

# 1. Broken Access Control

## Nedir?

Belirli kaynak yada sistemlere erişimin yetkilendirilmediği zaman yada kötü yetkilendirilmişse meydana gelen zafiyet çeşididir.

Bu mekanizmada yetkilendirme üzerine kurgulanmıştır, bu sayede herkes herşeye dileceğinde erişemeyecek, hassas veriler güvende olacaktır.

Erişim Kontrolü, 4 alt başlıkta sıralanabilir.

### 1. Discretionary Access Control (DAC):

Kaynak sahibi(owner) erişim izinlerini kişisel olarak yönettiği sistemdir.

### 2. Mandatory Access Control (MAC):

Erişim kuralları admin tarafından belirlenir ve uyulması esastır, adminin koyduğu erişim kuralları değiştirilmez.

### 3. Role-Based Access Control (RBAC):

Her kullanıcının rol havuzundan gelen rolü olur ve yetkiler de rolüyle sınırlıdır. Erişim düzeyleri rollere göre belirlenir.

### 4. Attribute-Based Access Control (ABAC):

Erişim yetkileri, kullanıcının yönü ve daha birçok faktöre bağlı olarak dinamik olarak değişir.

Broken Access Control, kullanıcıların yetkileri dışında kaynaklara veya verilere erişim sağladığı durumları ifade eder.

Yaygın sorunlar şunlardır:

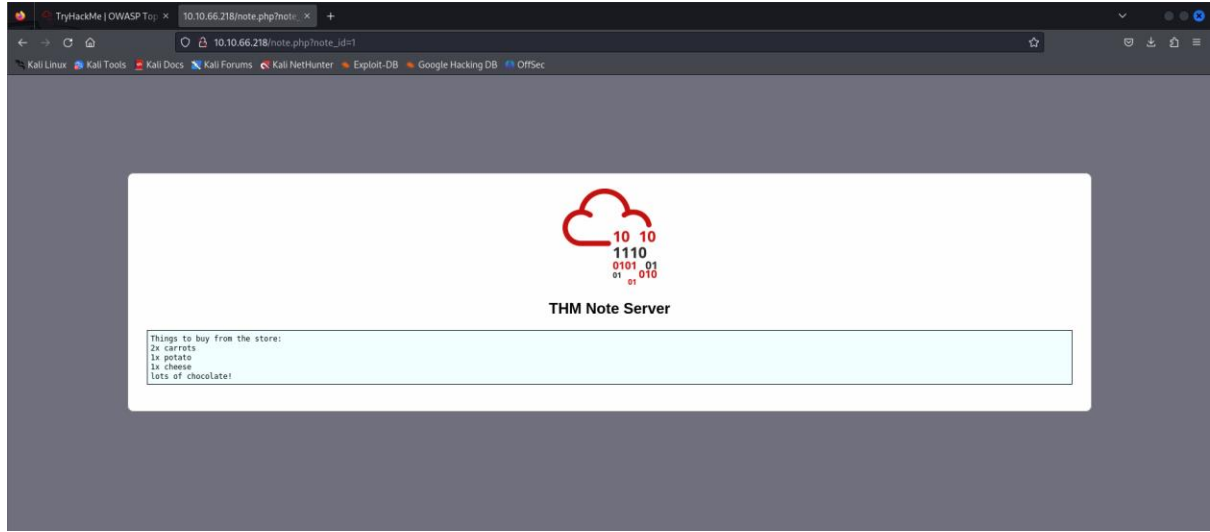
**Yatay Ayrıcalık Yükseltmesi:** Aynı erişim seviyesine sahip diğer kullanıcıların verilerine erişim. Örneğin, URL'deki kullanıcı kimliğini değiştirerek başka birinin hesabına girme.

**Dikey Ayrıcalık Yükseltmesi:** Daha yüksek yetkilere sahip kullanıcıların kaynaklarına erişim. Örneğin, normal bir kullanıcının yönetim işlevlerine erişmesi.

**Yetersiz Erişim Kontrolleri:** Erişim kontrolü yeterince sıkı değilse, hassas verilere erişim sağlanabilir. Örneğin, kullanıcıların izinlerini kontrol etmeden verileri görüntülemelerine izin verme.

**Güvenli Olmayan Doğrudan Nesne Referansları (IDOR):** Tahmin edilebilir isimlerle hassas verilere erişimdir. Örneğin, kolayca tahmin edilebilen ID'lerle verilere ulaşma.

## 1.a) <https://tryhackme.com/r/room/owasptop102021> - Broken Access Control (IDOR Challenge)



### Senaryo:

Bu makinada URL üzerinden belirli notlara erişim sağlanıyor. [http://10.10.66.218/note.php?note\\_id=1](http://10.10.66.218/note.php?note_id=1)

