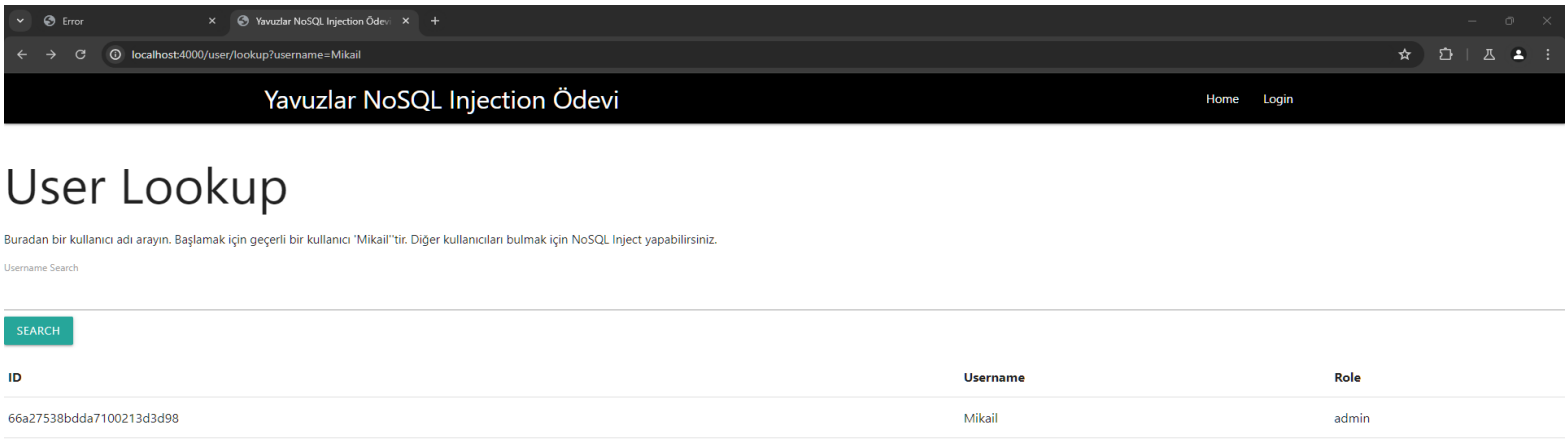
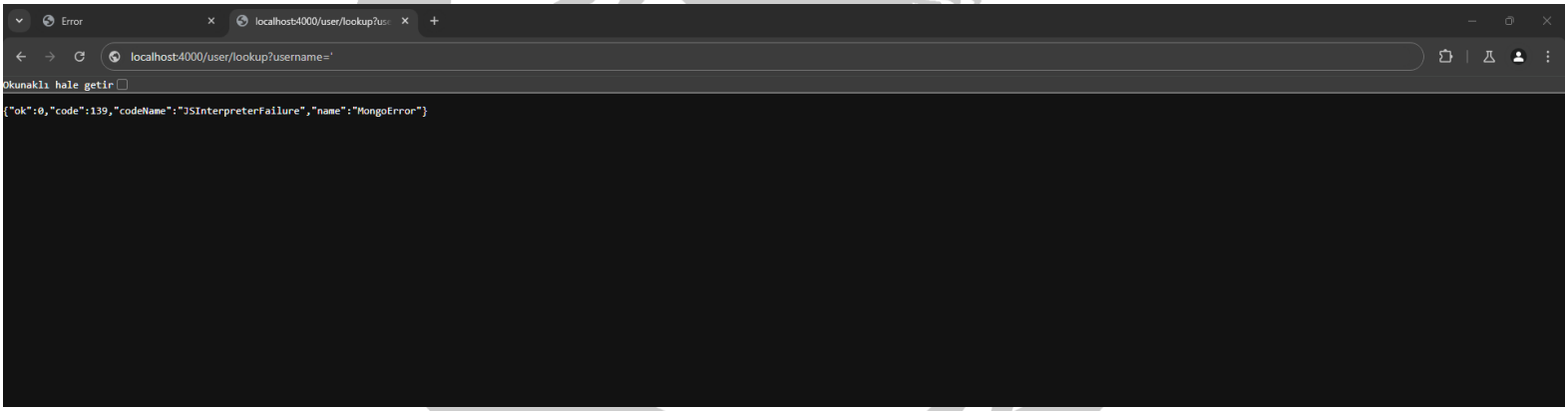
YAVUZLAR WEB GÜVENLİĞİ & YAZILIM TAKIMI
NoSQL Injection Ödevi

