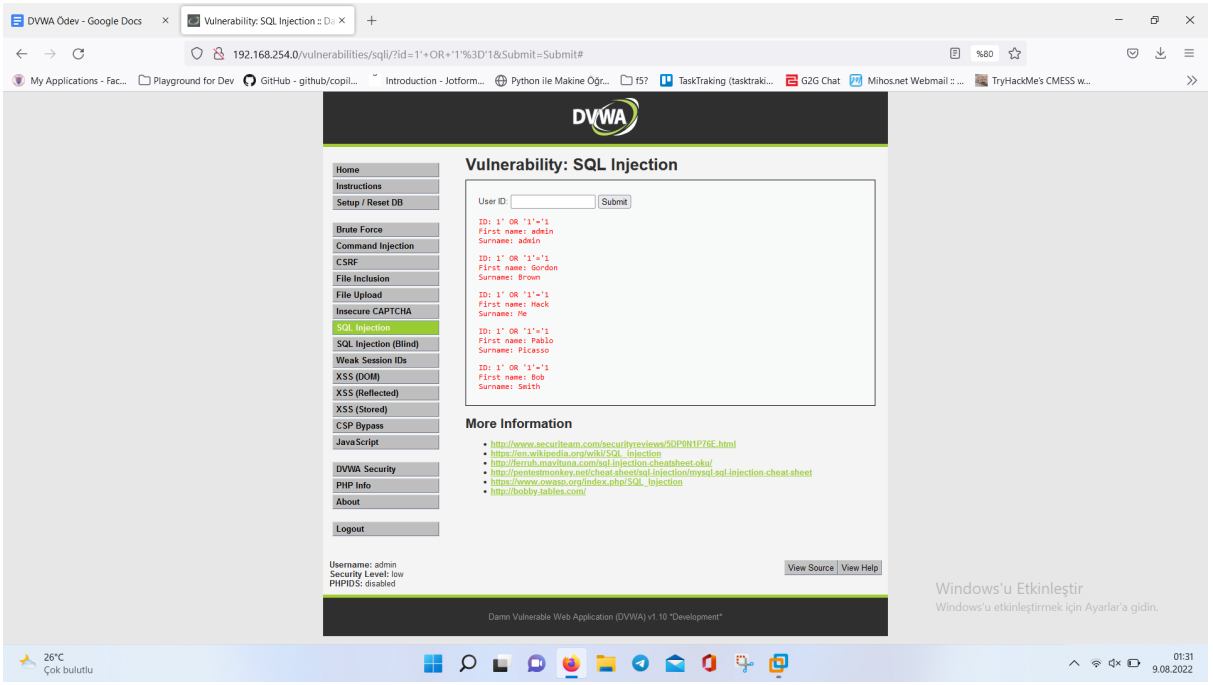
DVWA Ödev

1- SQL Injection

Low Seviye Kaynak Kodu

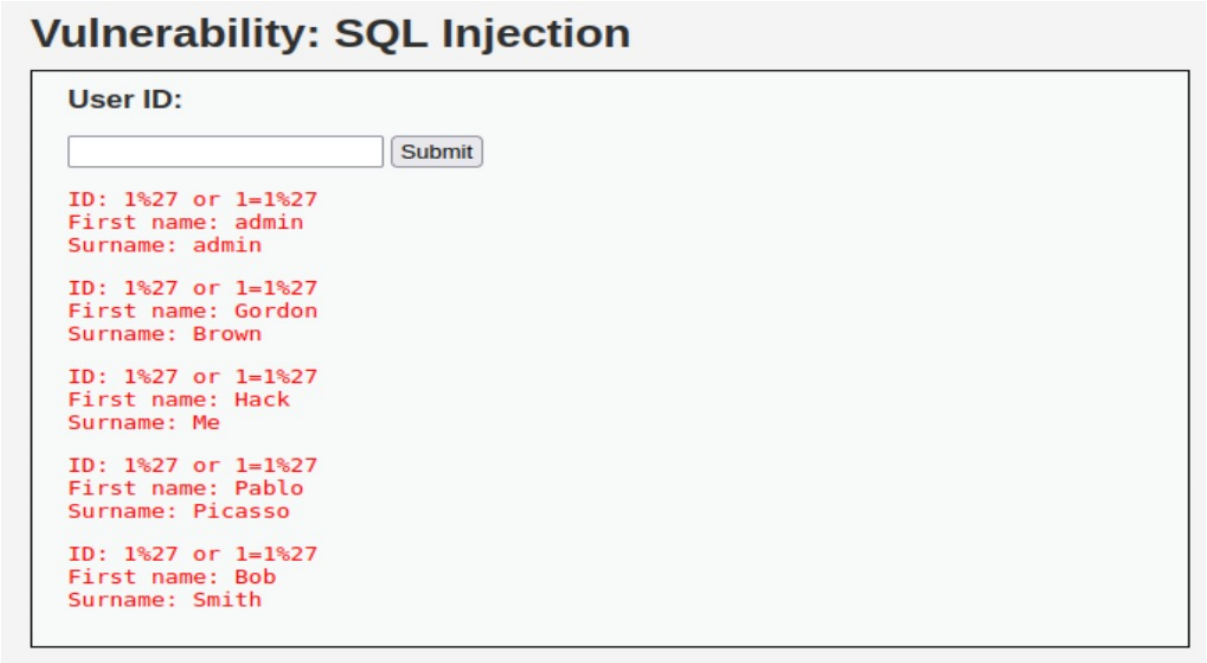
```
 $query  = "SELECT first_name, last_name FROM users WHERE user_id = '$id';";
$result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' .
((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) :
(($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : false)) . '</pre>' );
```

Yukarıdaki satırda girdi direkt olarak sorgu içerisine eklendiği için SQL Injection zafiyeti oluşmaktadır. Daha sonra da herhangi bir filtreleme işlemi olmadan query çalıştırılmaktadır.



