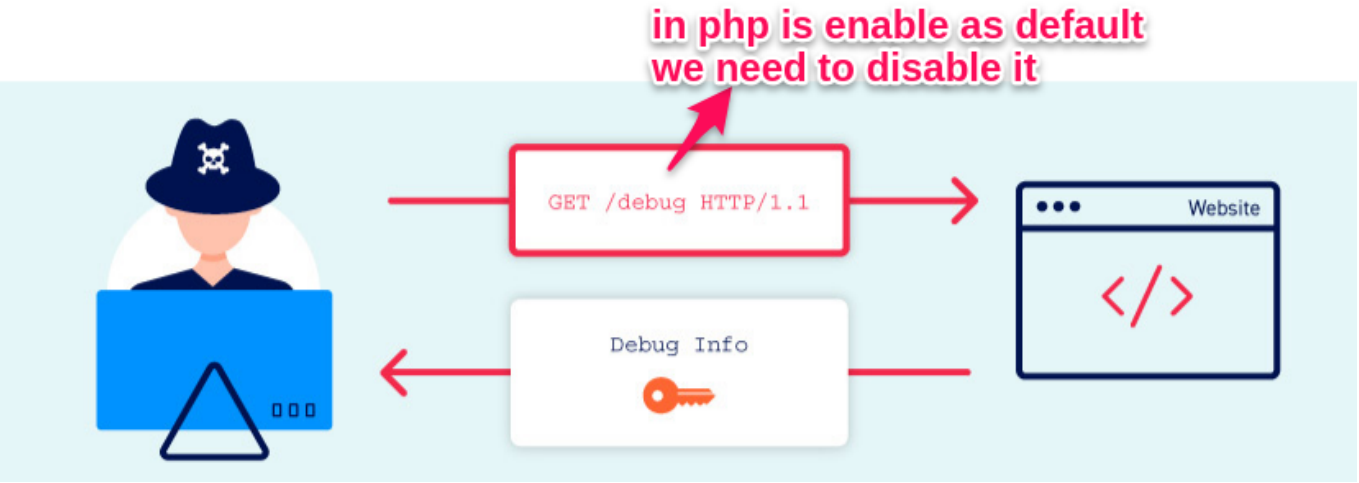
INFORMATION DISCLOSURE

Information disclosure, also known as information leakage, is when a website unintentionally reveals sensitive information to its users.

Depending on the context, websites may leak all kinds of information to a potential attacker, including:

- data about other users, such as usernames of financial information like card numbers
- sensitive commercial or business data
- technical details about the website and its infrastructure

More commonly, however, an attacker needs to elicit the information disclosure by interacting with the website in unexpected or malicious ways. They will then carefully study the website's responses to try and identify interesting behavior.
Example:



- in this case, the attacker can access the debug page and get some useful information about the system under the website

Examples of information disclosure

So basic information disclosure we can find are

- files for web crawlers
  - we try to access to /robots.txt
  - sitemap.xml
- directory listing
  - we use brute force with a wordlist attack to discover unintended files
  - for example, WordPress is weak and it uses as prefix /wp/files
  - it is used also in sites based on php
- developers comments
  - credentials or known bugs
- error messages and debugging data
  - stack trace and other internal data
  - is better to use in these cases global exception handler
- backup files and version control history that contain source code or credentials
  - in these case, we have to change anything that is accidentally leaked

Sometimes websites encode errors in the URL.

Using gobuster or dirb or wfuzz we can enumerate all directories and files in the websites using a wordlist.

```
gobuster dir -u https://example.com -w /wordlists/Discovery/Web-Content/big.txt -o results.txt
```

We can also use wappalizer to extract info from a website (chrome extension)

Prevention

To prevent information disclosure we have to:

- identify all sensitive information
  - audit code for potential information disclosure

    - don't hardcode credentials and sensitive information
- use generic error messages
    - implement a global exception handler
- disable debug and diagnostic features
    - test for them in the deployed system
- don't use third-party technologies or configurations that are not clear

DIRECTORY TRAVERSAL (PATH TRAVERSAL)

