SQL Injection

SQL meta karakterleri

```
--Select all:
SELECT * FROM Customers;
```

```
SELECT * FROM Customers -- WHERE City='Berlin';
```

```
SELECT City, Country FROM Customers
WHERE Country='Germany'
UNION
SELECT City, Country FROM Suppliers
WHERE Country='Germany'
ORDER BY City;
```

SQL Injection

Kullanıcıdan gelen data'nın tam ya da kısmi olarak SQL sorgusunda doğrudan kullanılması sonucunda oluşur

```
String query = "SELECT * FROM accounts WHERE username='" + request.getParameter("id") + "'";
```

```
http://example.com/app/accountView?id=' or '1'='1
```

```
Query HQLQuery = session.createQuery("FROM accounts WHERE custID='" +
request.getParameter("id") + "'");
```

SQL Injection

Uygulama

Login Bypass

UNION Based SQL Injection

SQL Injection

Önlemler

- Prepared Statement
- Stored Procedures
- Girdi Denetimi

```
// This should REALLY be validated too
String custname = request.getParameter("customerName");
// Perform input validation to detect attacks
String query = "SELECT account_balance FROM user_data
WHERE user_name = ?";
PreparedStatement pstmt =
connection.prepareStatement(query);
pstmt.setString(1, custname);
ResultSet results = pstmt.executeQuery();
```

07

Kimlik Doğrulama

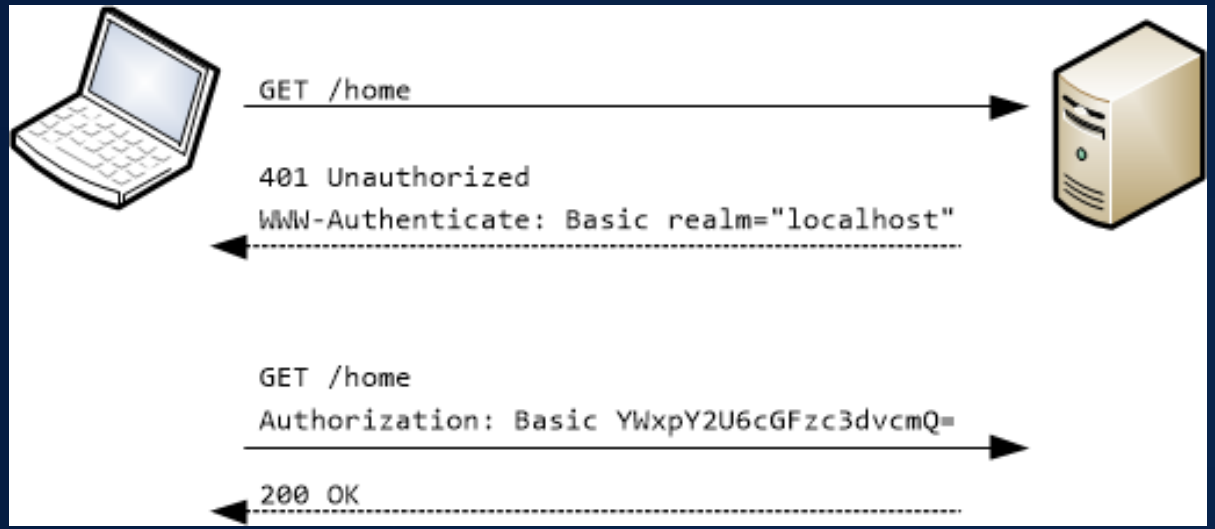
Kimlik Doğrulama
Kimlik Yöntemleri
MFA
Captcha
Kimlik Doğrulama Sistemi Saldırıları

Kimlik Doğrulama

Tarafların gerçekten tanımlı taraflar olup olmadıklarını kontrol etmek için uygulamalarda kimlik doğrulama yapılır.

