

PENTESTING NOSQL DB'S WITH NOSQL EXPLOITATION FRAMEWORK



Hack In Paris June 23-27, 2014

About Me

- ◆ Independent Security Researcher
- ◆ Member @ OpenSecurity
- ◆ Currently Pursuing My Bachelors Degree (Amal Jyothi College of Engineering)
- ◆ Spoken @ a couple of Conferences
- ◆ Sleep @ Morning, Researches and Codes @ Night

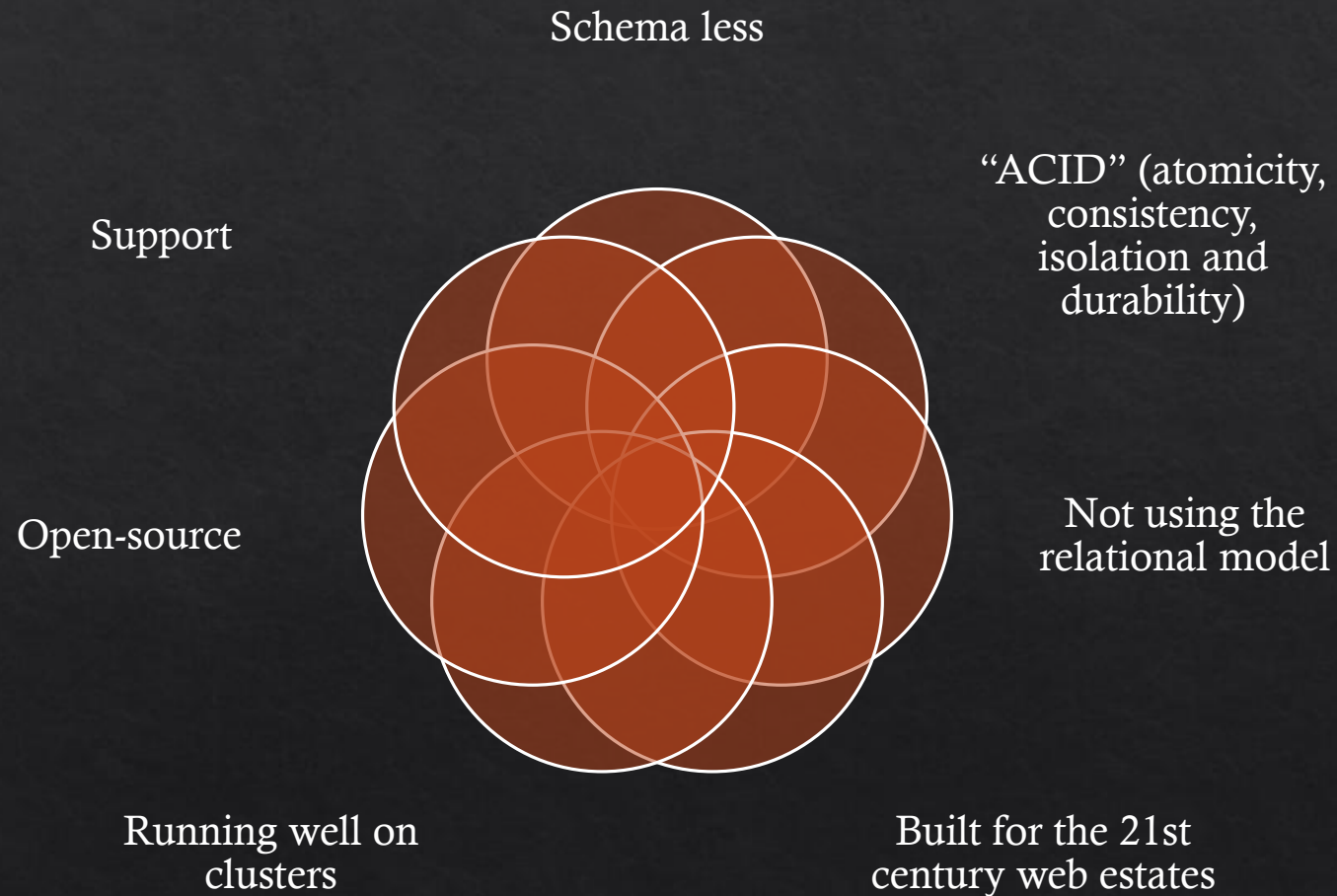


Agenda

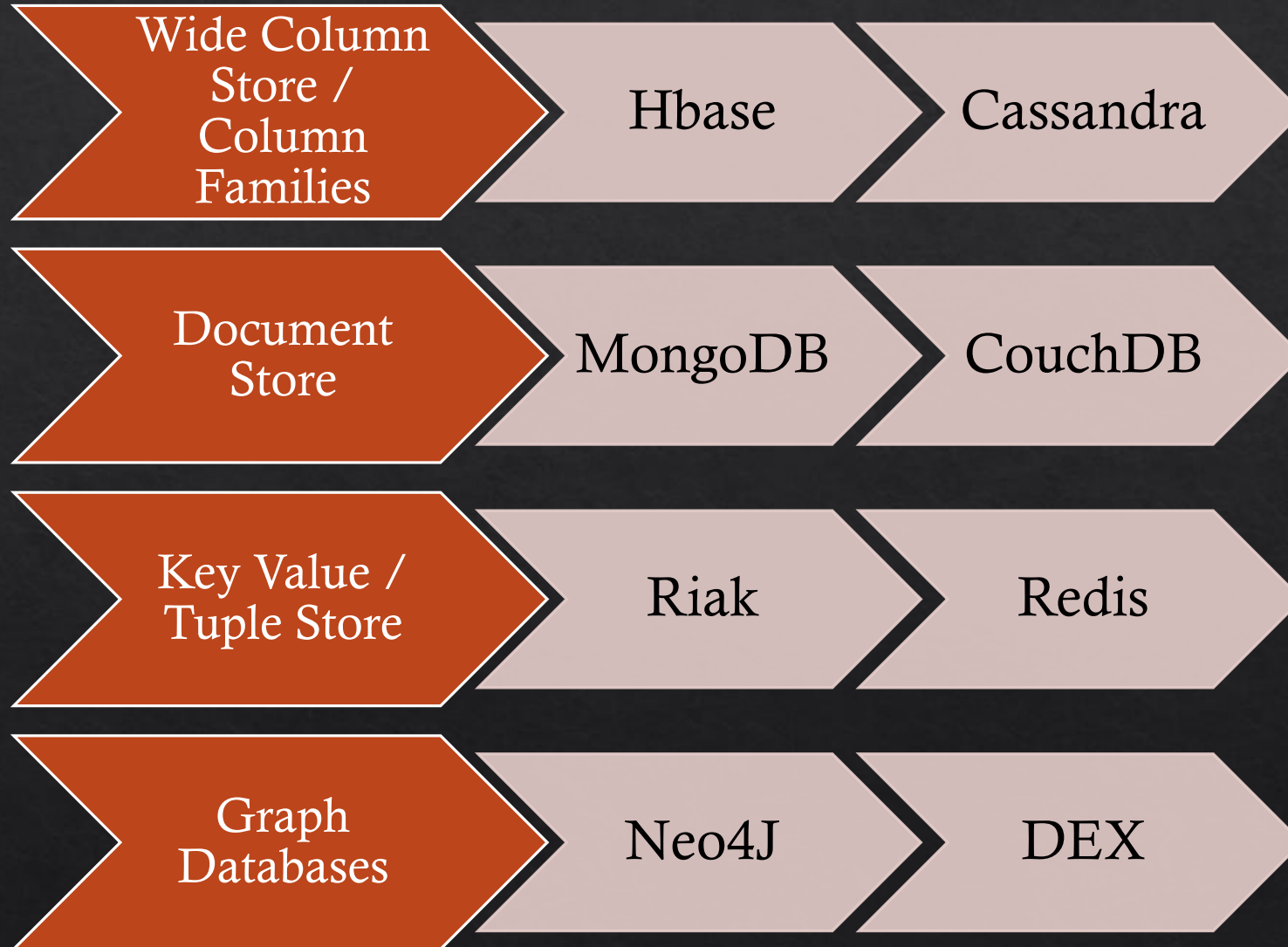
- ◇ More Emphasis Given on the Server-Client and Server Management Consoles of NoSQL DB's
- ◇ Pentesting Scenarios are given more importance.
- ◇ We will not deal with Memory Related Bugs or issues.
- ◇ NoSQL Exploitation Framework
- ◇ Demos

INTRO TO NOSQL

Key Takeovers



Mainly 4 types




NoSQL Security

You Will Love this Part 😊


SHODAN Search

Results 1 - 10 of about 84388 for port:27017



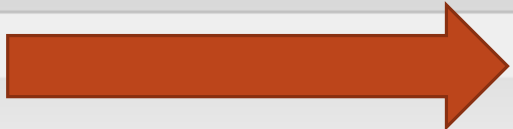
SHODAN Search

Results 1 - 10 of about 1736 for {"couchdb":"Welcome","version":""}



SHODAN Search

Results 1 - 10 of about 37413 for 6379



ABUSING API CALLS

No Proper Validation in API Calls

Developers Use them to Develop various Applications

PHP is easy to abuse for Mongo , Couch, Cassandra.

MongoDB

Mongo Trivia

Written in: C++

Main point: Retains some friendly properties of SQL. (Query, index)

Protocol: Custom, binary (BSON)

Mongod is the "Mongo Daemon" running on Port 27017 by Default

Web Interface Runs on 28017

Mongo is the Client → Mongod

Uses MongoDB Wire Protocol (TCP/IP Socket)