Medium Seviye Kaynak Kodu

Parametre sorguya eklenmeden önce “mysqli_real_escape_string” fonksiyonu kullanılarak girdi filtrelenmektedir. Örn: Tırnak (‘) karakteri eklendiğinde (‘/) yaparak açtığımızı tırnakların devamına kaçış karakteri eklemektedir. Ancak, bu filtreleme url encoding yapılarak bypass edilebilir bir fonksiyon olduğu için yeterli değildir.



High Seviye Kaynak Kodu

$query = "SELECT first_name, last_name FROM users WHERE user_id = '$id' LIMIT 1;";

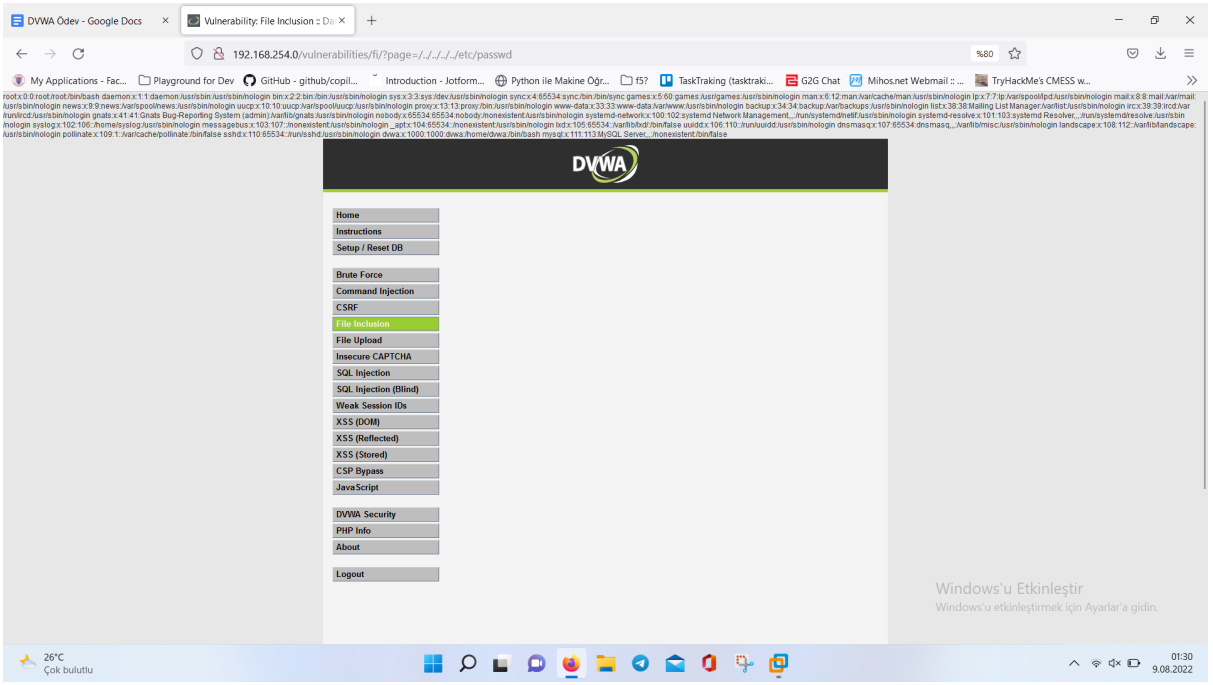
Yukarıdaki kodda sorgunun sonuna “LIMIT 1” ekleyerek dönen sonucun 1 tane ile sınırlamasını sağlamışlar.