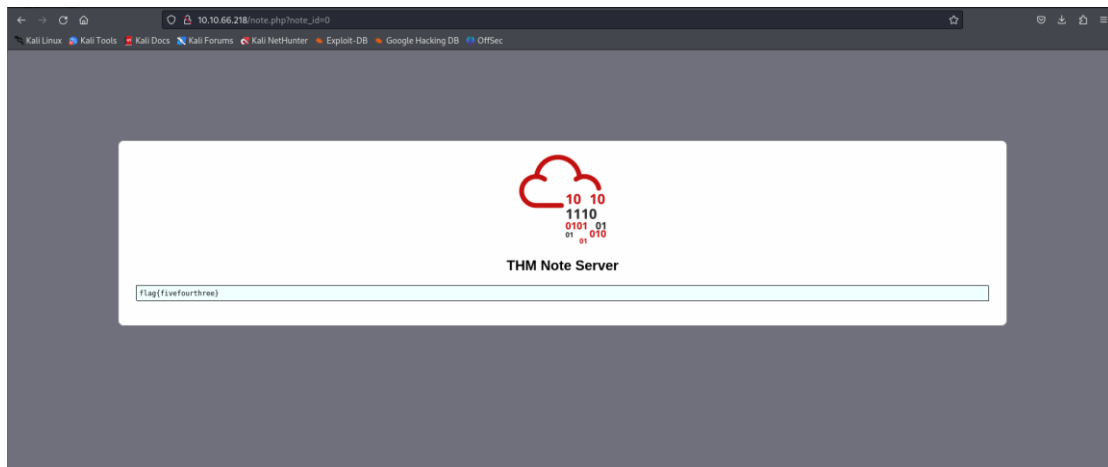
note\_id parametresi, erişilmek istenen notun kimliğini belirtiyor.

Normalde, bu parametreye erişim yetkisi olan bir kullanıcının yalnızca kendi notlarına erişebilmesi beklenir.

### Problemin Açıklaması:

Bu uygulamada, note\_id parametresinin değerini değiştirerek farklı notlara erişim sağlanabiliyor.

Örneğin, note\_id=0 olarak değiştirince, özel bir flag ortaya çıkıyor.



**Yatay Ayrıcılık Yükseltmesi:** note\_id = 0 gibi URL den kolayca değiştirilecek bu işlemde diğerlerinin bilgilerine de sızıntı yapılabilmektedir.

Bu tarz zafiyetleri önlemek için, kullanıcının başkalarının hassas bilgilerine bu şekilde referans değiştirme yordamıyla erişmesinin önüne geçilmelidir.

Ve her erişim isteğini kullanıcıdan alınan token vasıtasıyla sunucu tarafında doğrulamamız gerekmektedir.

## 1.b) <https://tryhackme.com/r/room/owaspbrokenaccesscontrol>

Welcome To VulnerableApp

Creating an account is absolutely free!

Create an account

First Name

Your first name

Last Name

Your last name

Email

Enter a valid email

Password

Password

Re-enter Password

Password confirmation

Create account

Already have an account? [Login](#)

Welcome To VulnerableApp

Login to view your dashboard.

Login

Email

Enter email

Password

Enter your password

Submit

Don't have an account? [Register](#)

Welcome, qwe

[Logout](#)

Announcements

Status Update Test

by: admin

Application building in progress

Report the bugs

by: admin

Pis email me at admin@admin.com for any bugs that you will encounter. Thanks

Online users

admin@admin.com

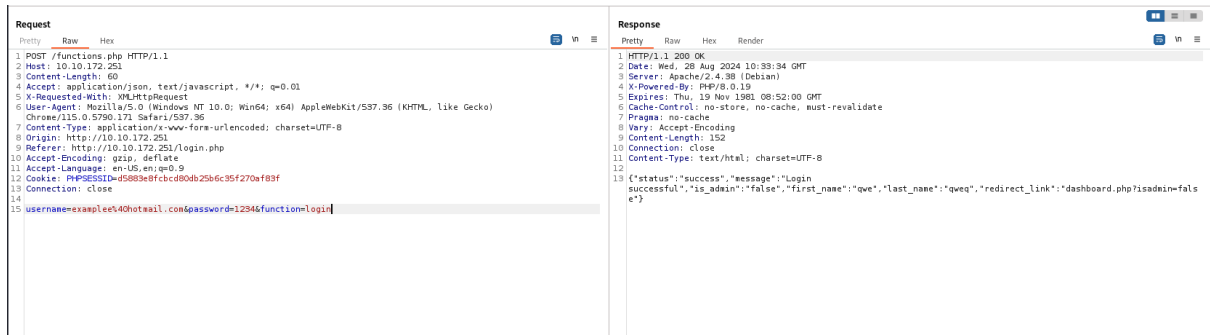
example@hotmail.com

Bu makine, **Dashboard**, **Login** ve **Registration** formlarına sahip.

Senaryo olarakta:

Penetrasyon testçisi olarak, önce bir hesap kaydedilecek, ardından login fonksiyonunu kontrol ederek erişim kontrol zafiyetlerini test edilecek.

Http Trafikini BurpSuit kullanarak yakaladım.



Login sonrası **functions.php**'ye request yollamaktadır.

Oturum açma işlemi tamamlandıktan sonra, response olarak; **status**, **message**, **first\_name**, **last\_name**, **is\_admin** ve **redirect\_link** bilgilerini içeren JSON yanıt gelmektedir.

Response kısmından aşağıdaki soruları cevaplayabiliriz.