Data is Represented using JSON format

Mongo Architecture

Mongo Client

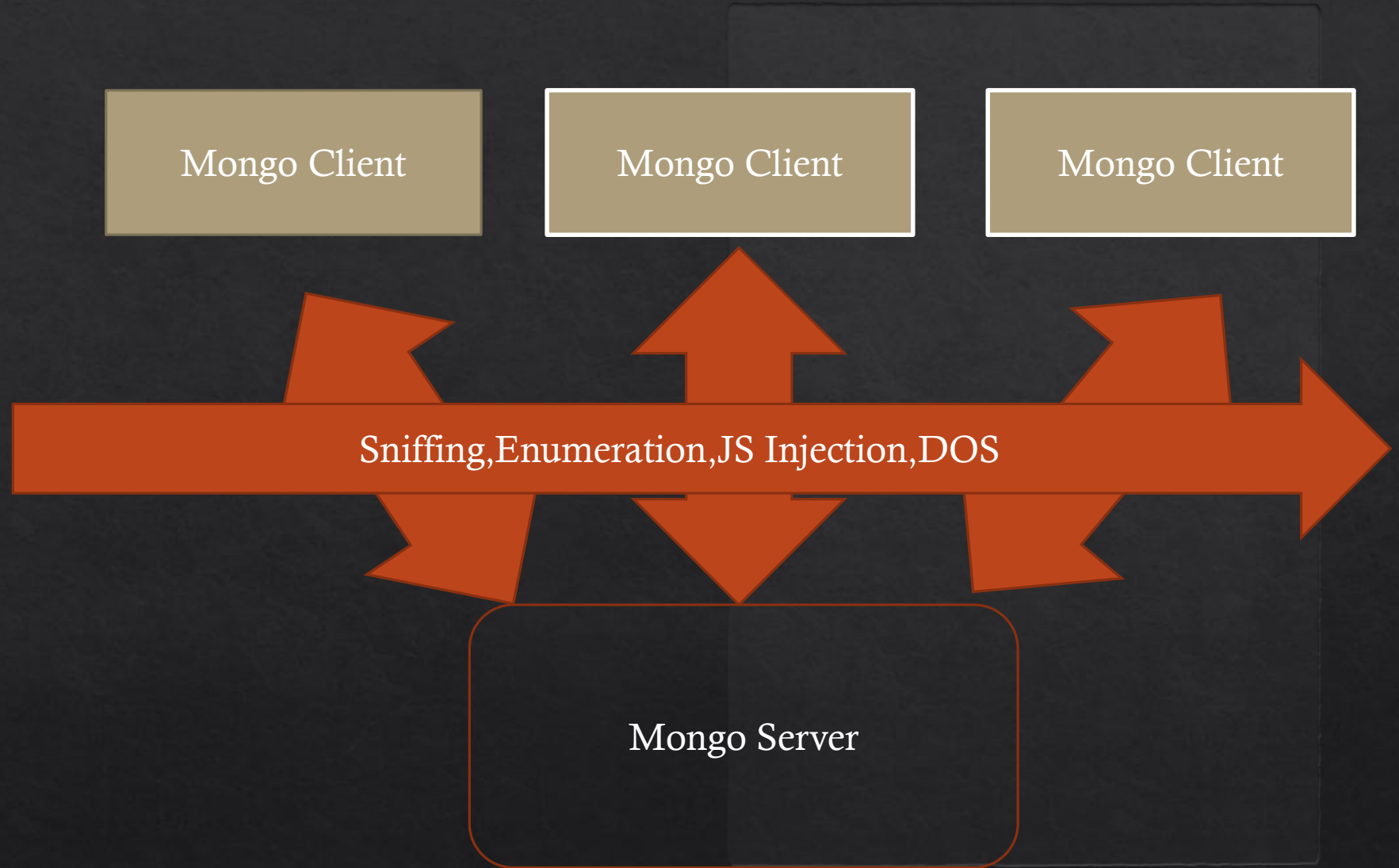
Mongo Client

Mongo Client

Mongo Server



Attackers Perspective



JS Attack Surface

Issues

JavaScript Attacks mostly used against MongoDB

Vulnerabilities Keep Popping Up

- Run command RCE

Mongo Shell Functions Purely Based on JavaScript

Possible Chances to Overwrite Functions

Resource Exhaustion

Regex Matching , plenty of JavaScript operations could be used

Mapping SQL Logical Commands to MongoDB

- and mapped to &&
- or to ||
- '=' to '=='

Saving JavaScript

Post Exploitation Phase

- ◆ Allows the attacker to write JavaScript functions and save them

- ◆ Can use for further attacks when needed rather than

- ◆ `db.system.js.save(`

```
{
```

```
  _id : "hip2014paris " ,
```

```
  value : function (x, y){ return x + y; }
```

```
}
```

```
);
```

- ◆ `db.loadServerScripts()`

Injecting JavaScript

Reference to DB in Mongo

- ◇ Mongo Functions get to refer the db object and its main functions
- ◇ An attacker who finds an Injection point could abuse this.
- ◇ Found in Versions 2.2 or less
- ◇ Mongo Patched for versions above.

Does JS Injection end here?

Timing Based Checks

- ◇ Application can be tested for time based attacks
- ◇ Mongo Shell Supports most of JavaScript functions
- ◇ `function(){ return sleep(500);}` would render the application response delayed by 5sec.
- ◇ Module added to NoSQL framework while testing for JS Injection attacks

THIS Pointer Reference

- ◇ Although mongo blocks reference to the db has ended
- ◇ Attacker could use this pointer to return objects and dump as always

Vulnerable APP DEMO

What if THIS is Blocked? Version to Rescue

- ◇ Version command by default binds to mongo instances defined functions
- ◇ So if an admin blocks the “this” pointer

- ◇ `function(){return this}`

Blocked

- ◇ `Function(){return version}` or `function(){return version()}`

- ◇ Scenario useful when developer uses \$where to evaluate js code.