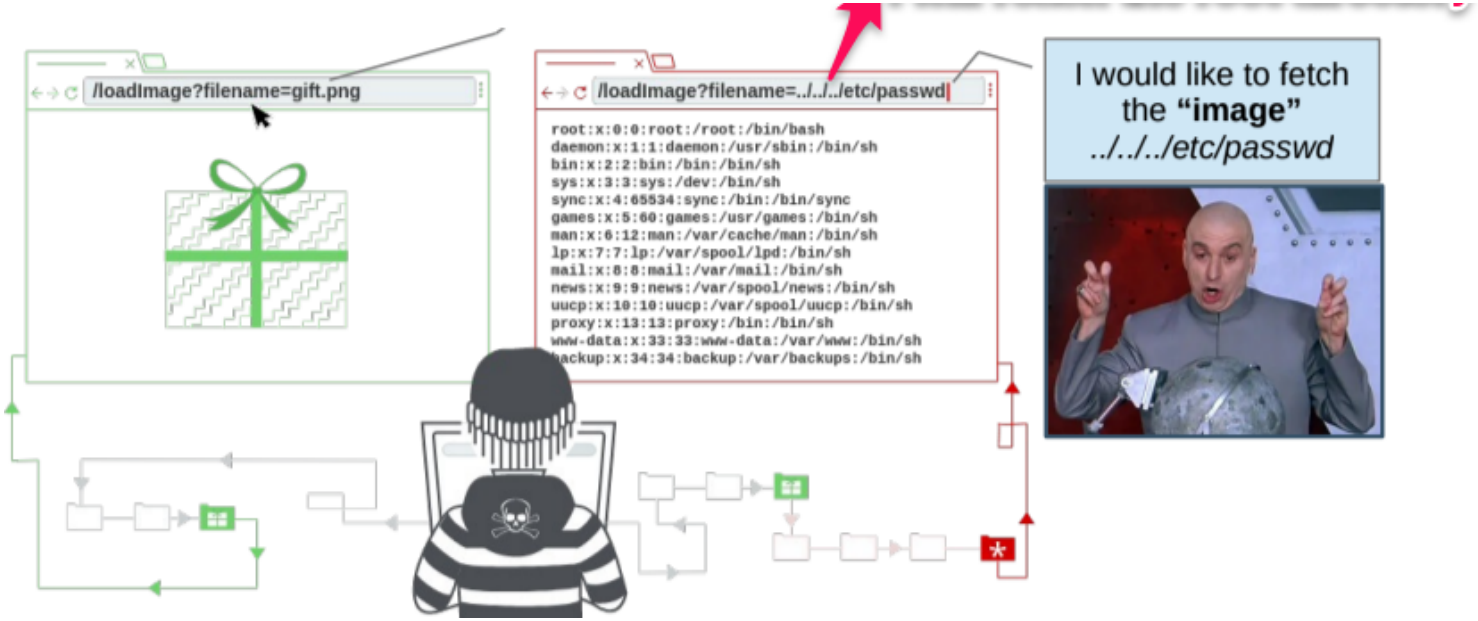
Path traversal is also known as directory traversal. These vulnerabilities enable an attacker to read arbitrary files on the server that is running an application.

This might include:

- Application code and data.
- Credentials for back-end systems.
- 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.

Example:



Common obstacles to exploiting path traversal vulnerabilities

In general, websites have some protection against this problem but in some cases, this protection can be bypassed:

- You might be able to use absolute paths `filename=/etc/passwd`.
  - These can be used if `../` is not allowed
- You might be able to use nested traversal sequences, such as `....//` or `....\/`.
  - These revert to simple traversal sequences when the inner sequence is stripped. (for example, if the website removes ../)
- You can sometimes bypass this kind of sanitization by URL encoding, or even double URL encoding, the `../` characters. This results in `%2e%2e%2f` and `%252e%252e%252f` respectively
  - these can be used when the website checks the URL before decoding it
- It might be possible to include the required base folder followed by suitable traversal sequences. For example: `filename=/var/www/images/../../../etc/passwd`
  - this is when the website requires a fixed prefix
- It might be possible to use a null byte to effectively terminate the file path before the required extension. For example:

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`
    - this is when the website requires a fixed suffix

Sometimes browsers block the file traversal (for example, it cannot render the image if the image is the passwd file), but we can solve this problem using terminal requests or ZAP.

- we can install httpie which has a good terminal interface

How to prevent a path traversal attack

The most effective way to prevent path traversal vulnerabilities is to avoid passing user-supplied input to filesystem APIs altogether.

In general:

- validate the user input before processing it
- after validating the supplied input, append the input to the base directory and use a platform filesystem API to canonicalize the path. Verify that the canonicalized path starts with the expected base directory.

Example in java:

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

OWASP Top Ten

A broad consensus about the most critical security risks to web applications