**Soru 1:** Web uygulamasını barındıran sunucunun türü nedir? Bu, Burp Suite'teki isteğin yanıtında bulunabilir.

**Cevap:** Apache

**Soru 2:** Yeniden yönlendirme bağlantısı içeren oturum açma isteğinden gelen JSON yanıtındaki parametrenin adı nedir?

**Cevap:** redirect\_link

**Soru 3:** Hangi Burp Suite modülü kendimizle hedefimiz arasındaki istekleri ve yanıtları yakalamamıza olanak tanır?

**Cevap:** Proxy

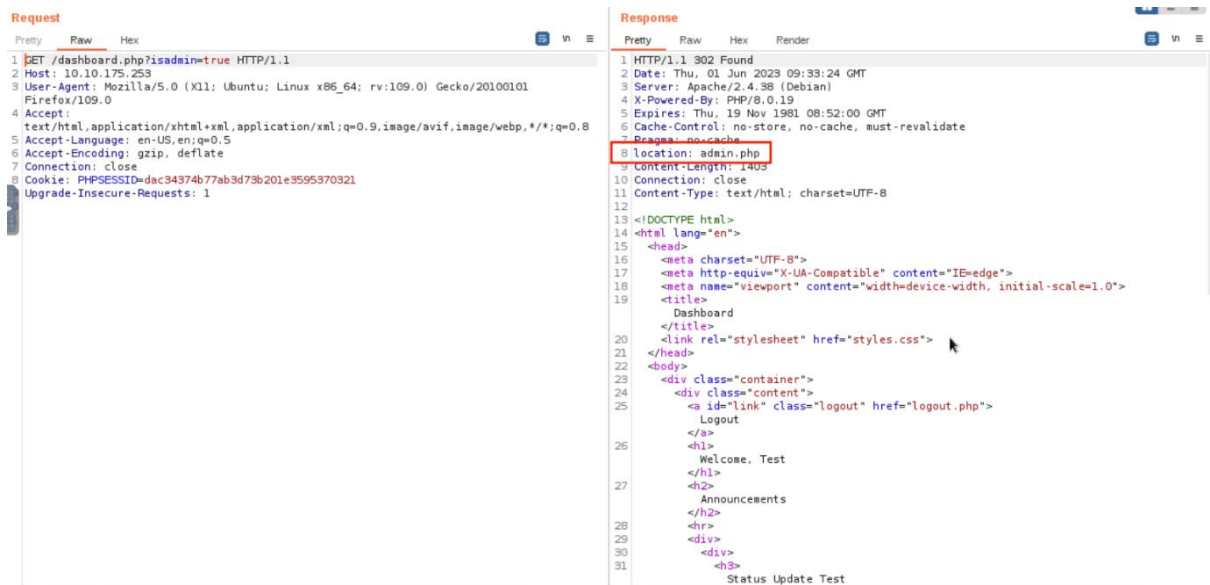
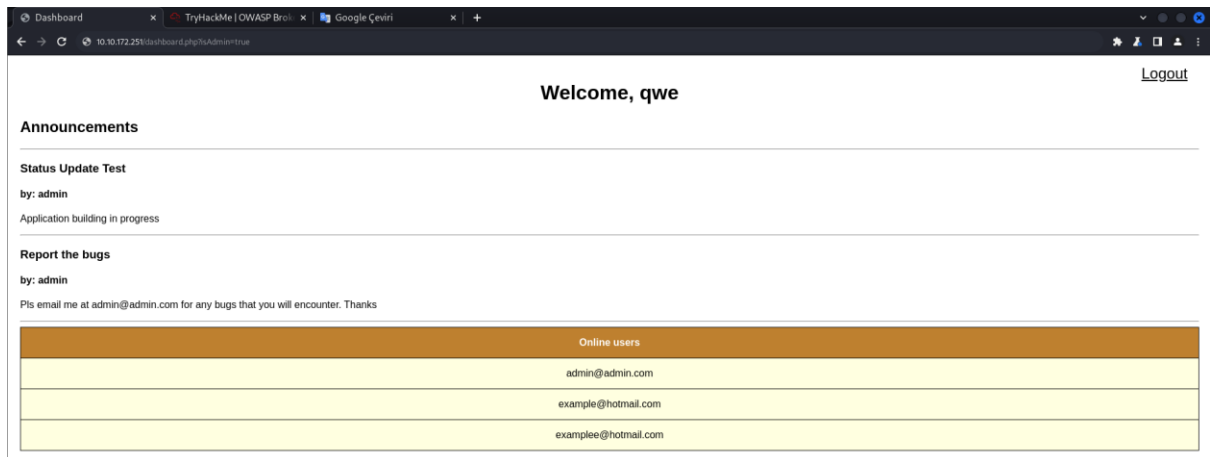
**Soru 4:** Çevrimiçi kullanıcılar tablosunda bulunabilecek yönetici e-postası nedir?

Dashboard > Online Users > [admin@admin.com](mailto:admin@admin.com) olduğu gözükmektedir.