2- File Inclusion

Low Seviye Kaynak Kodu

```
$file = $_GET[ 'page' ];
```

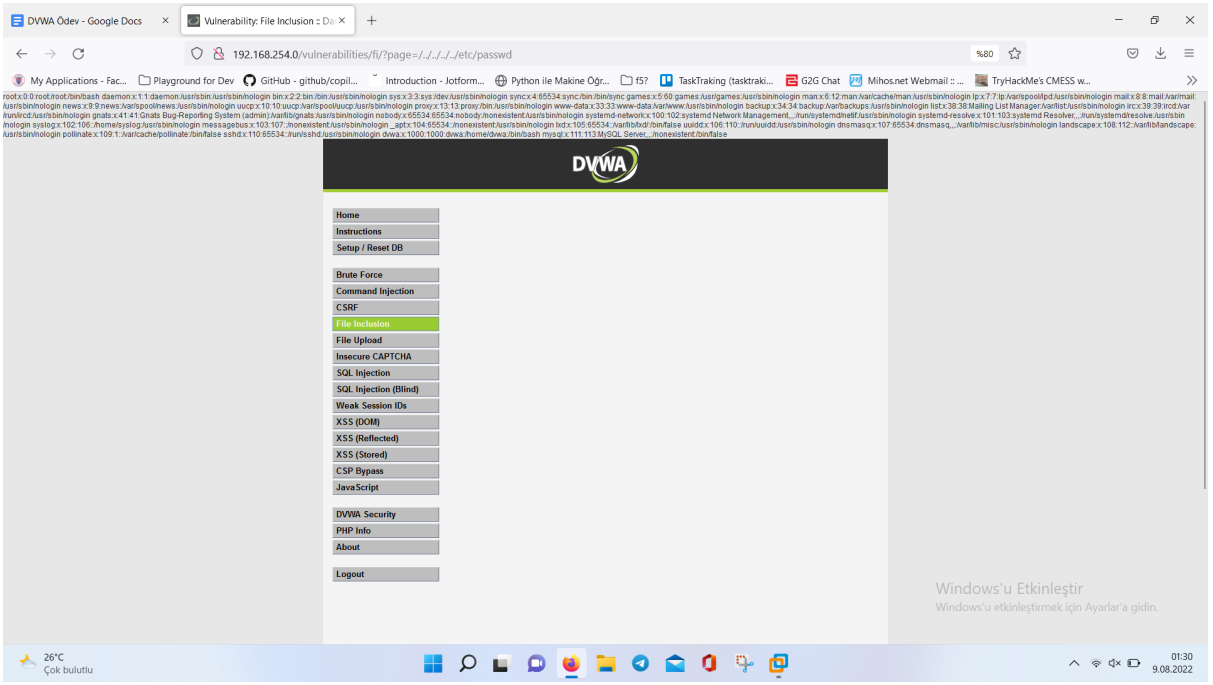
Yukarıdaki kaynak kodu page parametresi ile gelen girdi değerini direkt olarak dizinler arasında arayıp çağırmaktadır. Bu noktada parametreye istediğimiz değeri girerek istediğimiz dosyayı elde edebiliriz. Örneğin, “/../../../../etc/passwd” yazarak bulunduğumuz dizinden çıkıp /etc/passwd dosyasını okuyabiliriz.