Vulnerable APP DEMO

Mongo With PHP

PHP converts parameter with brackets to arrays.

- Already addressed issue in previous researches

Lets Look at Some New vectors

- \$exists
- \$type
- \$all

Vulnerable APP DEMO

Resource Exhaustion

Mongo on 32 bit environment is too easy for attackers (Max Size limit 2GB)

Use command creates arbitrary schemas on the fly

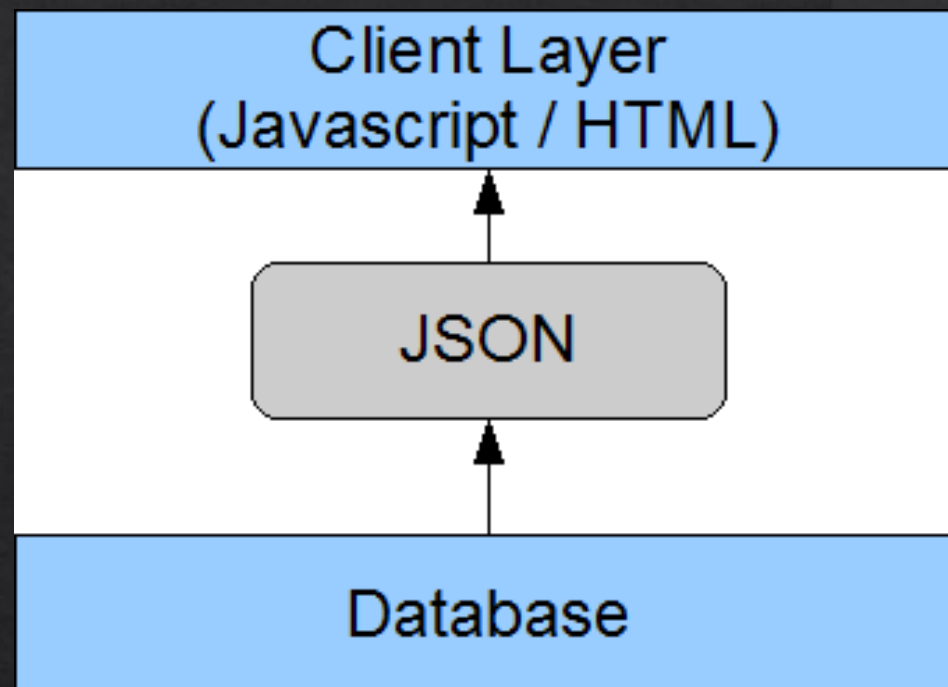
Attacker could run it continuously exhausting the disk space resource as well as memory.

```
var i=1;while(1){use i=i+1;}
```