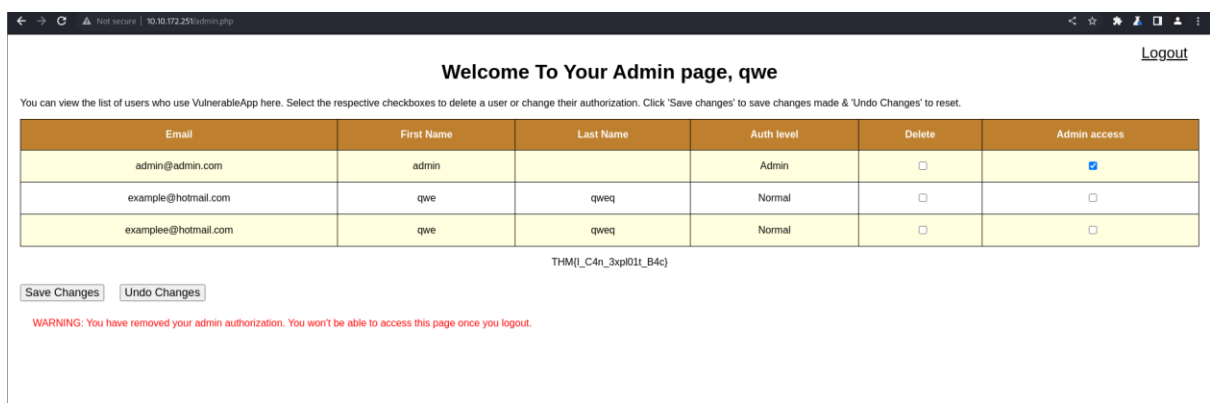
**Cevap:** admin@admin.com

“redirect\_link” : “dashboard.php?isadmin=false” key/value parametresi ise, kullanıcıyı admin olup olmadığına göre **dashboard.php**'ye yönlendirmekte kullanılacaktır.

False yerine True parametresi geçip admin yetkisi alacağım.



Admin.php'ye yönlendirdi.



**Soru 5:** Admin.php'ye eriştikten sonra ne tür bir ayrıcalık artışı yaşandı?

**Cevap:** Vertical

**Soru 6:** Saldırganın yönetici sayfasına erişmesine hangi parametre izin verir?

**Cevap:** isAdmin

**Soru 7:** Yönetici sayfasındaki bayrak nedir?

**Cevap:** THM{I\_C4n\_3xpl01t\_B4c}

**Php** uygulamasında **Broken Access Control** zafiyetinden kaçınmanın birkaç yolu vardır.

### 1. Implement Role-Based Access Control (RBAC): (Rol Tabanlı Erişim Kontrolü)

**Nedir?:** Kullanıcıların hangi işlemleri yapabileceğini rollerine göre belirlemek. Örneğin, bir admin her şeyi yapabilirken, user sadece okumaya izinelidir.

**Nasıl Yapılır?:** Kullanıcının rolüne göre izinler tanımlanır ve bir işleme izin verilip verilmediği kontrol edilir.

### 2. Use Parameterized Queries: (Parametrelili Sorgular Kullanmak)

**Nedir?:** Kullanıcı girişlerini sorgunun içine doğrudan ekleyerek değil, yer tutucular kullanarak yapılır.

**Nasıl Yapılır?:** SQL sorgularında kullanıcı verileri için yer tutucular{} kullanılır ve bu veriler güvenli bir şekilde sorguya eklenir. Bu, **SQL injection**'ı önler.

### 3. Proper Session Management: (Doğru Oturum Yönetimi)

**Nedir?:** Oturum sürelerinin izlenmesi ve süresi dolmuş oturumun kapatılmasıdır.

**Nasıl Yapılır?:** Oturum başlatılır, son aktivite zamanı kaydedilir. Belli süre geçmiş ve hala aktivite yoksa oturum kapatılır.

### 4. Use Secure Coding Practices: (Güvenli Kodlama Uygulamaları Kullanmak)

**Nedir?:** Kodun güvenliği için kullanıcı giriş yerlerini temizlemek ve kodlama standartlarına uymaktır.

**Nasıl Yapılır?:** Kullanıcıdan gelen veriler temizlenir ve şifreler güvenli bir şekilde hashlenir (örneğin, md5 yerine password\_hash kullanılır).

## 2. Cryptographic Failures

### Nedir?

Şifreleme yada şifre çözmedeki zayıflıklar bu zafiyeti oluşturur. Bu hatalar yüzünden hassas bilgiler ifşa olabilir. Anahtar yönetiminin iyi olmaması, zayıf şifreleme algoritmaları veya uygulamada hatalar buna sebep olabilir. Cryptographic Failures, genellikle aşağıdaki durumları kapsar:

### Zayıf Şifreleme Algoritmaları:

**Açıklama:** Eski veya zayıf şifreleme algoritmaları, verilerin kolayca şifresini çözülmesine neden olabilir. Örneğin, RC4 gibi eski algoritmalar modern saldırılara karşı zayıftır.

**Örnek:** 2014'te Heartbleed açığı, OpenSSL kütüphanesindeki bir hata nedeniyle kullanıcı verilerini ifşa etti.

### Zayıf Anahtar Yönetimi:

**Açıklama:** Şifreleme anahtarlarının zayıf yönetimi, anahtarların güvenliğinin tehlikeye atılmasına neden olabilir. Anahtarların düz metin olarak saklanması veya aynı anahtarın birden fazla sistemde kullanılması bu tür hataların örneklerindendir.

**Örnek:** Equifax veri ihlali, yetersiz anahtar yönetimi nedeniyle büyük miktarda kişisel verinin ifşasına yol açtı.

### Uygulama Hataları:

**Açıklama:** Kriptografik sistemlerdeki küçük uygulama hataları, büyük güvenlik açıklarına neden olabilir. Yanlış algoritma kullanımı veya hatalı yapılandırma bu kategoriye girer.

**Örnek:** Android cihazlarda rastgele sayı üretme hatası, kriptografik anahtarların tahmin edilmesini kolaylaştırdı.

### Çözüm ve Önleyici Tedbirler

- Güçlü Şifreleme Algoritmaları Kullanın:** Güncel olan ve güvenlik açık testi yapılmış şifreleme algoritmaları kullanmak gerekir. AES ve RSA gibi modern algoritmalar tercih edilmelidir.
- Uygun Anahtar Yönetimi:** Anahtarlar string olarak tutulmamalı ve her sistem için farklı anahtar kurgulanmalıdır.