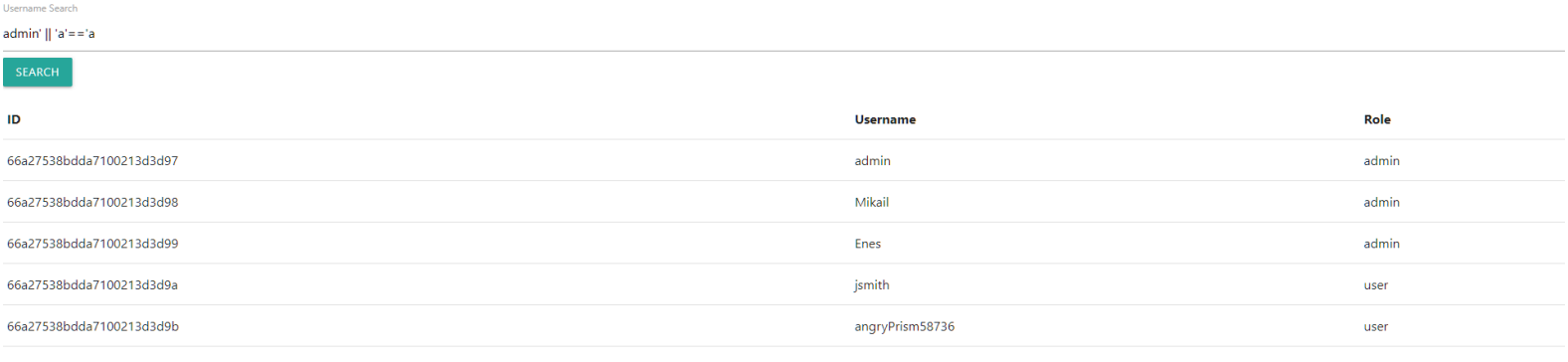
1- user/lookup



Burada gözüktüğü üzere GET metodu ile bizden bir kullanıcı adı alıyor ve ardından aldığı kullanıcı adı ile ilgili rol, kullanıcı adı, ID gibi bilgileri ekrana basıyor. Burada bir injection denemesi yapıyoruz



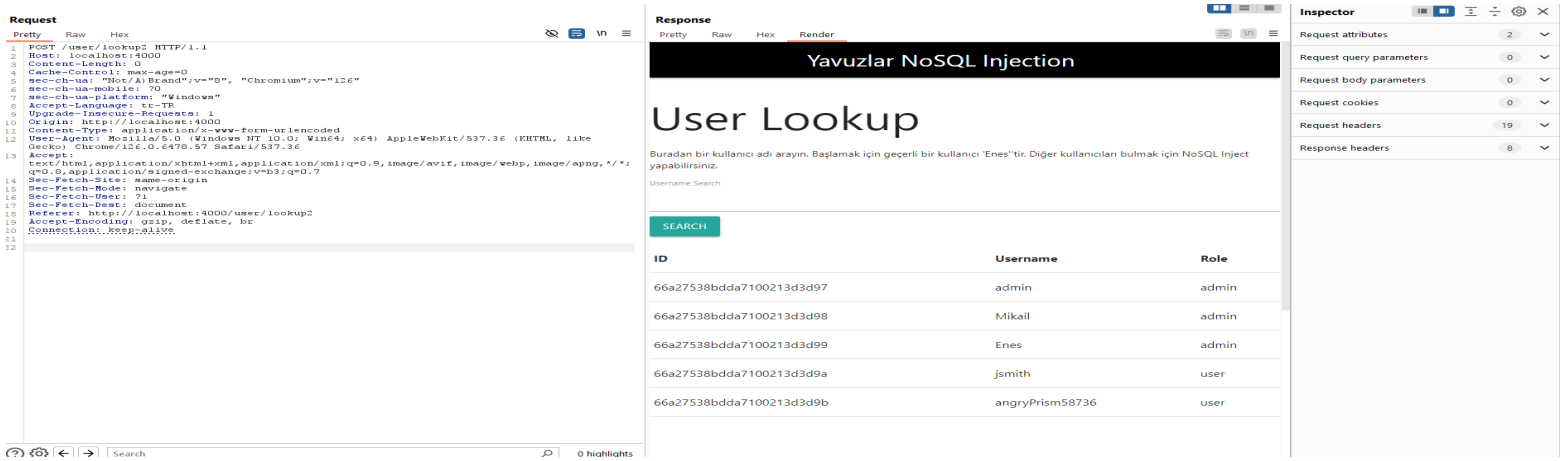
Bir MongoDB sorgusuna ' atmam ile bana bir MongoError verdi. Buradan anlyoruz ki NoSQL injection denemesi yapabiliriz.



Burada payload'n yapt, yukarda Mongo query'de gözüktüü gibi ya da sorgusu ile tüm bilgileri dökmesi. Buradaki zaafiyet, yazlmcnn bizim sorguyu bozmamz engellememesidir.

