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.

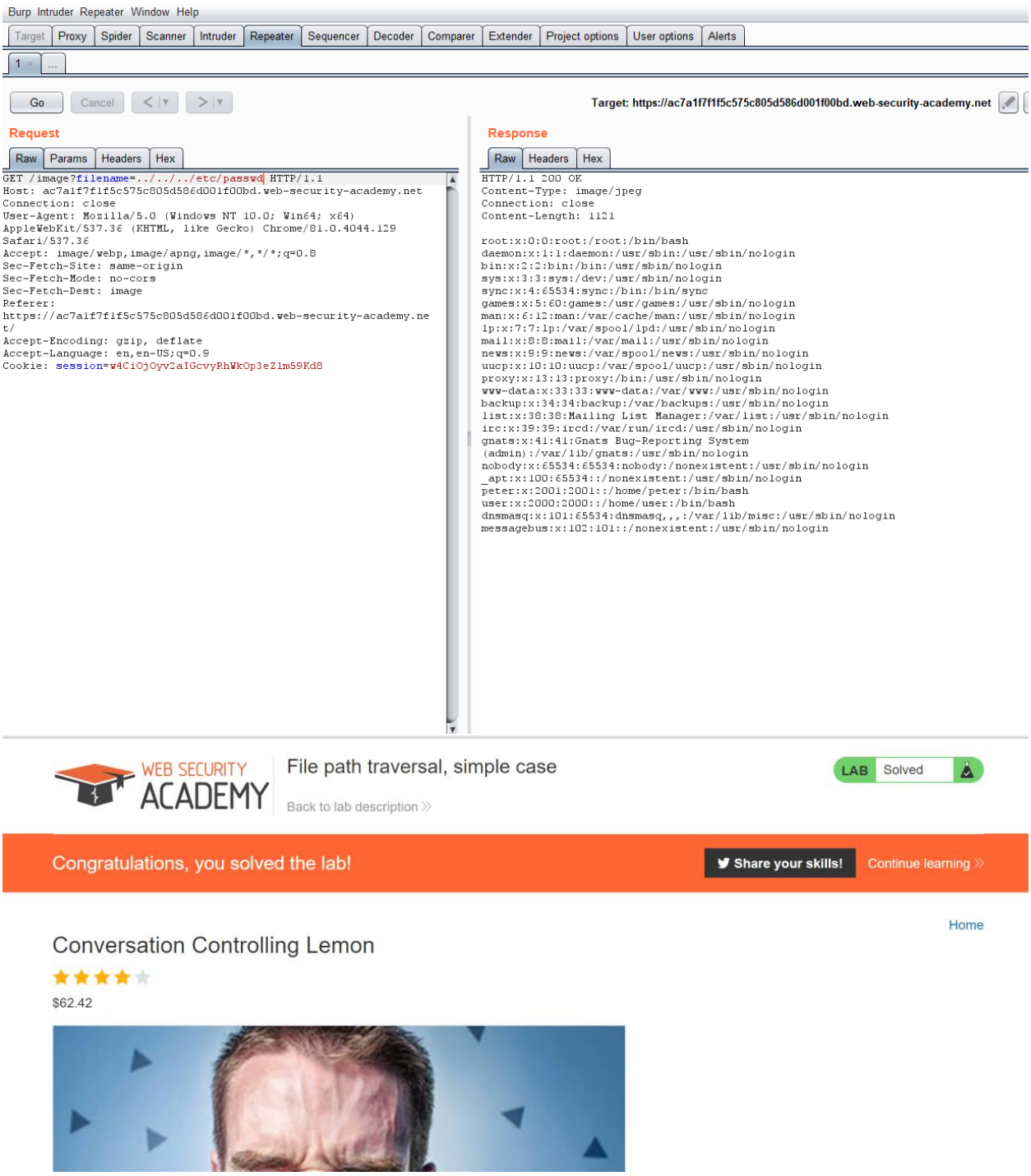
Lab1: File path traversal, simple case

Description: This lab contains a file path traversal vulnerability in the display of product images.

To solve the lab, retrieve the contents of the /etc/passwd file.

Testing procedure and snapshot:

- Use Burp Suite to intercept and modify a request that fetches a product image.
- Modify the filename parameter, giving it the value ../../../etc/passwd.
- Observe that the response contains the contents of the /etc/passwd file.