3. **Kriptografiyi Doğru Şekilde Uygulayın:** Kriptografinin düzgün bir şekilde yapılandırıldığından emin olunmalıdır.
4. **Kriptografik Sistemleri Düzenli Olarak İnceleyin:** Kriptografik sistemlerimize düzenli testler yapmalıyız.

## 2.a) <https://tryhackme.com/r/room/owasptop102021> - Cryptographic Failures (Supporting Material 1)

### Flat-File SQLite Veritabanı İnceleme ve Hash Kırma:

#### Senaryo:

Bu makinadan example.db adında SQLite veritabanı dosyası indirildi.

Bu veritabanında kullanıcı bilgileri hashlenmiş olarak tutuluyor.

Bizim amacımızsa bu hashleri çözüp içeriğe ulaşmaktır.

#### 1. Veritabanı Dosyasının Türünü Kontrol Etme

Dosya türünün kontrolü için file komutunu kullandım

```
user@linux$ ls -l
-rw-r--r-- 1 user user 8192 Feb  2 20:33 example.db

user@linux$ file example.db
example.db: SQLite 3.x database, last written using SQLite version 3039002, file counter 1, database pages 2, cookie 0x1, schema 4, UTF-8, version-valid-for 1
```

#### 2. SQLite Veritabanına Erişim

SQLite veritabanına erişmek için sqlite3 komutunu kullandım

```
user@linux$ sqlite3 example.db
SQLite version 3.39.2 2022-07-21 15:24:47
Enter ".help" for usage hints.
sqlite>
```

Sqlite3 komutu, SQLite komut satırı arayüzüne giriş yapmamı sağladı.

#### 3. Veritabanı Yapısını İnceleme & Tablodan Veri Çekme

Tables komutuyla tabloları gördüm

```
user@linux$ sqlite3 example.db
SQLite version 3.39.2 2022-07-21 15:24:47
Enter ".help" for usage hints.
sqlite> .tables
customers
```

customers tablosunun yapısını öğrenmek için PRAGMA komutunu kullandım

Tablodaki tüm verileri çekmek için SELECT komutunu kullandım



```
sqlite> PRAGMA table_info(customers);
0|custID|INT|1||1
1|custName|TEXT|1||0
2|creditCard|TEXT|0||0
3|password|TEXT|1||0

sqlite> SELECT * FROM customers;
0|Joy Paulson|4916 9012 2231 7905|5f4dcc3b5aa765d61d8327deb882cf99
1|John Walters|4671 5376 3366 8125|fef08f333cc53594c8097eba1f35726a
2|Lena Abdul|4353 4722 6349 6685|b55ab2470f160c331a99b8d8a1946b19
3|Andrew Miller|4059 8824 0198 5596|bc7b657bd56e4386e3397ca86e378f70
4|Keith Wayman|4972 1604 3381 8885|12e7a36c0710571b3d827992f4cfe679
5|Annett Scholz|5400 1617 6508 1166|e2795fc96af3f4d6288906a90a52a47f
```

Tablo bilgilerinden dört sütun olduğunu gördüm ve bunlar

**custID, custName, creditCard ve password**

İlk satırı inceleyeceğim

İlk Satır = **“0|Joy Paulson|4916 9012 2231 7905|5f4dcc3b5aa765d61d8327deb882cf99”**

#### 4. Hash Kırma Süreci (Crackstation Kullanarak)

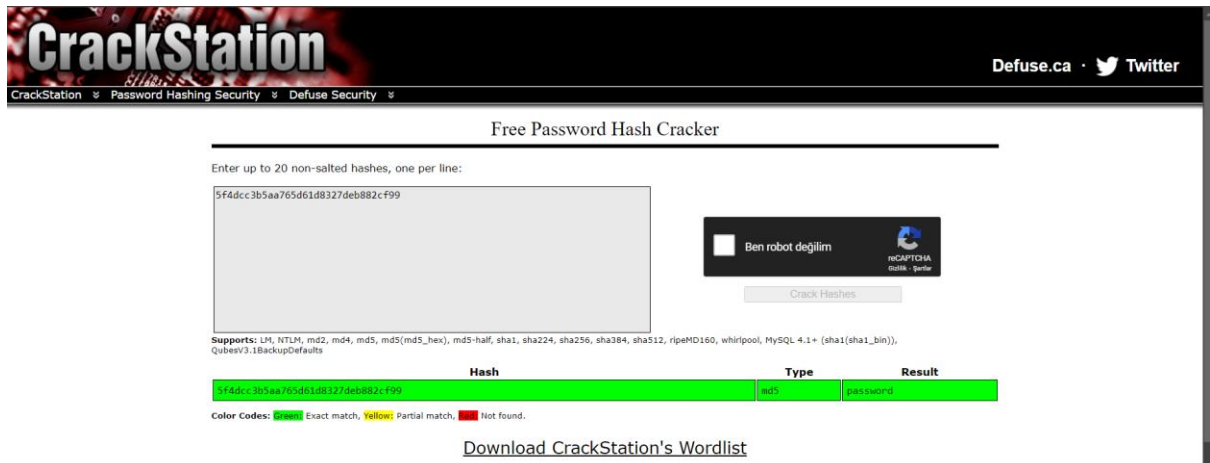
Crackstatin Web sitesinde, hashleri girebileceğin alanla mevcut. Orayı kullanacağız.