Medium Seviye Kaynak Kodu

```
$file = str_replace( array( "http://", "https://" ), "", $file );
$file = str_replace( array( "../", "..\"" ), "", $file );
```

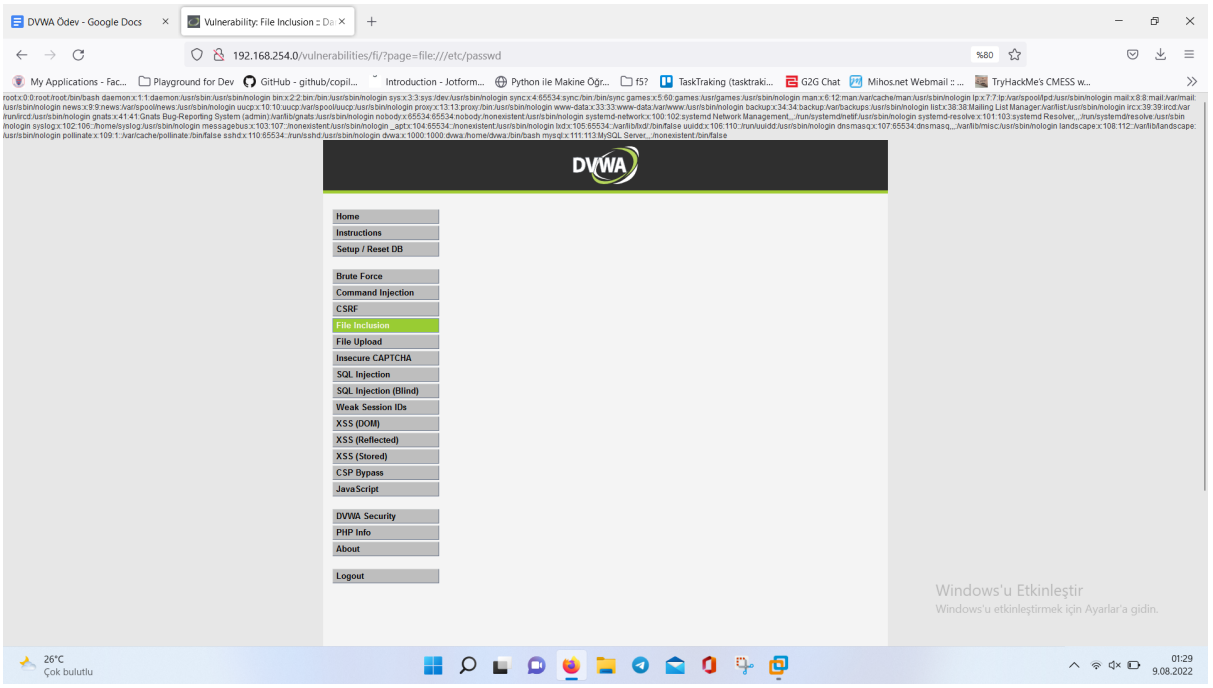
Düşük seviye zafiyet üzerine girdi validasyonu ekleyerek filtreleme yapılmaya çalışılmıştır. Örneğin, “http” istekleri “https” isteklerine ve “../” kaçış karakterleri “..\”” ile değiştirilmektedir. Yine aynı şekilde, “/../../../../etc/passwd” yazarak bulunduğumuz dizinden çıkıp /etc/passwd dosyasını okuyabiliriz.



High Seviye Kaynak Kodu

```
if( !fnmatch( "file*", $file ) && $file != "include.php" ) {
```

Burada girdi validasyonu için if koşulu içerisinde “fnmatch” fonksiyonu kullanılarak girilen girdinin Unix dosya ismi patternına uyup uymadığı ve file ile başlayıp başlamadığı kontrol edilir. Diğer kısımda “include.php” dosya isminin include.php içerisinde olup olmadığı kontrol edilir. Daha sonra, “&&” ile iki eşitliğin sonucu birleştirilir. Burada da “file:///etc/passwd” kullanarak istediğimiz dizindeki dosyayı elde edebiliriz.