Lab2: File path traversal, traversal sequences blocked with absolute path bypass

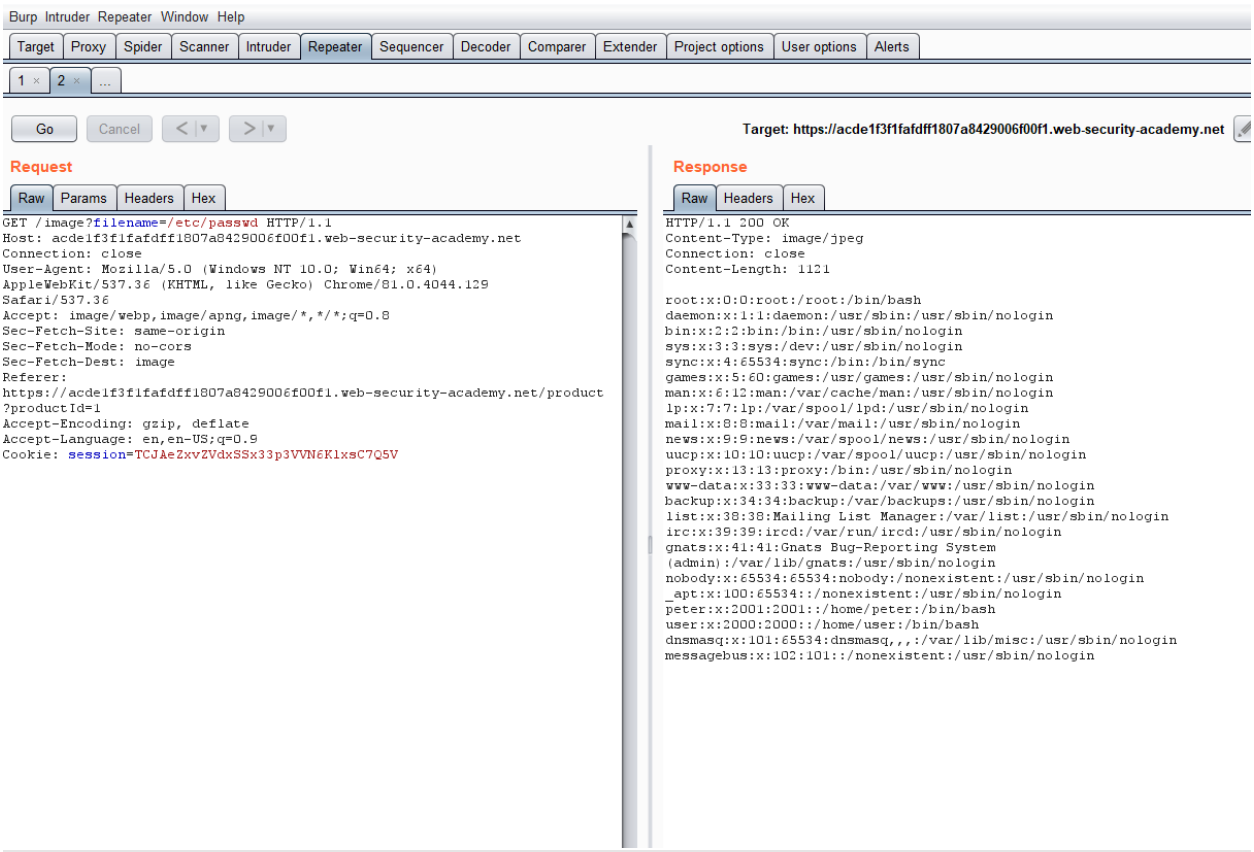
Description: This lab contains a file path traversal vulnerability in the display of product images.