The image shows the CrackStation website's 'Free Password Hash Cracker' interface. At the top, there's a navigation bar with 'CrackStation', 'Password Hashing Security', and 'Defuse Security' links, along with 'Defuse.ca' and 'Twitter' social media links. The main heading is 'Free Password Hash Cracker'. Below it, a text input area is labeled 'Enter up to 20 non-salted hashes, one per line:'. To the right of the input area is a CAPTCHA challenge with the text 'I'm not a robot' and a 'Crack Hashes' button. Below the input area, there's a list of supported hash types: 'Supports: LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, rlpMD160, whirlpool, MySQL 4.1+ (sha1/sha2\_bin), QubesV3.1BackupDefaults'. At the bottom, there's a link to 'Download CrackStation's Wordlist'.

Kırmak istediğiniz hash'i ekleyeceğiz.

Örneğimizde, SQLite veritabanından aldığınız 5f4dcc3b5aa765d61d8327deb882cf99 hash'ini ekliyoruz.



The image shows the CrackStation website's 'Free Password Hash Cracker' interface with the hash '5f4dcc3b5aa765d61d8327deb882cf99' entered in the input area. The CAPTCHA challenge is now 'Ben robot değilim'. Below the input area, the same list of supported hash types is shown. At the bottom, there's a table with the following data:

Hash	Type	Result
5f4dcc3b5aa765d61d8327deb882cf99	md5	password

Below the table, there's a legend for color codes: 'Color Codes: Green Exact match, Yellow Partial match, Red Not found.' At the bottom, there's a link to 'Download CrackStation's Wordlist'.

5f4dcc3b5aa765d61d8327deb882cf99 hash'ini "password" olarak kırdı.

## 2.Injection Nedir?

Bu zafiyet, kullanıcının girdiği girdilerin uygulama içerisinde çalıştırılacak hale bürünmesidir ve oldukça tehlikeli bir hal alabilir.

Temelde 2 tür Injection zafiyeti vardır.

### SQL Injection:

SQL Injection, kullanıcı girdisinin doğrudan SQL sorgularına dahil edilmesiyle oluşur.

Bu saldırı, saldırganın veri tabanında izinsiz işlemler yapmasını sağlar.

Örneğin, saldırgan, kullanıcı adı ve şifre sorgusunu manipüle ederek tüm kullanıcı bilgilerine erişebilir.

### SQLmap ile SQL Injection Testi

SQLmap, web uygulamalarındaki SQL Injection zafiyetlerini otomatik olarak tespit edip istismar edebilen bir araçtır.

Bu araç, SQL sorgularını manipüle ederek veritabanı sunucusuna erişim sağlamayı ve veritabanı üzerinde çeşitli işlemler yapmayı mümkün kılar. Örneğin:

```
root@kali:~# sqlmap -u "http://testphp.vulnweb.com/index.php" --dbs
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
[*] starting @ 15:31:39 /2024-08-28/
[15:31:40] [INFO] testing connection to the target URL
[15:31:40] [INFO] checking if the target is protected by some kind of WAF/IPS
[15:31:40] [INFO] testing if the target URL content is stable
[15:31:41] [INFO] target URL content is stable
[15:31:41] [CRITICAL] no parameter(s) found for testing in the provided data (e.g. GET parameter 'id' in 'www.site.com/index.php?id=1'). You are advised to rerun with '--forms --crawl=2'
[*] ending @ 15:31:41 /2024-08-28/
```

### Komut=

sqlmap -u <http://testphp.vulnweb.com/index.php> --dbs

-u, Hedef URL

--dbs, veritabanlarını listeler

Bu komut, id parametresinin SQL Injection'a karşı zafiyet taşıyıp taşımadığını kontrol eder ve eğer varsa, mevcut veritabanlarını listeler.

### SQL Injection'dan Korunma Yöntemleri

SQL Injection zafiyetinden korunmak için en iyi yöntem, kullanıcılardan gelen girdilerin güvenliğini sağlamaktır.

SQL Injection'dan korunma teknikleri vardır.

**Prepared Statements** : SQL sorgularını dinamik olarak birleştirmek yerine, sabit kalıplar oluşturup, kullanıcı girdilerini bu kalıplara yerleştirme işlemidir. Yani veriler sorguya sonradan eklenmektedir.

**Stored Procedures** : Veritabanı üzerinde önceden tanımlanmış ve saklanmış SQL komutlardır. (**PROCEDURE** yapısı buna bir örnektir) Yani bir şey çalıştıracağı zaman ilgili PROCEDURE çalıştırılır her defasında dinamik olarak sorgu yollamayız.

**Input Validation** : Kullanıcıdan alınan tüm girdiler, özel karakterler veya SQL komutları içerip içermediğine göre kontrol edilip zararlı içeriklerin filtreleneip alınmadığı yapılanmadır.

### **Command Injection:**

Saldırganın web uygulamasından yada sisteminden faydalanıp, işletim sisteminde kod çalıştırmasına olanak sağlayan zafiyettir.

### **Command Injection Testi için Araçlar**

**Burp Suite**: Web uygulama testleri için yaygın olarak kullanılan bu araç, komut enjeksiyonu testleri için de kullanılabilir.