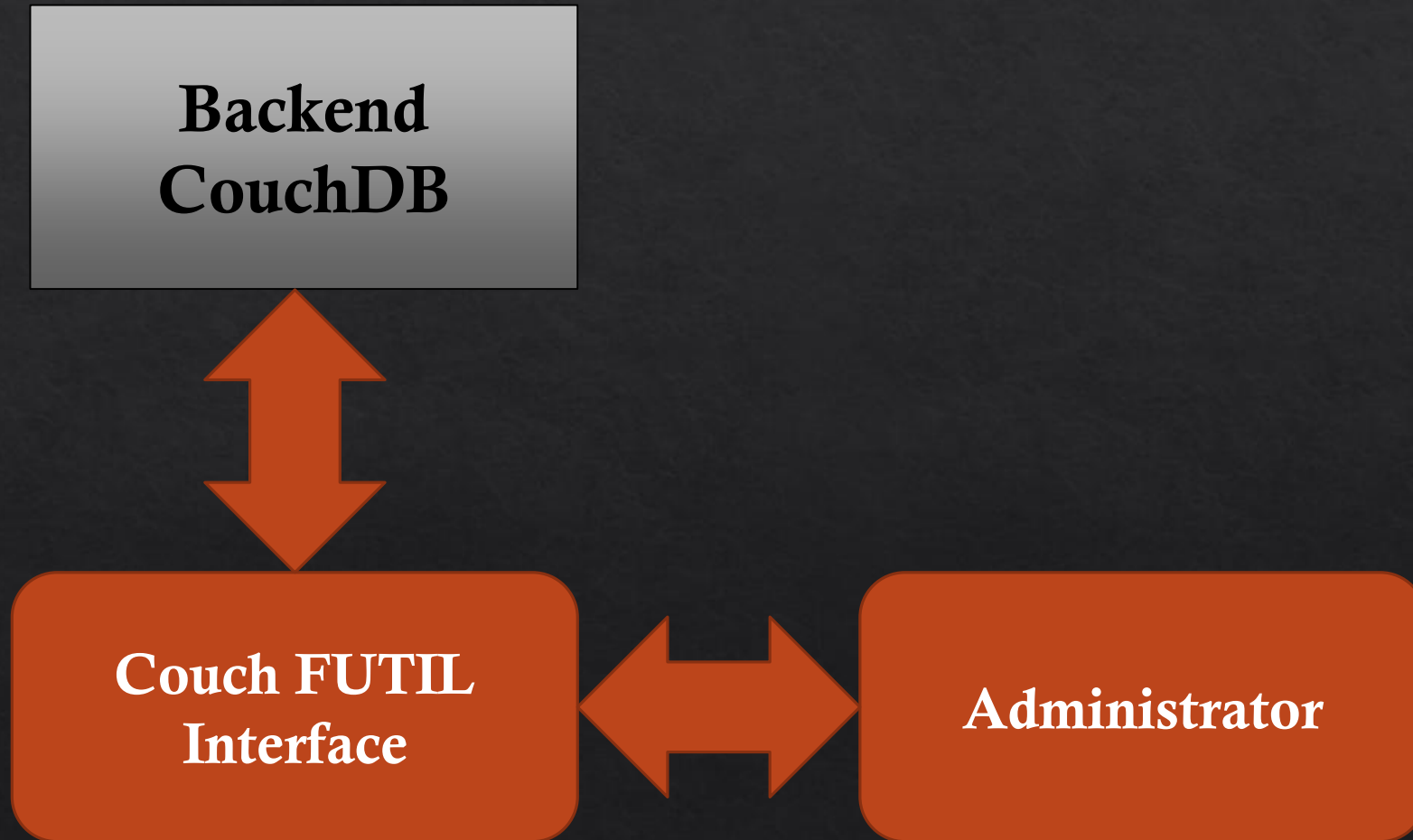
- An empty database takes up 192Mb

CouchDB

CouchDB Architecture



Couch Architecture



**Backend
CouchDB**



CouchDB
Attacker's perspective

XSS, CSRF, Sniffing

**Couch FUTIL
Interface**

Administrator

Key Features

Written in: Erlang

CouchDB document is a JSON object

Schema-Free

Main point: DB consistency, ease of use

Protocol: HTTP/REST

Distributed database system

Runs on Default Port : 5984, Binds to loopback interface by default

Client uses REST API to communicate with the Backend

Futon Web Interface

Attack Surface

Admin Party = Game Over.

Auth Cookie Sniffable

Credentials Send over Unencrypted Channel

XSPA attacks in Replication (limited to port web server ports)

XSS,HTML Injection in Futon Interface

DOS (Versions on 1.5 and below),File Enumeration attacks

Vulnerabilities

XSS at the token interface

HTML injection can be used by attackers to lure the victim to other sites.

XSPA Attack can be used in the replication to check whether port is open or not

Blind File Name Enumeration possible within the Replication

Addressing Auth Cookie

Defaults to Expire within 10 min

Attacker gaining access would want to use these 10 min Fruitfully

NoSQL Framework kicks in with automation session grabbing and dumping necessary info.

PHP on Couch Driver

Uses Curl Library to send the requests to the API

Un validated PHP APPS could result in calling Arbitrary API Call Execution

Download PHP on Couch:

<https://github.com/dready92/PHP-on-Couch/>


```
/**
 * fetch a CouchDB document
 *
 * @param string $id document id
 * @return object CouchDB document
 * @throws InvalidArgumentException
 */
public function getDoc ($id) {
    if ( !strlen($id) )
        throw new InvalidArgumentException ("Document ID is empty");

    if ( preg_match('/^_design/', $id) )
        $url = '/' . urlencode($this->dbname) . '/_design/' . urlencode(str_replace('_design/', '', $id));
    else
        $url = '/' . urlencode($this->dbname) . '/' . urlencode($id);

    $doc_query = $this->query_parameters;
    $this->query_parameters = array();

    $back = $this->_queryAndTest ('GET', $url, array(200), $doc_query);
    if ( !$this->results_as_cd ) {
        return $back;
    }
    $this->results_as_cd = false;
    $c = new couchDocument($this);
    return $c->loadFromObject($back);
}
```

```
* uses PHP cURL API
*
* @param string $method HTTP method to use (GET, POST, ...)
* @param string $url URL to fetch
* @param array $parameters additional parameters to send with the request
* @param string|array|object $data request body
* @param string $content_type the content type of the sent data (defaults to application/json)
*
* @return string|false server response on success, false on error
*
* @throws Exception
*/
public function _curl_query ( $method, $url, $parameters = array() , $data = NULL, $content_type = NULL ) {
    if ( !in_array($method, $this->HTTP_METHODS ) )
        throw new Exception("Bad HTTP method: $method");

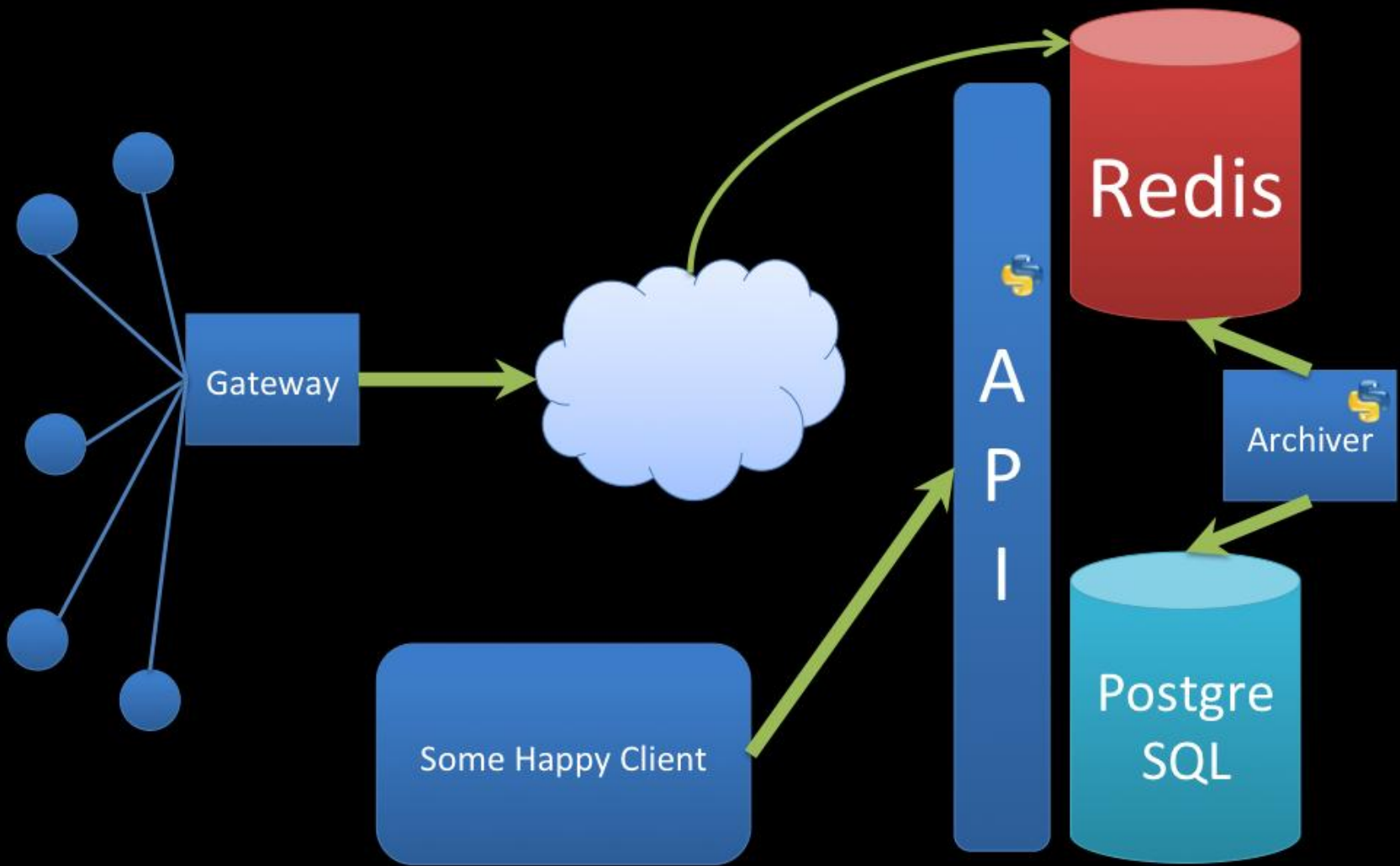
    $url = $this->dsn.$url;
    if ( is_array($parameters) AND count($parameters) )
        $url = $url.'?'.http_build_query($parameters);
    $http = $this->_curl_buildRequest($method,$url,$data, $content_type);
    $this->_curl_addCustomOptions ($http);
    curl_setopt($http,CURLOPT_HEADER, true);
    curl_setopt($http,CURLOPT_RETURNTRANSFER, true);
    curl_setopt($http,CURLOPT_FOLLOWLOCATION, true);

    $response = curl_exec($http);
    curl_close($http);
}
```

Vulnerable APP DEMO

Redis

Redis Architecture



Key Features

- ◇ Key Value storage engine.
- ◇ Contains Redis Server and Client
- ◇ Driven By a Config File
- ◇ Documentation is Laugh in a Park
- ◇ Redis supports five data structures:
 - ◇ strings, hashes, lists, sets and ordered sets.

Attacks Discussed

- ◇ Bruteforce Redis Passwords
- ◇ Denial of Service on the FLY.
- ◇ Command Killing
- ◇ Config Rewrite
- ◇ Arbitrary File Rewrite
- ◇ Blind File Enumeration (usefull in pentests)

```
##### SECURITY #####  
  
# Require clients to issue AUTH <PASSWORD> before processing any other  
# commands. This might be useful in environments in which you do not trust  
# others with access to the host running redis-server.  
#  
# This should stay commented out for backward compatibility and because most  
# people do not need auth (e.g. they run their own servers).  
#  
# Warning: since Redis is pretty fast an outside user can try up to  
# 150k passwords per second against a good box. This means that you should  
# use a very strong password otherwise it will be very easy to break.  
#
```

Redis Version

- ◇ Redis Version 2.6
 - ◇ No Support For Scripting.
- ◇ Redis Introduced version 2.8
 - ◇ Added Ability for LUA Scripts

Did We Just Say Scripting ?