- Kullanıcının, uygulamaya kendini tanıtması
- Uygulamanın, uygulamaya kendini tanıtması

Basic Authentication



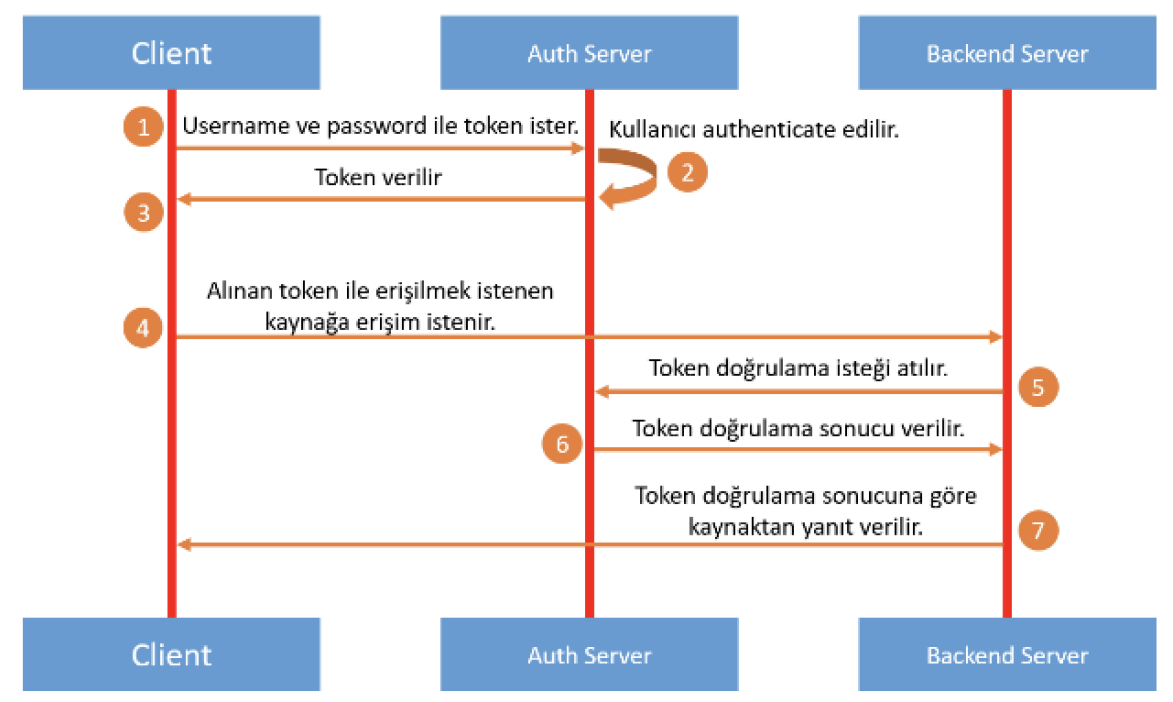
YWxpY2U6cGFzc3dvcmQ=

alice:password

Token Authentication

- Username – password bilgisi karşılığında token alınır.
- Kimliklendirme alınan token ile sağlanır.
- Özellikle API çağrıları için ideal yapı sunar.
- Çok daha esnektir.
- Oturum sonlandırma ve oturum yönetimi vardır.

Token Authentication

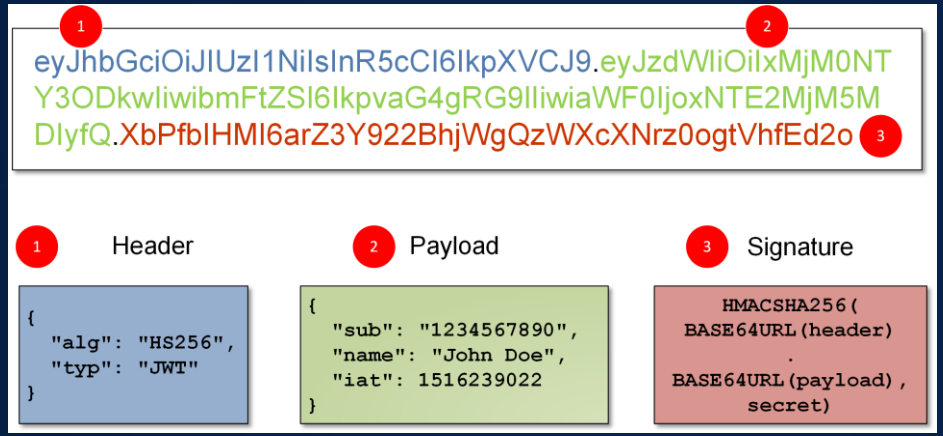


Token Authentication

JWT

JWT (JSON Web Token), JSON formatında veri içerebilen token yapılarıdır.