**Commix**: Özellikle komut enjeksiyonu zafiyetlerini otomatik olarak tespit etmek ve istismar etmek için geliştirilmiş bir araçtır.

### **Komut Enjeksiyonundan Korunma Yöntemleri**

**Input Validation**: Kullanıcıdan gelen tüm girdiler, **Command Injection**'a karşı doğrulanmalı ve filtrelenmelidir.

**Parametrized Queries**: Kullanıcının girdileri doğrudan kullanılmamalı, bunun yerine parametreler kullanılmalıdır.

**Least Privilege**: Uygulamanın çalıştığı kullanıcı hesabının ayrıcalıkları minimumda tutulmalı, sadece gerekli işlemler için yetki verilmelidir.

## **3.a ) <https://tryhackme.com/r/room/owasptop102021> - 3.1.**

### **Command Injection**

#### **Senaryo:**

Uygulamalarını geliştirmek için **Linux**'taki **cowsay** komutunu kullanmaya karar vermişler.

Ancak, web uygulamalarının güvenlik açıkları olduğunun farkında değiller.

Uygulama, sunucunun konsoluna doğrudan komut gönderen **passthru** işlevini kullanarak çalışıyor ve bu da **Command Injection** saldırılarına karşı savunmasız hale getiriyor.

```
<?php
if (isset($_GET["mooring"])) {
    $mooring = $_GET["mooring"];
    $cow = 'default';

    if(isset($_GET["cow"]))
        $cow = $_GET["cow"];

    passthru("perl /usr/bin/cowsay -f $cow $mooring");
}
?>
```

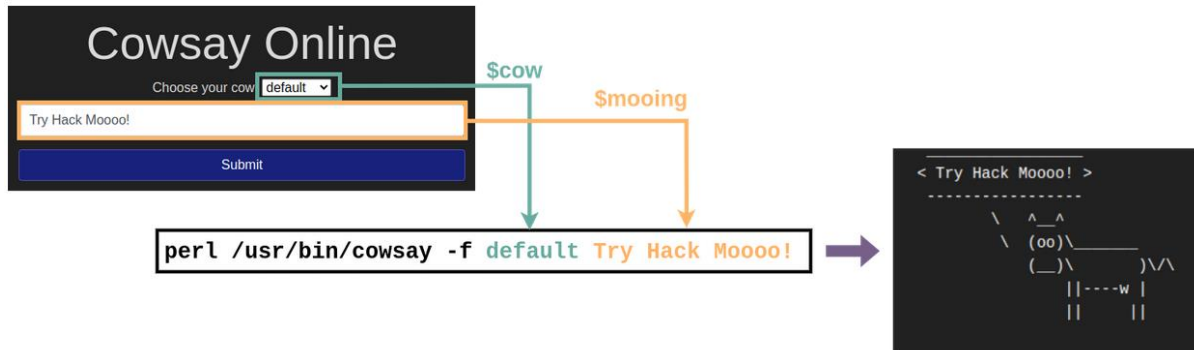
### Açıklama:

Kullanıcıdan gelen **mooring** parametresi, doğrudan **\$mooring** değişkenine atanır.

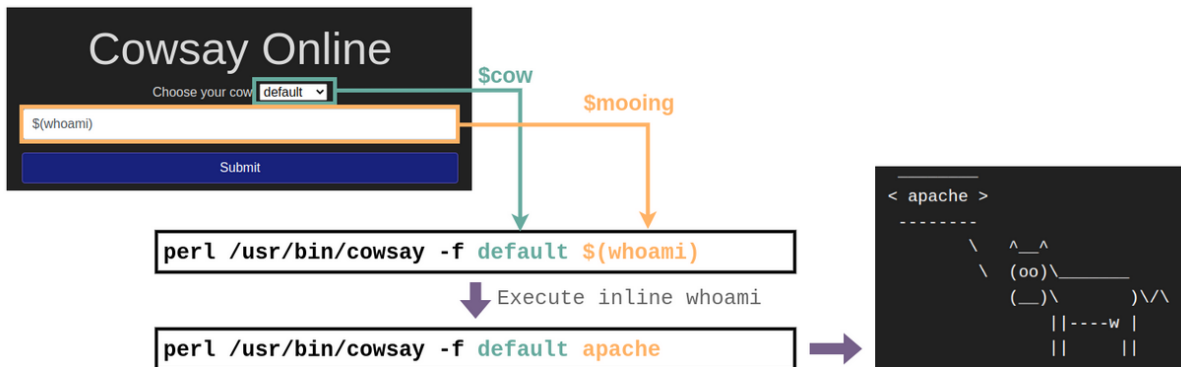
Eğer **cow** parametresi ayarlanmışsa, bu değer **\$cow** değişkenine atanır.

Son olarak, **passthru** fonksiyonu kullanılarak **cowsay** komutu çalıştırılır ve kullanıcıdan gelen parametreler komuta eklenir.

**Zafiyet ve Sömürü:** Bu kod parçası, doğrudan kullanıcı girdisini bir komuta eklediği için komut enjeksiyonuna açıktır. Saldırgan, mooring ya da cow parametrelerine ekleyeceği zararlı komutlar ile sunucunun işletim sistemi komutlarını çalıştırabilir. Bu, saldırganın sunucudaki dosyaları listelemesine, içeriklerini okumasına, sistem bilgilerini toplamasına ve daha birçok zararlı işlem gerçekleştirmesine olanak tanır.