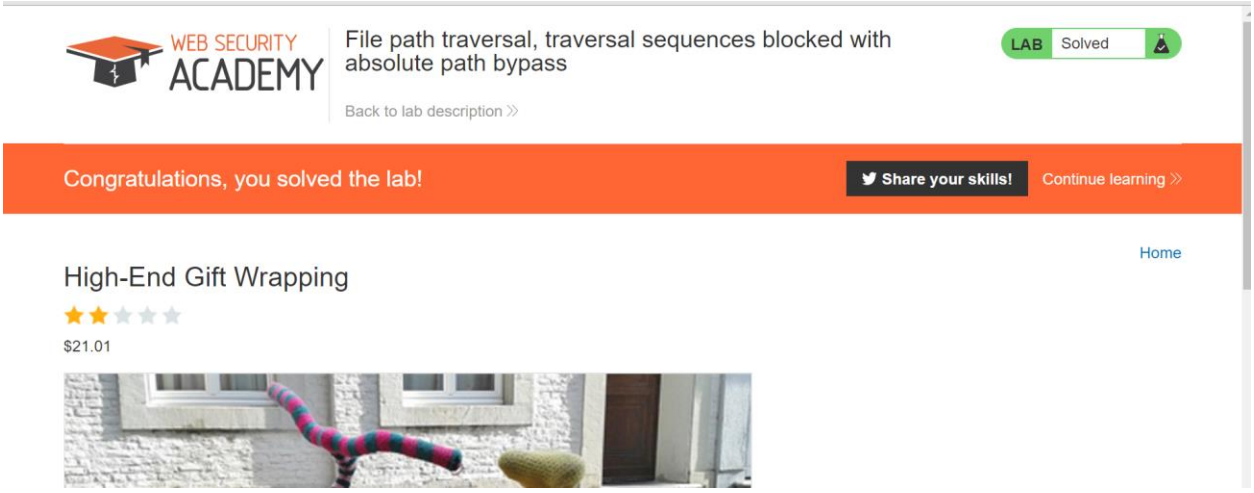
The application blocks traversal sequences but treats the supplied filename as being relative to a default working directory.

To solve the lab, retrieve the contents of the /etc/passwd file.

Testing procedure and snapshot:

- Use Burp Suite to intercept and modify a request that fetches a product image.
- Modify the filename parameter, giving it the value /etc/passwd.
- Observe that the response contains the contents of the /etc/passwd file.





Lab3: File path traversal, validation of start of path

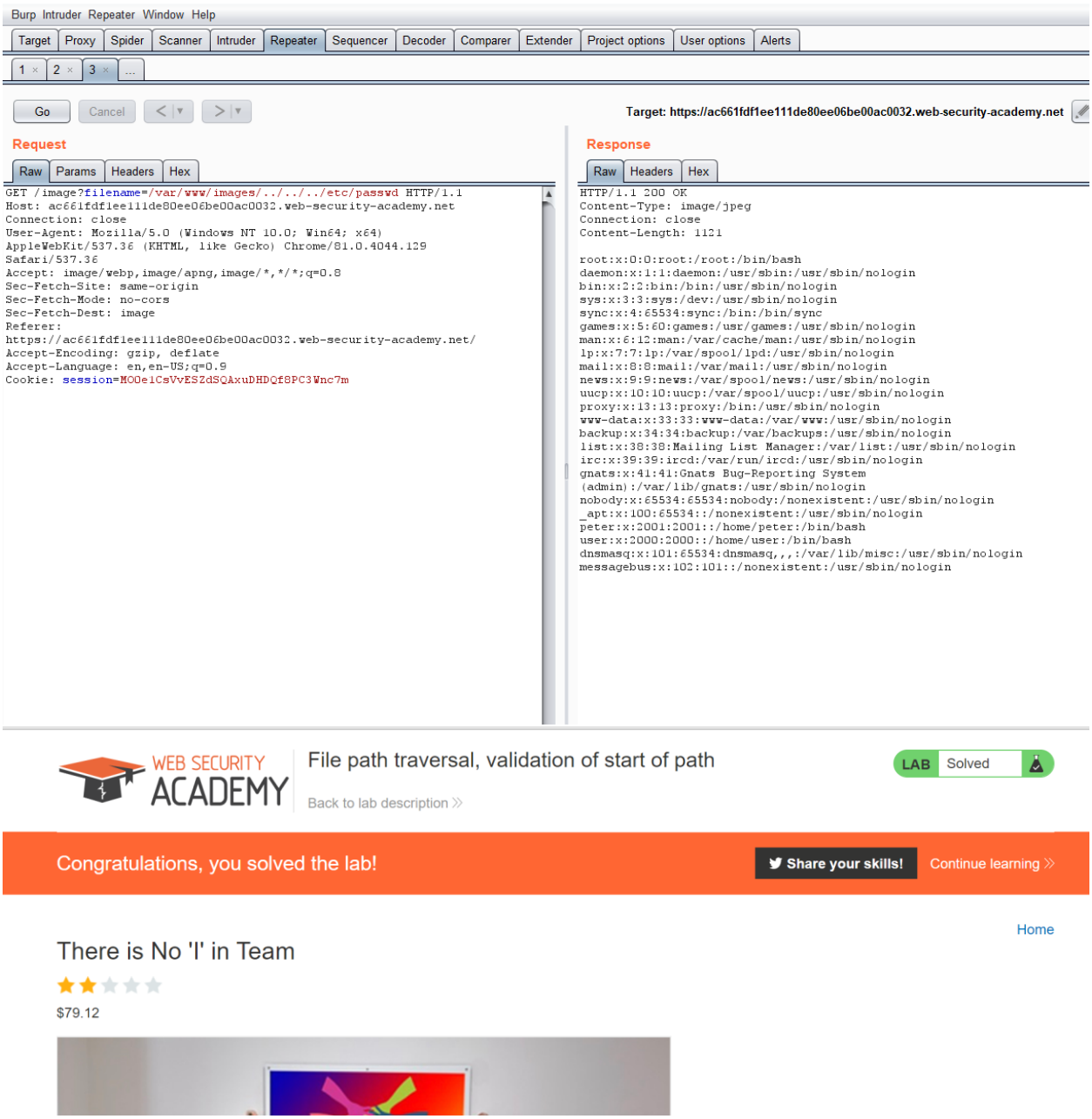
Description: This lab contains a file path traversal vulnerability in the display of product images.