3 ana bölümden oluşur.



https://research.securitum.com/jwt-json-web-token-security/

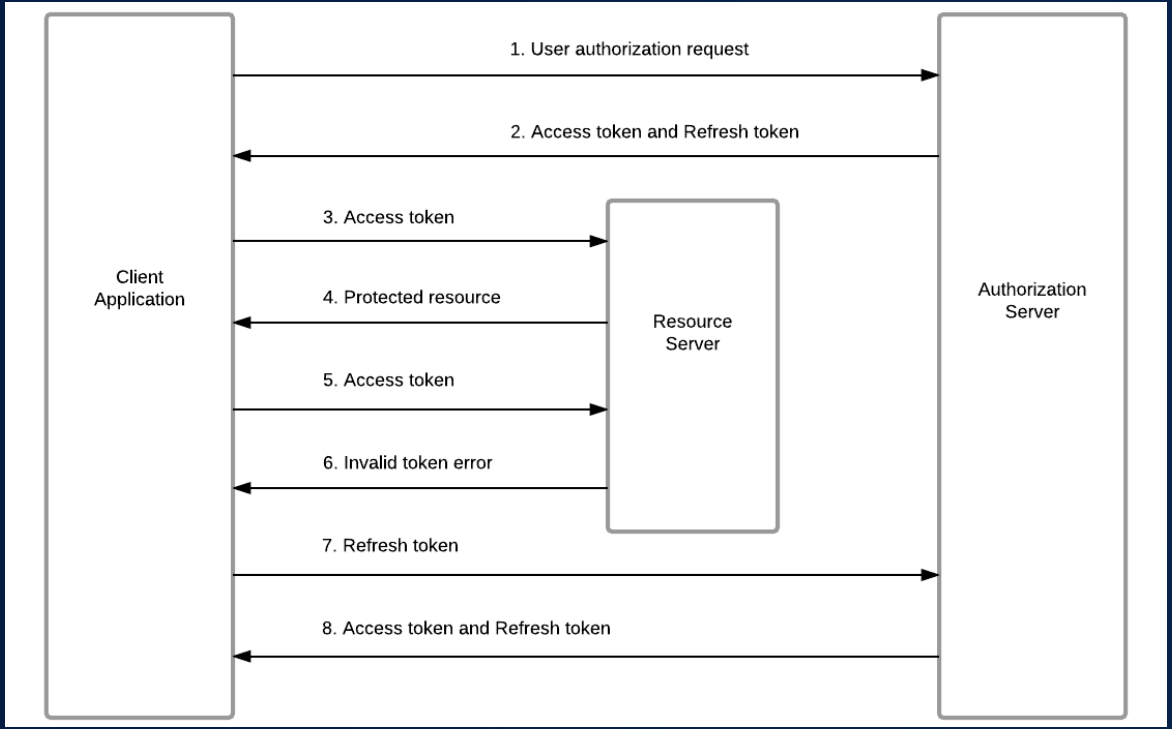
Veri taşıması kolaylık sağlaması ile birlikte bazı risklere neden olabilir.

İçerisinde hassas veri taşınmamalıdır.

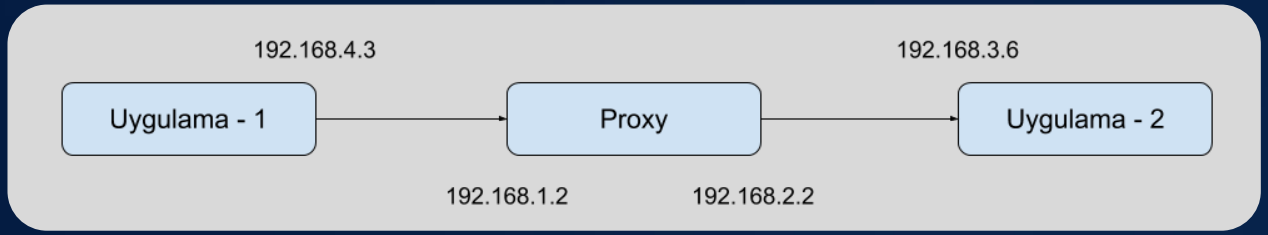
Tokenlar sadece süresi dolduğunda geçersiz olur. Kullanıcı kendi tokenını iptal edemediği için oturum çalınması durumunda risk oluşabilir.