3- Reflected XSS

DVWA makinemde XSS’ler çalışmadığı için görüntüleri koyamadım. Aşağıdaki payloadlar örnek olarak adım adım kullanılabilir.

<script>alert('XSS Test');</script>
<img src=”#” onclick=alert(“XSSed!”) >
<img src=”#” onclick=alert(document.cookie) >

Low Seviye Kaynak Kodu

```
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
```

Burada name parametresinde verilen girdi hiç bir filtreleme işlemi yapılmadan çalıştırıldığı için xss zafiyeti ortaya çıkarmaktadır. Herhangi bir kayıt işlemi olmadığı içinde reflected’dır.

Medium Seviye Kaynak Kodu

```
$name = str_replace( '<script>', '', $_GET[ 'name' ] );
```

Low seviyesinin üzerine “<script>” etiketini filtreleyecek bir fonksiyon eklenmiştir. Ancak, bu da filtreleme sonucunda “<script>” kelimesini ortaya çıkaracak iç içe parçalar eklenerek aşılabilir.

High Seviye Kaynak Kodu

```
$name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $_GET[ 'name' ] );
```

Medium seviyesinde “str_replace” fonksiyonun yetersizliğinden dolayı aşıldığı için burada filtreleme işlemi yapılırken "preg_replace” fonksiyonu kullanılmıştır. Ancak, bu yöntem de yetersizdir. Büyük küçük harfler kullanılarak aşılabilir.

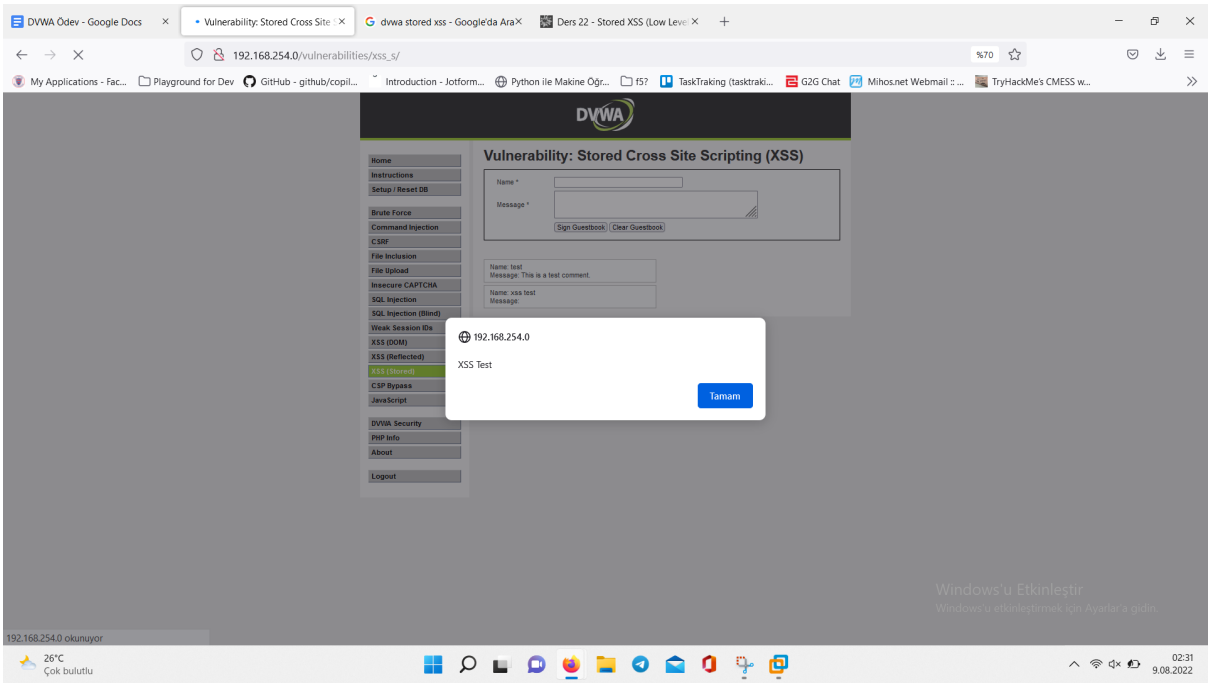
4- Stored XSS

DVWA için reflected XSS payloadları ile aynıdır.