The application transmits the full file path via a request parameter, and validates that the supplied path starts with the expected folder.

To solve the lab, retrieve the contents of the /etc/passwd file.

Testing procedure and snapshot:

- Use Burp Suite to intercept and modify a request that fetches a product image.
- Modify the filename parameter, giving it the value /var/www/images/../../../etc/passwd.
- Observe that the response contains the contents of the /etc/passwd file.



Lab4: File path traversal, validation of file extension with null byte bypass

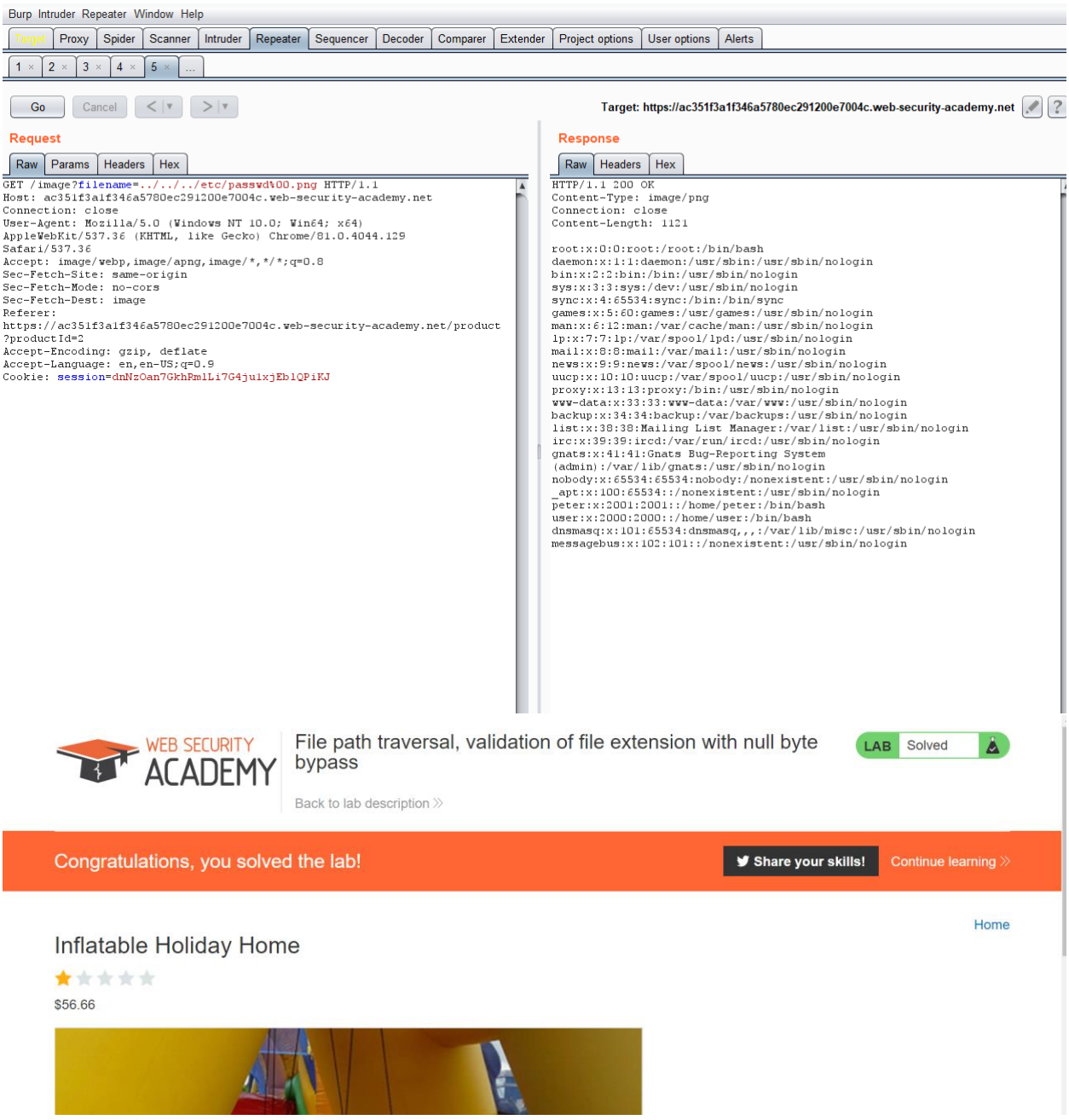
Description: This lab contains a file path traversal vulnerability in the display of product images.