Token Authentication

Access Token ve Refresh Token



IP Kısıtlama



getHeader[“x-forwarded-for”];

192.168.4.3, 192.168.1.2, 192.168.2.2

getIP();

192.168.2.2

```
GET / HTTP/1.1
User-Agent:
Accept: */*
Postman-Token:
Host: .
Accept-Encoding: gzip, deflate
Connection: close
X-Originating-IP: 127.0.0.1
X-Forwarded-For: 127.0.0.1
X-Remote-IP: 127.0.0.1
X-Remote-Addr: 127.0.0.1
```

MFA
(Multi-factor Authentication)

Bir kullanıcının bir sistemde kimlik doğrulaması yapmak için birden fazla kanıt türü sunması gereken yapıdadır.

Kritik işlemler içeren uygulamalarda kullanılır.

Bu kanıt türlerini aşağıdaki gibi belirtebiliriz

- Bildiğiniz bir şey (Parola, PIN, güvenlik soruları)
- Sahip olduğunuz bir şey (Donanım veya yazılım tokenları, sertifika, email, SMS)
- Sizden bir şey (Parmak izi, yüz tanıma, göz taraması)
- Lokasyon (Kaynak IP bilgileri, coğrafi konum bilgileri)

https://www.dignited.com/30668/configure-two-factor-authentication-before-you-get-locked-out-of-your-own-account/

Captcha

- Turing Testi
- Önce Captcha, sonra kullanıcı adı – şifre kontrolü
- Makine tarafından çözülemeyecek kadar karmaşık
- Genellikle kullanıcı oluşturma, şifre resetleme, kullanıcı silme, yorum yazma vb. alanlarda kullanılır.

Kimlik Doğrulama Sistemi Saldırıları

- Injection kontrolü eksikliği
- Kaba kuvvet (Brute Force)
- Varsayılan / basit kullanıcı adı şifre kullanımı
- TLS kullanılmaması

Kimlik hırsızlığı
Bilgi ifşası
Yetkisiz Erişim

Uygulama

Brute Force Saldırıları

Kimlik Doğrulama Sistemi Saldırıları

Captcha Zafiyetleri

- Görüntü işleme ile çözülememeli
- Captcha için olan görüntü seti dar olmamalı
- Captcha tek seferlik kontrolde geçerli olmalıdır Captcha Replay saldırıları
- Farklı fontlar ve karakter bozma yöntemleri içermelidir.

Kimlik Doğrulama Sistemi Saldırıları

Captcha Replay

```
j_username=[redacted]&j_password=[redacted]&captcha=YT6BQ3&captchaHash=B
3A3E912FF94F098A1EDAE1371DCEB49&checkAuth=submit
```

Girilen captcha değerinin session'dan silinmemesidir. Doğru ve yanlış her denemeden sonra captcha ve hash değeri silinmeli, geçerliliği iptal edilmelidir.

```
String captcha = request.getParameter("captcha");
c = (Captcha) session.getAttribute(Captcha.NAME);
//captcha değeri sessiondan silinir.
session.setAttribute(Captcha.Name, null);
 if (c != null && c.isCorrect(captcha)){
    // doğru resim
}
else {
    // yanlış resim
}
```

08

Oturum Yönetimi

Cookie ve Session
CSRF Zafiyeti
Dikkat Edilmesi Gereken Maddeler

Cookie ve Session

- HTTP stateless bir protokoldür
- Cookie bir kimliktir, session ise bu kimliğe ait bilgilerin tutulduğu objedir

```
POST /login HTTP/1.0
Host: example.com
...

username=test&password=Passw0rd!
```

```
HTTP/1.1 200 OK
Content-Type: text/html
Set-Cookie: SessionId=asqDd13cCqDSad123wfasf145lk15jHlk512; path=/;
```