Low Seviye Kaynak Kodu

```
$query  = "INSERT INTO guestbook ( comment, name ) VALUES ( '$message', '$name' );";
```

Sorgu kısmına gelene kadar girdiler üzerinde kaçış karakterleri, trim, strip dışında filtreleme yapılmamaktadır. Daha sonra yukarıdaki sorgu ile girilen değerlerimiz veri tabanına kaydedilmektedir (stored). Sayfada veritabanından çekilen veriler hakkında herhangi bir js filtrelemesi bulunmamaktadır. Bu yüzden girdi olarak js kodu (<script>alert('XSS Test');</script>) ekleyerek, sayfayı yenileyerek veri tabanından yüklemesini sağlayıp istediğimiz komutu çalıştırabiliriz.



Medium Seviye Kaynak Kodu

```
$name = str_replace( '<script>', '', $name );
```

Low seviye kodundan farklı olarak “htmlspecialchars” fonksiyonu ile özel html karakterleri filtrelenmiştir. “str_replace” fonksiyonu içerisinde “<script>” etiketi filtrelenmiştir. Ancak, bu iki yöntem de aşılarak zafiyet sömürülebilir. Filtreleme işleminden sonra filtrelenen etiket kalacak şekilde iç içe script kelime parçaları yazılarak aşılabilir.

High Seviye Kaynak Kodu

```
$name = preg_replace( '/<(.*)s(.*)c(.*)r(.*)i(.*)p(.*)t/i', '', $name );
```

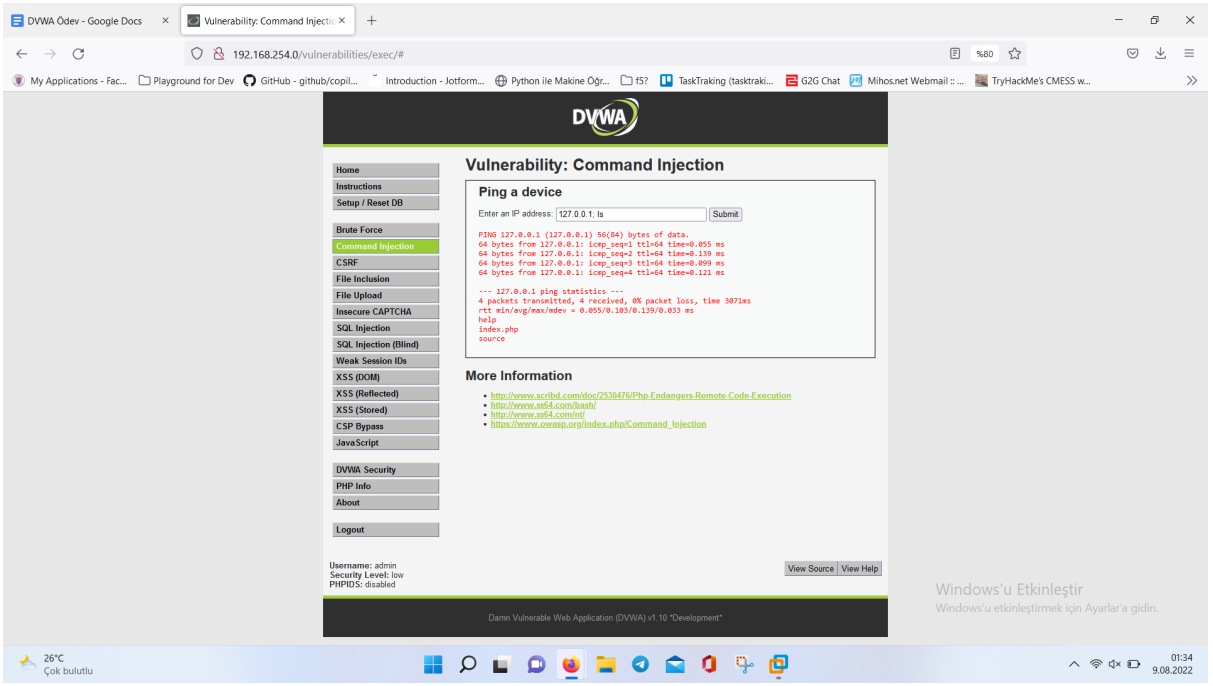
Medium seviyesinde “str_replace” fonksiyonun yetersizliğinden dolayı aşıldığı için burada filtreleme işlemi yapılırken "preg_replace” fonksiyonu kullanılmıştır. Ancak, bu yöntem de yetersizdir. Büyük küçük harfler kullanılarak aşılabilir.