The application validates that the supplied filename ends with the expected file extension.

To solve the lab, retrieve the contents of the /etc/passwd file.

Testing procedure and snapshot:

- Use Burp Suite to intercept and modify a request that fetches a product image.
- Modify the filename parameter, giving it the value ../../../etc/passwd%00.png.
- Observe that the response contains the contents of the /etc/passwd file.



Lab5: File path traversal, traversal sequences stripped non-recursively

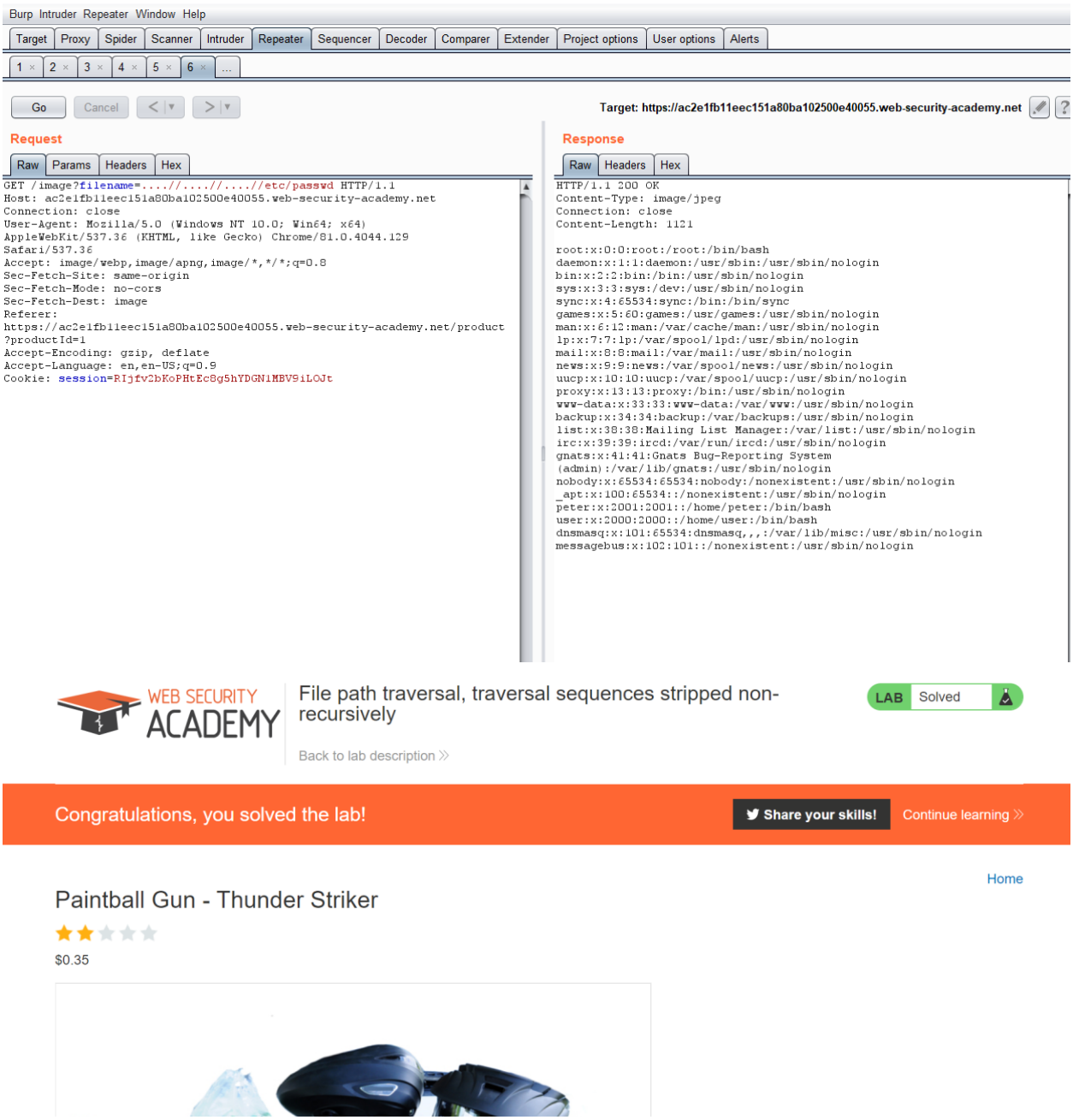
Description: This lab contains a file path traversal vulnerability in the display of product images.