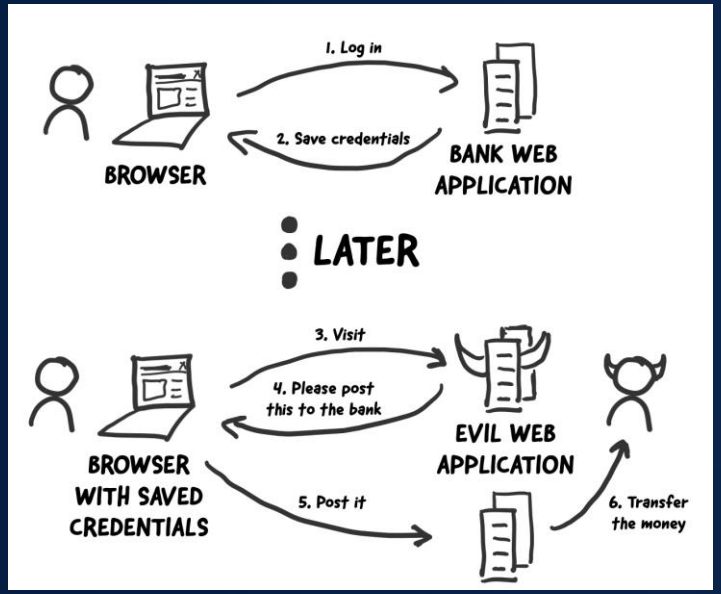
```
GET /profile HTTP/1.0
Host: example.com
Cookie: SessionId=asqDd13cCqDSad123wfasf145lk15jHlk512
```

Cookie Özellikleri

- Domain, Path
- HTTPOnly, Secure
- Expires, Max-age
- SameSite

CSRF
(Cross-site Request Forgery)

CSRF (Cross-site Request Forgery), A uygulaması üzerinden, B uygulaması üzerinde oturumu açık olan kullanıcının, bilgisi dahilinde olmadan işlem yaptırılabilmesine olanak tanıyan bir güvenlik açığıdır.



https://medium.com/tresorit-engineering/modern-csrf-mitigation-in-single-page-applications-695bcb538eec

CSRF
(Cross-site Request Forgery)

```
POST /email/change HTTP/1.1
Host: example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 30
Cookie:
session=yvthwsztyeQkAPzeQ5gHgTvlyxHfsAfE

email=haktan@example.com
```

example.com üzerinde mail değiştirme isteği

```
<html>
 <body>
  <form action="https://example.com/email/change"
method="POST">
   <input type="hidden" name="email" value="hacker@evil-user.net"
/>
  </form>
  <script>
   document.forms[0].submit();
  </script>
 </body>
</html>
```

hacker.com sayfasının kaynak kodu

Uygulama

CSRF

Örnek-1

CSRF
(Cross-site Request Forgery)

CSRF token kullanılarak, sunucu tarafında kontrol edilmelidir.

```
<input type="hidden" name="csrf-token" value="CIwNZNlR4XbisJF39I8yWnWX9wX4WFoz" />
```

CSRF token, oturum başına benzersiz ve tahmin edilemez olmalıdır.

GET metodu üzerinden durum değişikliğine sebep olacak bir işlem yapılmamalıdır.

CSRF token, cookie üzerinden taşınmamalıdır.

SameSite

None, Lax ve Strict

Oturum Yönetimi

Dikkat Edilmesi Gereken Maddeler

- Kullanılan framework'ün sunduğu oturum denetimleri kullanılmalıdır. Tekerleği yeniden icat etmeye gerek yok ☺
- Oturum kimliği oluşturma, güvenilir bir sunucuda yapılmalıdır.
- Oturum kimliği, benzersiz, yeterince rastgele üretilmelidir.
- Kullanıcıya ait bilgiler, oturum kimliğinde kullanılmamalıdır.
- Cookie'lerin domain ve path bilgileri ayarlanmalıdır.