5- Command Injection
Low Seviye Kaynak Kodu

```
$cmd = shell_exec( 'ping  ' . $target );
```

“shell_exec — Komutu kabukta çalıştırır ve çıktısının tamamını bir dizge olarak döndürür“

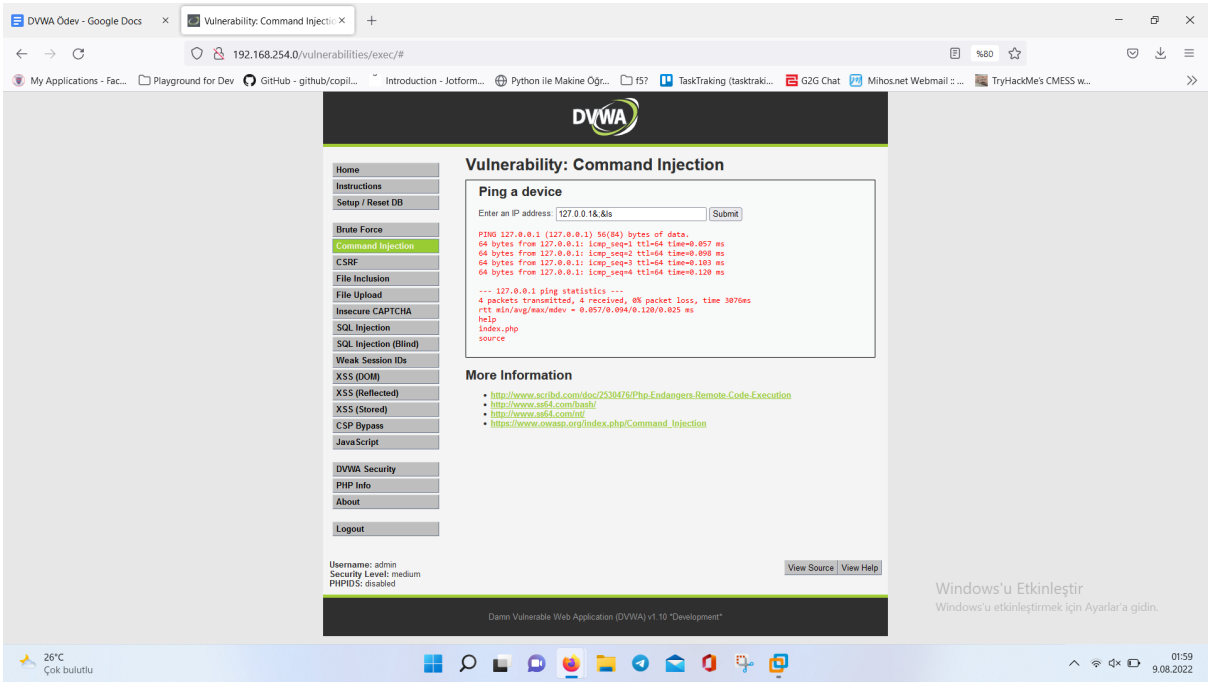
$target değişkenine girilen girdi direkt olarak “shell_exec” fonksiyonuna eklenmektedir. shell_exec fonksiyonu da yukarıda belirtildiği gibi bir kabuk çalıştırıp komutu içerisinde çalıştırır ve sonucu döner. Bu yüzden “127.0.0.1; ls” yazarak terminaldeki gibi komutları ard arda çalıştırıyoruz.