The application strips path traversal sequences from the user-supplied filename before using it.

To solve the lab, retrieve the contents of the /etc/passwd file.

Testing procedure and snapshot:

- Use Burp Suite to intercept and modify a request that fetches a product image.
- Modify the filename parameter, giving it the value: ….//….//….//etc/passwd
- Observe that the response contains the contents of the /etc/passwd file.



Lab6: File path traversal, traversal sequences stripped with superfluous URL-decode

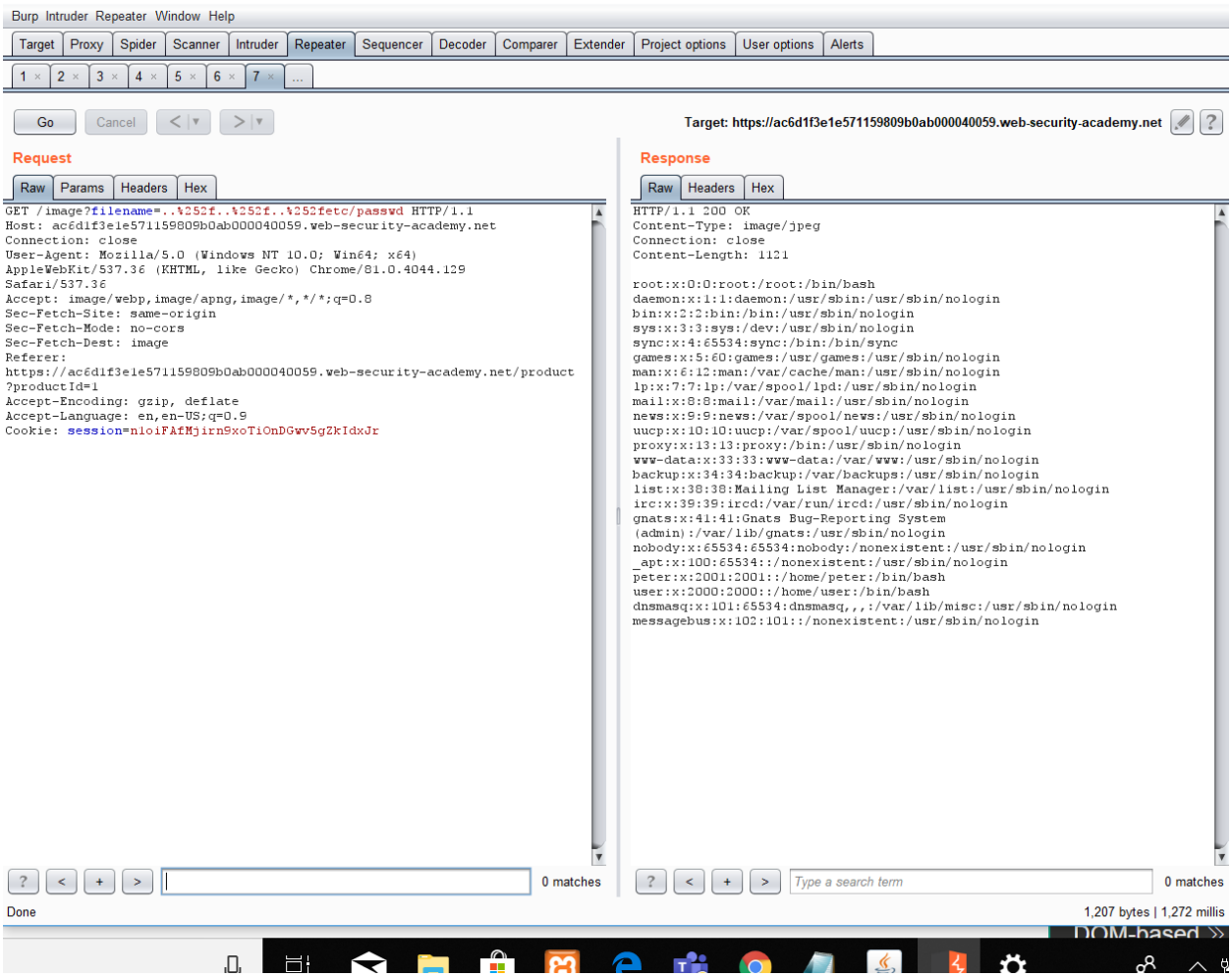
Description: This lab contains a file path traversal vulnerability in the display of product images.