Oturum Yönetimi

Dikkat Edilmesi Gereken Maddeler

- Cookie’lerde “Secure” ve “HTTPOnly” bayrakları işaretlenmelidir.
- Her başarılı kimlik doğrulama sonrasında, yeni bir oturum oluşturulmalıdır.
- Oturumu açmadan önce bir oturum oluşturulduysa, başarılı bir şekilde giriş yapıldıktan sonra yeni bir oturum oluşturulmalıdır.
- İş gereksinimine bağlı olarak, mümkün olduğunca kısa bir oturum haraketsizlik süresi belirlenmelidir.
- Oturum etkinken bile kalıcı oturum açma işlemlerine izin verilmemeli, periyodik olarak oturum sonlandırması uygulanmalıdır.

Oturum Yönetimi

Dikkat Edilmesi Gereken Maddeler

Mümkünse, aynı kullanıcıya ait eş zamanlı oturum açılabilmesine izin verilmemelidir.

Uygulamadan çıkış yapıldığında, sunucu tarafında ilişkili oturum sonlandırılmalıdır.

Oturum tanımlayıcıları, URL'lerde, hata mesajlarında ve günlüklerde gösterilmemelidir.

Herhangi bir ayrıcalık seviyesi değişikliğinden sonra oturum kimliğini yenilenmelidir.
(Parola değişiklikleri, izin değişiklikleri gibi)

Eğer, durum bilgisinin kullanıcı tarafında saklaması
gerekiyorsa, şifreleme ve bütünlük denetimleri, sunucu tarafında kontrol edilmelidir.

09

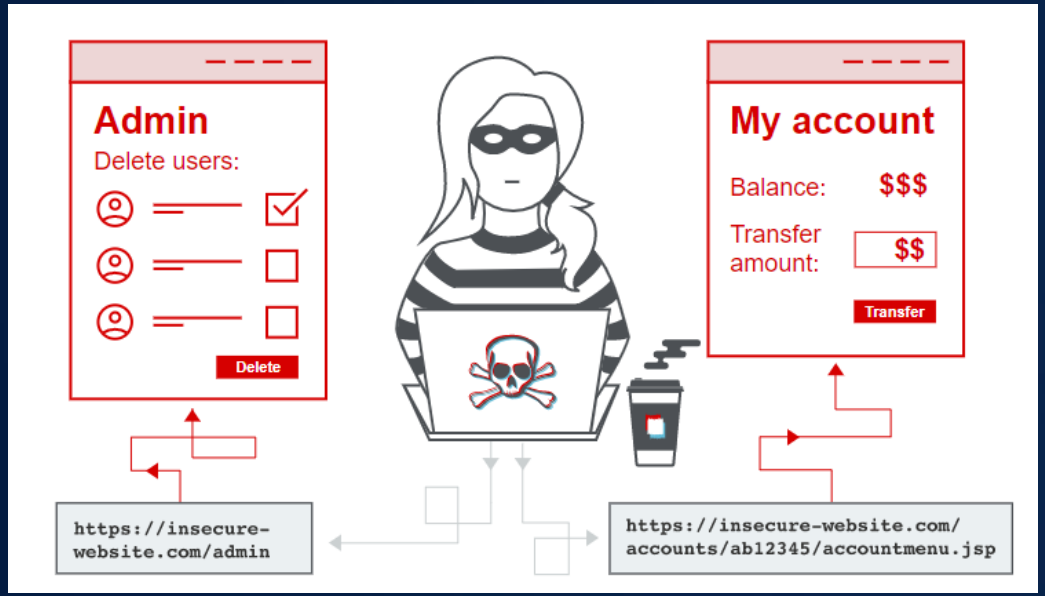
Yetkilendirme & Erişim

Yetkilendirme
Yatay & Dikey Yetki Yükseltme
Insecure Direct Object Reference (IDOR) Zafiyeti

Yetkilendirme

Yetkilendirme, kullanıcıların erişim haklarının belirlenmesi ve kontrol edilerek sadece yetkileri dahilindeki kaynaklara erişebilmesidir.