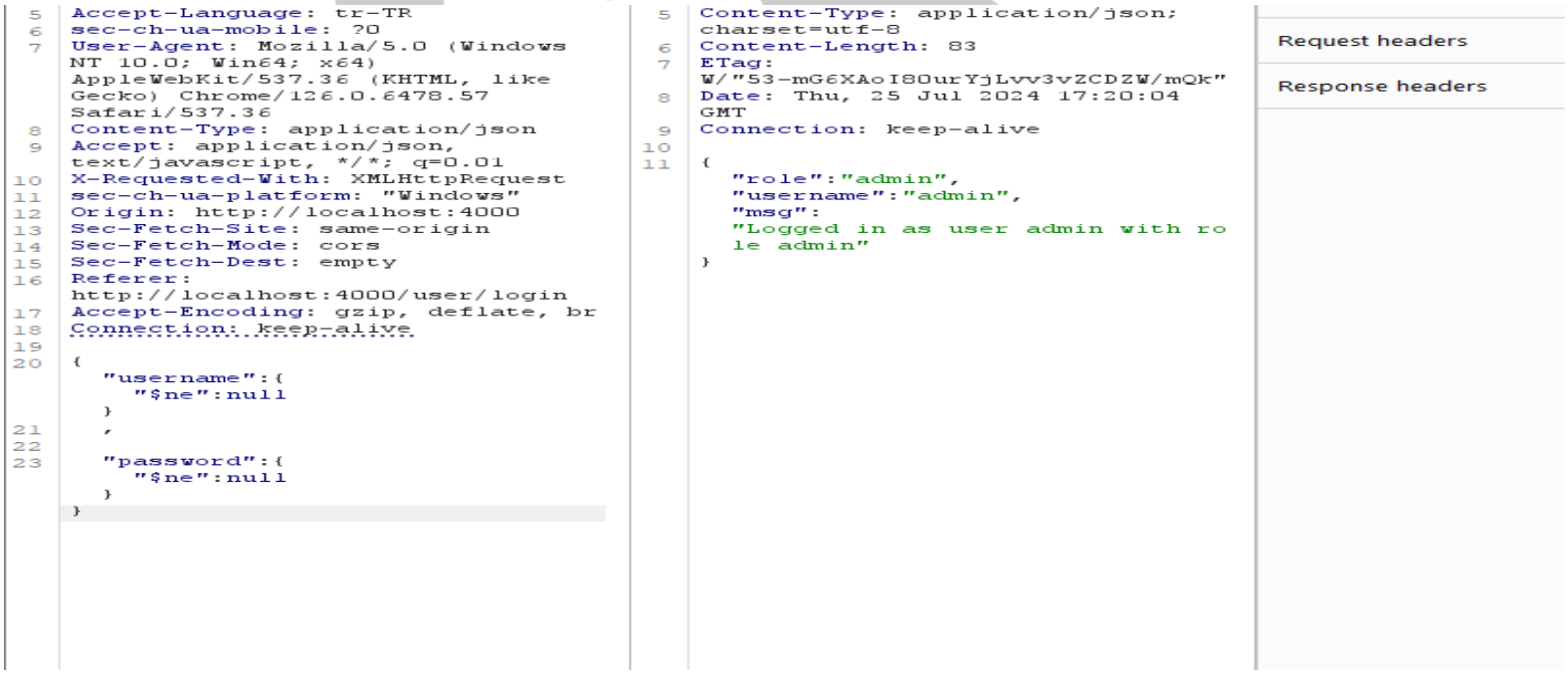
Açklama
Bu durumda, kullancdan alnan girdi dorudan sorguya dahil ediliyor ve potansiyel bir NoSQL injection zaafiyeti oluuyor. Bu tür güvenlik açklarn önlemek için kullanc girdilerini dorulamak ve uygun ekilde temizlemek gereklidir.

Birkaç denemeden sonra anlıyoruz ki gönderdiğimiz payload'ların temizlenmesi doğru bir ekilde yapılmış; fakat başka bir güvenlik açığı tespit ettik.



Buradaki zaafiyet, sorgunun kullanıcı girdisine bağlı olması ve bu girdinin eksik veya boş olması durumunda tüm veritabanı belgelerinin döndürülmesidir. Bu, özellikle kullanıcı girişleri veya yetkilendirme işlemlerinde büyük bir güvenlik açığı oluşturabilir.