Welcome to Redis LUA Script Engine and Basics

- ◇ Redis uses LUA to manage scripts
- ◇ LUA engine is properly sandboxed and offers enough security
- ◇ **Global variables protection**
- ◇ Scripts are Executed using eval (Available from Redis 2.6)

Key Points

- ◇ EVAL and EVALSHA are used to evaluate scripts using the Lua interpreter built into Redis.
- ◇ SCRIPT KILL,LISTS,EXISTS
- ◇ Important NB: When a script is running no other functions can be accessed or any operations can be performed

Sample Lua One Line DOS

- redis-cli eval "\$(cat test.lua)" 0
- test.lua

```
test.lua x
1 while(1) do
2
3 |end
```

Commands Disabled By an Attacker

- ◇ rename-command API Call Used
- ◇ Sample Command
 - ◇ rename-command CONFIG I33tshit
- ◇ rename-command CONFIG ""
 - ◇ Disables the command completely

Arbitrary File Rewrite

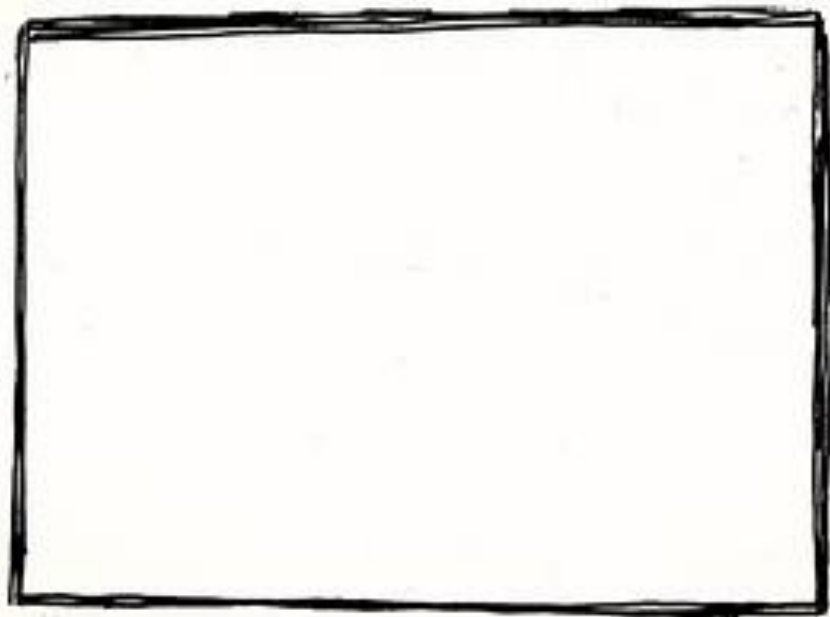
- ◇ CONFIG GET
 - ◇ Gives the Current set of Configuration
- ◇ CONFIG SET
 - ◇ Sets the configuration of the default command
- ◇ CONFIG SET dir /var/www

File Name Enumeration

- ◆ Restricted Environment
- ◆ Allows to use dofile (open file in lua scripting)
- ◆ Although file doesn't open gives the file or directory exists or not

- ◆ `eval "dofile('/var/www')"` 0
 - ◆ Directory Exists but cant open file
- ◆ `eval "dofile('/var/wwws')"` 0
 - ◆ No such directory exists

it's DEMOtime!



Vulnerable APP DEMO

Cassandra

Key Takeaways

- ◊ Written in: Java
- ◊ Main point: Store huge datasets in "almost" SQL
- ◊ Protocol: CQL3 & Thrift
- ◊ CQL3 is very similar SQL, but with some limitations that come from the scalability (most notably: no JOINS, no aggregate functions.)
- ◊ Runs on Port : 9160

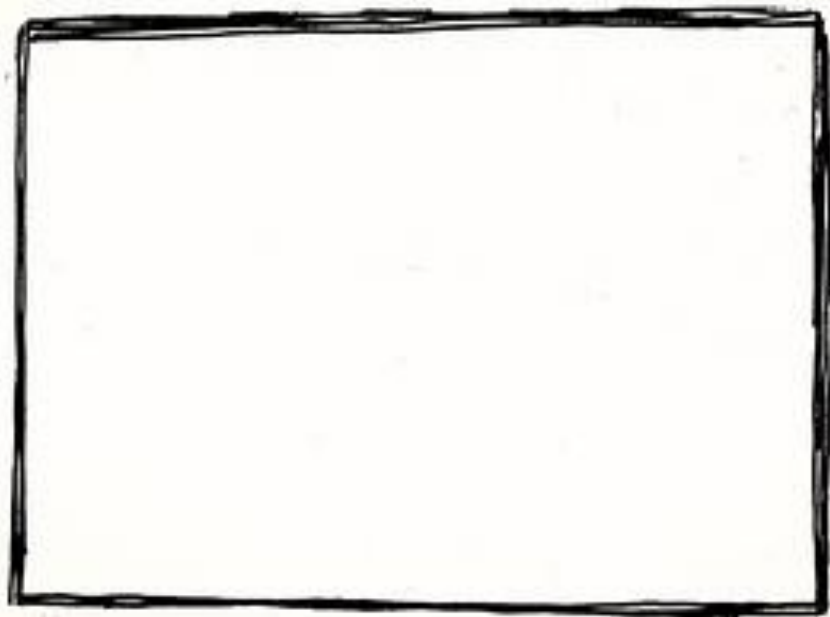
Sad Facts ?