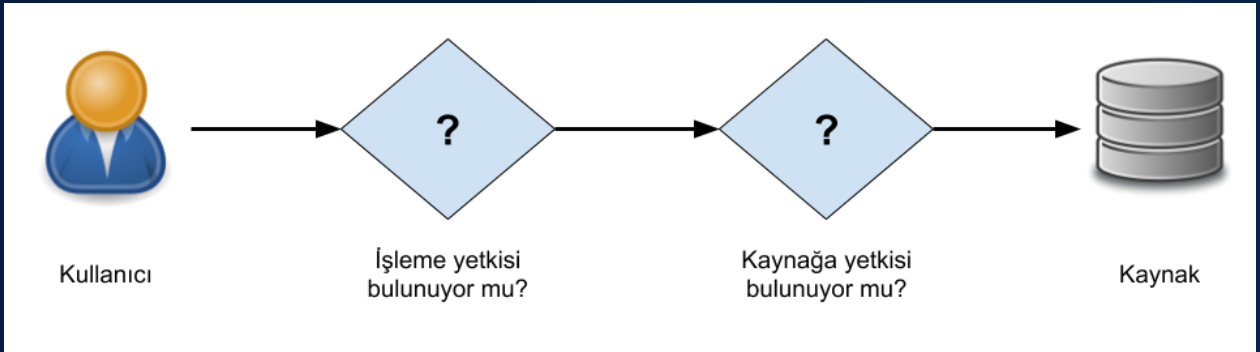
Yatay & Dikey Yetki Yükseltme



https://portswigger.net/web-security/access-control

IDOR
(Insecure Direct Object Reference)

IDOR (Insecure Direct Object Reference), yetkilendirme kontrolünü eksikliği sebebiyle meydana gelen bir güvenlik açığıdır.

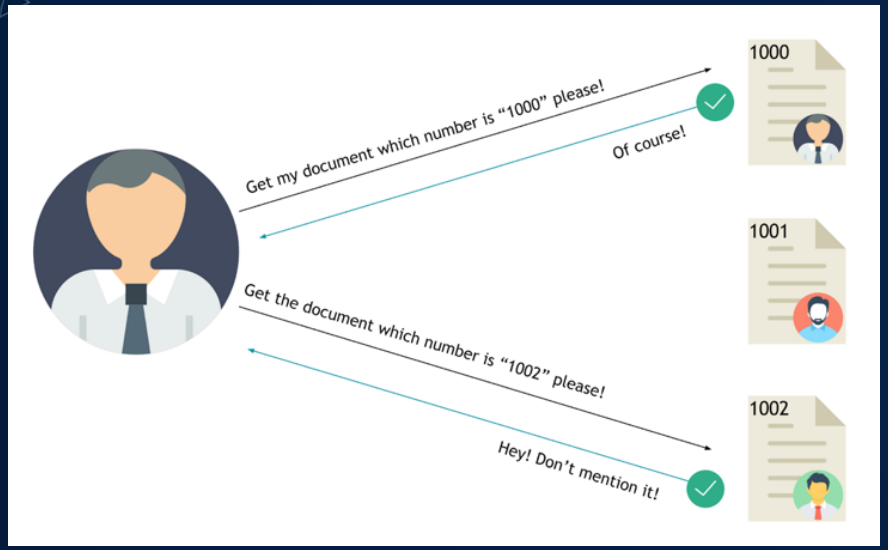



```
pstmt.setString(1, request.getParameter("acct"));

ResultSet results = pstmt.executeQuery( );
```

```
http://example.com/app/accountInfo?acct=notmyacct
```

IDOR
(Insecure Direct Object Reference)



https://example.com/previewDocument?Id=1000
https://example.com/previewDocument?Id=1002

Kullanıcı	Doküman Id
User1	1000
User2	1002

Broken Function Level Authorization

/api/users/user/myinfo
/api/admins/users/all

Uygulama

IDOR

Örnek-1
Örnek-2
Örnek-3
Örnek-4

IDOR
(Insecure Direct Object Reference)

Önlem

- Sunucu tarafında, kullanıcının yapacağı her işlem için yetki kontrolü sağlanmalıdır. İsteği yapan kullanıcının, erişmek isteğini nesneye erişim yetkisinin olup, olmadığı kontrol edilmelidir.

- Uygulama üzerindeki korunaklı kaynaklar (bağlantılar, fonksiyonlar, nesneler vs.) sadece yetkili kullanıcılar tarafından erişilebilecek şekilde kısıtlanmalıdır.

- Uygulama mekanizmaları varsayılan olarak tüm erişimleri reddetmeli, sadece uygun olan role yönelik işlemlere izin verilmelidir.