(Normal Kullanım şekli)



(Sömürülen Kullanım şekli)

whoami

id

ifconfig/ip addr  
uname -a  
ps -ef komutları da denenebilir.

**Soru 1:** Web sitesinin kök dizininde hangi garip metin dosyası var?

**Cevap :** drpepper.txt

**Soru 2:** Kök olmayan/hizmet dışı/arka plan programı olmayan kaç kullanıcı var? (How many non-root/non-service/non-daemon users are there?)

İstedğimiz şey, sistemde kök kullanıcı, hizmet, veya arka plan programı olmayan kullanıcıların sayısını bulmak.

Bu tür kullanıcılar genellikle günlük görevleri yerine getiren normal kullanıcılar veya sistemde yerleşik olmayan kullanıcılar olabilir.

# Cowsay Online

Choose your cow:

Submit

```
/ root:x:0:0:root:/root:/bin/ash \
| bin:x:1:1:bin:/bin:/sbin/nologin |
| daemon:x:2:2:daemon:/sbin:/sbin/nologin |
| adm:x:3:4:adm:/var/adm:/sbin/nologin |
| lp:x:4:7:lp:/var/spool/lpd:/sbin/nologi |
| n_sync:x:5:0:sync:/sbin:/bin/sync |
| shutdown:x:6:0:shutdown:/sbin:/sbin/shu |
| tdown halt:x:7:0:halt:/sbin:/sbin/halt |
| mail:x:8:12:mail:/var/mail:/sbin/nologi |
| n |
| news:x:9:13:news:/usr/lib/news:/sbin/no |
| login |
| uucp:x:10:14:uucp:/var/spool/uucppublic |
| :/sbin/nologin |
| operator:x:11:0:operator:/root:/sbin/no |
| login |
| man:x:13:15:man:/usr/man:/sbin/nologin |
| postmaster:x:14:12:postmaster:/var/mail |
| :/sbin/nologin |
| cron:x:16:16:cron:/var/spool/cron:/sbin |
| /nologin |
| ftp:x:21:21::/var/lib/ftp:/sbin/nologin |
| sshd:x:22:22:sshd:/dev/null:/sbin/nolog |
| in |
```

**Kod Analizi:** \$(cat /etc/passwd)

/etc/passwd dosyası, Linux/Unix sistemlerinde kullanıcı hesap bilgilerini içerir. Her satır bir kullanıcıyı temsil eder ve şu formatta bilgiler içerir:

**kullanıcı\_adı :x:kullanıcı\_id:grup\_id:kullanıcı\_ismi:kullanıcı\_dizini :kabuk**  
**root :x:0:0:root :/root :/bin/ash**

### Çıktının İncelenmesi

Çıktıda, nologin ya da false kabuğuna sahip kullanıcılar, sisteme giriş yapması amaçlanmayan hizmet veya arka plan kullanıcılarıdır.

Yani nologin veya false kabuğu, bu kullanıcıların sistem üzerinde etkileşimli bir kabuk erişimine sahip olmadığını gösterir.

**(Normal Kullanıcıları Bulma:** nologin veya false kabuğu yerine bir kabuk (/bin/bash, /bin/sh vb.) olan kullanıcılar sistemde giriş yapabilen normal kullanıcılardır.)

**Cevap: 0**

### Soru 3: Bu uygulama hangi kullanıcı olarak çalışıyor?

Bir web uygulaması, bir sunucuda çalıştırılırken belirli bir kullanıcı hesabı altında çalışır. Bu kullanıcı, sunucunun üzerinde yüklü olan yazılımların çalıştırılması için atanmış bir hesap olabilir. Örneğin, Linux'ta bu kullanıcı genellikle **www-data**, **apache**, veya **nginx** gibi isimlerle anılan bir sistem hesabı olabilir.

# Cowsay Online

Choose your cow: default 

Enter your inner cow's mooing

Submit

< apache >

-----

```
\      ^__^
 \    (oo)\_______
      (__)\       )\/\
           ||----w |
           ||     ||
```

**Cevap:** Apache

## Soru 4: Kullanıcının kabuk seti nasıldır?

Kabuk (Shell) Nedir?

Kabuk, bir kullanıcıya komutlar vermek için kullanılan bir arayüzdür.

Örneğin, yaygın kabuklar arasında **bash**, **sh**, **zsh**, **csch** bulunur.

Her kullanıcının sistemde tanımlı bir kabuğu vardır, ve bu bilgi **/etc/passwd** dosyasında saklanır.

# Cowsay Online

Choose your cow:

Submit

```
/ apache:x:100:101:apache:/var/www:/sbin/ \
\nologin /
-----
\      ^__^
\      (oo)\_______
      (____)\       )\/\
              ||----w |
              ||     ||
```

**Cevap:** /sbin/nologin

(Kullanıcının kabuğu /sbin/nologin olarak ayarlanmış. Bu, kullanıcının doğrudan terminal oturumu açmasını engeller. Kısacası, bu kullanıcıya doğrudan erişim sağlanamaz.)

**Soru 5: Alpine Linux'un hangi sürümü çalışıyor?**

# Cowsay Online

Choose your cow:

Submit

```
< 3.16.0 >
-----
\      ^__^
\      (oo)\_______
      (____)\       )\/\
              ||----w |
              ||     ||
```

Bu komut, /etc/alpine-release dosyasındaki sürüm bilgisini ekrana yazdırır. **Cevap:** 3.16