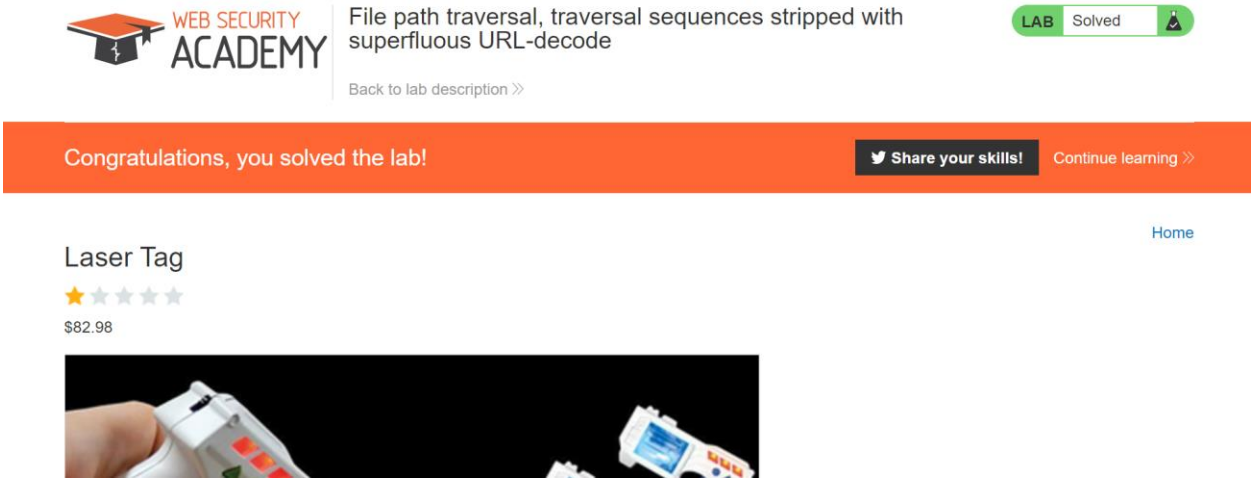
The application blocks input containing path traversal sequences. It then performs a URL-decode of the input before using it.

To solve the lab, retrieve the contents of the /etc/passwd file.

Testing procedure and snapshot:

- Use Burp Suite to intercept and modify a request that fetches a product image.
- Modify the filename parameter, giving it the value ..%252f..%252f..%252fetc/passwd.
- Observe that the response contains the contents of the /etc/passwd file.





How to prevent a directory traversal attack?

The most effective way to prevent file path traversal vulnerabilities is to avoid passing user-supplied input to filesystem APIs altogether. Many application functions that do this can be rewritten to deliver the same behavior in a safer way.

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
}
```