- ◇ No OR
- ◇ No UNION
- ◇ No subrequests
- ◇ Terms must be indexed
- ◇ Primary key can only be queried for

Security Issues

- ◇ Cassandra model › Keyspace (=database) › ColumnFamily > Data
- ◇ CQL injection on Web Apps
- ◇ Shell Commands can be useful to attacker(during privilege escalation).
 - ◇ SOURCE command
 - ◇ Reads the contents of the file
 - ◇ Blind File Enumeration
- ◇ Database Enumeration and Dumping using NoSQL Exploitation Framework

it's DEMOtime!



H-Base

Key Points

- ◇ Written in: Java
- ◇ Main point: Billions of rows X millions of columns
- ◇ Protocol: HTTP/REST (also Thrift)
- ◇ Port:6379,8080(Rest API)
- ◇ Emphasis on Trusted Environment

Security Issues

- ◇ By Default offers No Security
- ◇ Man in The Middle Attacks
- ◇ REST API exposed
- ◇ Database Scanning and Enumeration using NoSQL Exploitation Framework

NoSQL DB's Never End!!

More Research Needed

- ◇ Neo4j, Memcache, Riak are under scanners (Some Discussed Vulnerabilities applies to those also)
- ◇ Support for Neo4j, Memcache and Riak soon to be added
- ◇ Memory Leaks and Overflows on the Rise
- ◇ An excellent address to Neo4j security issue was written recently
 - ◇ Link: <http://blog.scrt.ch/2014/05/09/neo4j-enter-the-graphdb/>

Is Automation Needed?

Do We have a framework ?

NoSQL Exploitation Framework

Key Points

A framework of one of its Kind

Open Source, Written In Python

- I am not a hardcode coder(Bugs are prone 😊)

Documented API's

Code Download:nosqlproject.com

Key Features

- ◇ Support for Mongo, Couch, Redis-Base and Cassandra
- ◇ Support for:
 - ◇ NoSQL Run Web Applications
 - ◇ Tests for JavaScript Attacks
 - ◇ Mongo DB \$ Attacks
 - ◇ Couch PHP Driver Attack Vectors
 - ◇ Multithreaded Mass IP List Scanner

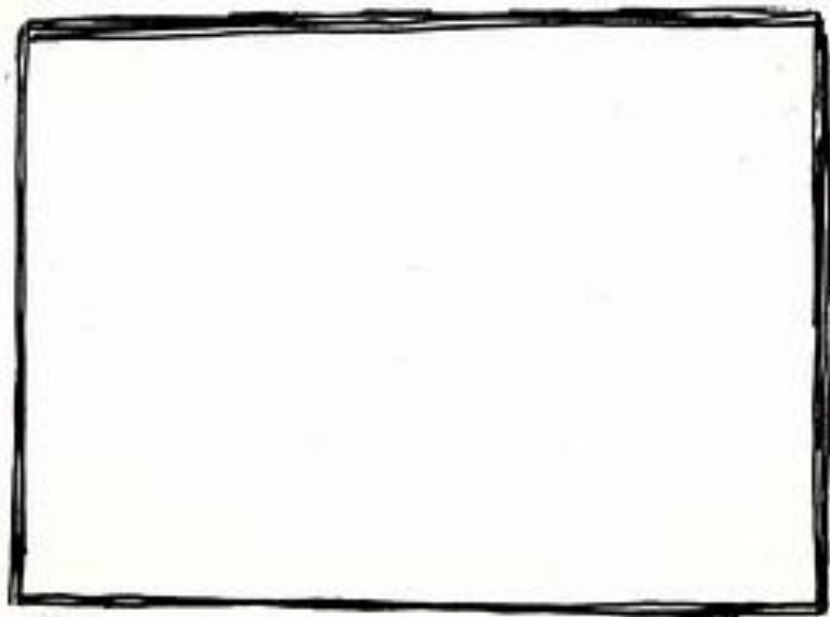
And the List Continues

- ◇ Database Cloning Feature
- ◇ Dictionary attacks
- ◇ Post Exploitation Module Integrated
- ◇ Shodan IP List Grabber
- ◇ Sniffing DB credentials and cookies
- ◇ Added More Payload List

Future Updates

- ◇ Updated Cassandra and HBase attacks
- ◇ Resource Exhaustion
- ◇ Support for Riak, Memcache and Ne04j on its way.
- ◇ Pentest Report Generation
- ◇ More Stable (Bug Less 😊)

it's DEMOtime!



Bugs or Contribute 😊

- ◇ Official Mailing List: feedback@nosqlproject.com
- ◇ Contribute By pulling from GITHUB.

References

- ◇ <http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis>

Thank You



**KEEP
CALM
AND
ASK
QUESTIONS**

I'm On

- ◇ Facebook : francis.alexander.33
- ◇ Twitter: @torque59
- ◇ Github: torque59
- ◇ LinkedIn: <http://in.linkedin.com/pub/francis-alexander/86/a94/521/>