4- user/login



Bu payload, aslında kullanıcı adı veya ifre alanının null olmadığı tüm belgeleri sorgular. Bu, veritabanında kullanıcı adı ve ifre alanları dolu olan tüm belgelerin döndürülmesine neden olur. Bu nedenle, herhangi bir belirli kullanıcıyı hedeflemek yerine, bu koşulları sağlayan tüm belgeler sorguya dahil edilir. Bu da veritabanındaki potansiyel olarak hassas bilgilerin açığa çıkmasına neden olabilir ve bir güvenlik açığı oluşturur. Büyük ihtimalle veri tabanında bulunan ilk kayıt admin olduğu için onun bilgileri ile giriş yaptı

YAVUZLAR WEB GÜVENLİĞİ & YAZILIM TAKIMI
NoSQL Injection Ödevi

Zafiyetlerin Kapatılması

lookup-1

```
userRoutes.route('/lookup').post(function(req, res) {
    let username = req.body.username;
    console.log("request " + JSON.stringify(username));
    if (typeof username !== 'undefined') {
        query = { $where: `this.username == '${username}'` }
        console.log("Mongo query: " + JSON.stringify(query));
        User.find(query, function (err, users) {
            if (err) {
                console.log(err);
                res.json(err);
            } else {
                console.log("Data Retrieved: " + users);
                res.json({users});
            }
        });
    }
    else {
        res.json({});
    }
});
```

MongoDB'de $where operatörü kullanarak yapılan sorgular, NoSQL Injection saldırılarına açık olabilir Kullanıcıdan gelen username değeri doğrudan sorguya eklenir. Kötü niyetli bir kullanıcı, username değerine JavaScript kodu ekleyebilir. Örneğin : username = "admin' || 1==1 //"
Sorgu u ekilde olur: {}{}{ $where: `this.username == 'admin' || 1==1 //` {}}

Çözüm:Girdi Doğrulama: Kullanıcıdan gelen girdileri doğrulayın ve geçerli karakterlerle sınırlayın ' " < > kullanımı engelliyoruz.

```
const temizsorgu = /^[a-zA-Z0-9_]+$/;
    if (temizsorgu.test(username)) {
        const query = { username: username }; // Temizlenmiş sorgu
```

lookup-2
Çözüm:Girdi Doğrulama: Kullanıcıdan gelen girdileri doğrulayın ve geçerli karakterlerle sınırlayın ' " < > kullanımı engelliyoruz.

```
        if (!/^[a-zA-Z0-9_]+$/.test(username)) {
            return res.status(400).json({ error: 'Invalid username format' });
        }
    }

    let query = {};
    if (username) {
        query.username = username;
    }
```

login

Çözüm:Girdi Doğrulama: Kullanıcıdan gelen girdileri doğrulayın ve geçerli karakterlerle sınırlayın ' " < > kullanımı engelliyoruz.

```
userRoutes.route('/login').post(function(req, res) {
    let uname = req.body.username;
    let pass = req.body.password;
    console.log("Login request " + JSON.stringify(req.body));
    let query = {
        username: uname,
        password: pass
    }
```

```
const blacklist = /[\>\<\(\)\[\]\{\}\|\;\&\'\"\\\`\~\!\@\#\$\%\^\*\+\=\_\-]/;

if (blacklist.test(cleanedUsername) || blacklist.test(cleanedPassword)) {
   return res.status(400).json({ msg: 'Invalid characters in username or password.' });
}


if (!cleanedUsername || !cleanedPassword) {
    return res.status(400).json({ msg: 'Username and password are required.' });
}
```

username ve password gibi kullanıcı girdilerini blacklist kullanarak temizleyen ve güvenli bir ekilde veri tabanına sorgu gönderen bir örnek sunuyorum

```
[
  {
    "_id": "66a27538bdda7100213d3d98",
    "username": "admin",
    "password": "2TR6uTRAuMUr5vARs9fYgdqY",
    "first_name": "",
    "last_name": "",
    "role": "admin",
    "email": "admin@yavuzlar.com",
    "__v": 0
  },
  {
    "_id": "66a27538bdda7100213d3d99",
    "username": "Mikail",
    "password": "password",
    "first_name": "Mikail",
    "last_name": "KOCADAĞ",
    "role": "admin",
    "email": "mikail@yavuzlar.com",
    "__v": 0
  },
  {
    "_id": "66a27538bdda7100213d3d9a",
    "username": "Enes",
    "password": "abc123",
    "first_name": "Enes",
    "last_name": "KORKMAZ",
    "role": "admin",
    "email": "enes@yavuzlar.com",
    "__v": 0
  },
  {
    "_id": "66a27538bdda7100213d3d9a",
    "username": "jsmith",
    "password": "SuPeRsEcR3T",
    "first_name": "John",
    "last_name": "Smoth",
    "role": "user",
    "email": "jsmith@yavuzlar.com",
    "__v": 0
  },
  {
    "_id": "66a27538bdda7100213d3d9b",
    "username": "angryPrism58736",
    "password": "L1g7tM3Up!",
    "first_name": "Gary",
    "last_name": "Jorgen",
    "role": "user",
    "email": "prismman@yavuzlar.com",
    "__v": 0
  }
]
```