Medium Seviye Kaynak Kodu

```
$substitutions = array(
        '&&' => '',
        ';'  => '',
);
```

Low level üzerinde yukarıdaki kod taban alınarak kara liste filtrelemesi eklenmiştir. Burada “&&” ve “;” stringleri bulunduğunda siliniyor. Ancak, stringleri parçalayıp birbirleri ile birleştirerek zafiyeti sömürebiliriz (“&;&”) -> (“;”).

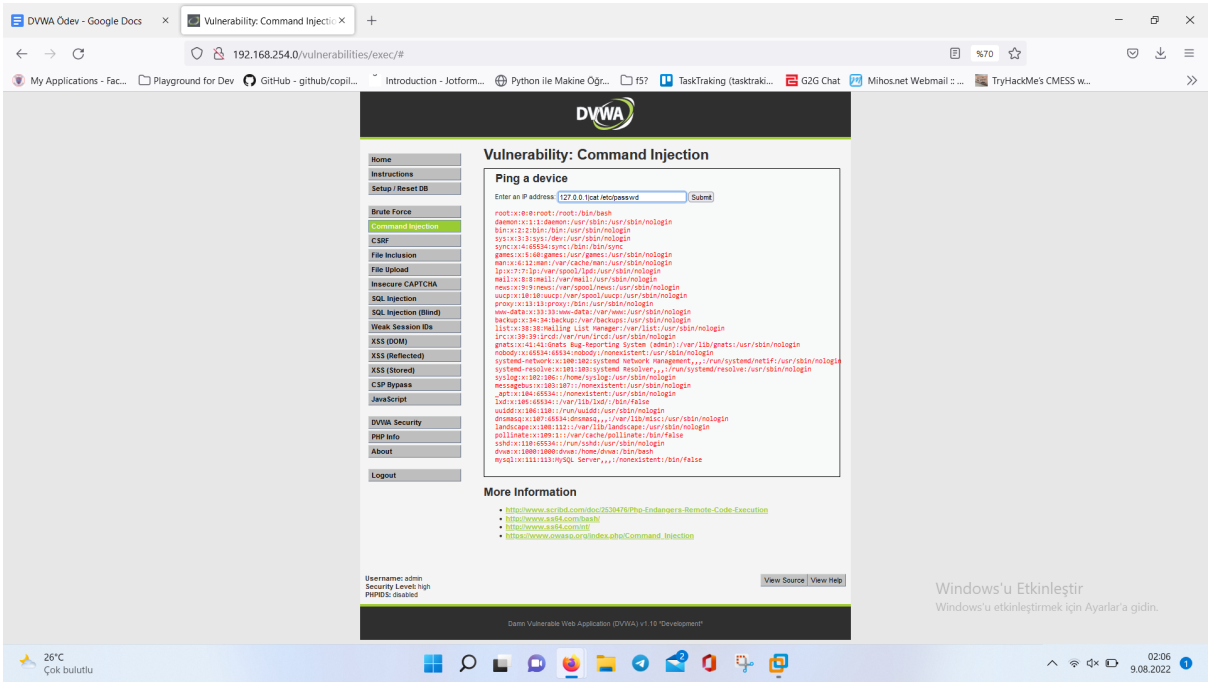


High Seviye Kaynak Kodu

```
// Get input
$target = trim($_REQUEST[ 'ip' ]);

// Set blacklist
$substitutions = array(
	'&'  => '',
	';'  => '',
	'| ' => '',
	'-'  => '',
	'$'  => '',
	'('  => '',
	')'  => '',
	'`'  => '',
	'||' => '',
);
```

Benzer şekilde ip parametresi ile gönderilen girdi trim edilerek girdinin başındaki ve sonundaki boşluk karakterleri silinmiştir. Daha sonra kara liste filtrelemesi genişletilerek belirtilen bütün karakterler silinmiştir. Kara liste filtrelemesi kullanırken “str_replace” fonksiyonu kullanılmıştır. Bu fonksiyonun “değiştirme sırası sorunsalı” isimli filtrelemeyi eksik yapmasına neden olan bir problemi olduğu için, “127.0.0.1|cat /etc/passwd” yazarak zafiyeti sömürebiliriz.



Faruk Ulutaş