Abusing NoSQL Databases

Ming Chow
Email: mchow@cs.tufts.edu
Twitter: @0xmchow

Why Care?

- That was then: a few SQL database options for any application
- This is now: a plethora of database options, you have to choose the right database for the right job
- Many NoSQL databases are built for performance, scalability, and flexibility
- Security of NoSQL databases? Weak, inconsistent, the wild wild west

Why Am I Here?

- I talked on abusing HTML5 back at DEF CON 19
- Bryan Sullivan scratched the surface with his BlackHat 2011 work "Server-Side JavaScript Injection"
- The rise of client and *server-side* JavaScript
- There is a lot to just the database side of things

Straight Out-of-the-Box General Issues: The Defaults

- Easy win: know the database vendor, IP address, and an open port number. The default open port numbers:
  - Mongo: 27017, 28017, 27080
  - CouchDB: 5984
  - Hbase: 9000
  - Cassandra: 9160
  - Neo4j: 7474
  - Redis: 6379
  - Riak: 8098

Straight Out-of-the-Box General Issues: Authentication and Encryption

- (Almost) No NoSQL database enables an administrator user or authentication by default
  - Even if users are enabled, weak password storage
    - Mongo uses md5
    - Plaintext for Redis
    - Weak salt or plaintext for CouchDB
- Client communicates with server via plaintext
- Database and data file encryption and auditing features are generally not available
- Emphasis on "trusted environments"

New Classes of Injection Attacks

1. **Query**: creating unsafe queries via string concatentation (sounds familiar?)
2. **Schema**: inserting a record into a schema that does not exist will automatically create the new schema
3. **JavaScript**: `$where`, `db.eval()` take in JavaScript *functions* as parameters

Schema Injection

- Source code of tool: https://github.com/mchow01/Security/blob/master/DEFCON21/pollute-nosql.rb
- Written in Ruby
- Usage: `pollute-nosql.rb couchdb|mongo|redis host wordfile [value_for_each_key]`
- Example: `ruby pollute-nosql.rb mongo 192.168.39.128 /path/to/metasploit-framework/data/wordlists/unix_passwords.txt BOO`

JavaScript Injection

- Vulnerable code 1 (in PHP): https://github.com/mchow01/Security/blob/master/DEFCON21/search_by_handle.php
  - HINT: `$cursor = $collection->find(array('screen_name' => $searchbox));`
- Vulnerable code 2 (in PHP): https://github.com/mchow01/Security/blob/master/DEFCON21/search_hackme.php
  - HINT: `$cursor = $collection->find(array('$where' => $searchbox));`

Equivalence in JavaScript: Exact Match

- Each of the following will return the same results:
  - `db.news.find({"screen_name":"CBSNews"})`
  - `db.news.find("this.screen_name == 'CBSNews'")`
  - `db.news.find({$where:"this.screen_name == 'CBSNews'"})`
  - `db.news.find({'$where':function() {return this.screen_name == 'CBSNews';}})`

Equivalence in JavaScript: Regular Expressions

- Each of the following will return the same results:
  - `db.news.find({"text":/Apple/})`
  - `db.news.find({'text':{$regex:'Apple'}})`
  - `db.news.find({'$where':"this.text.match(/Apple/)"})`
  - `db.news.find({'$where':function() {return this.text.match(/Apple/);}})`

Demonstration

Now knowing the equivalences in JavaScript, what inputs can you give to bring back “interesting” results?

A Problem in PHP

- Say you have http://domain/search_by_handle.php?searchbox=PandoDaily&submitbutton=Submit
- If you modify the URI to http://domain/search_by_handle.php?searchbox*[$ne]*=PandoDaily&submitbutton=Submit, PHP will automatically create associative arrays from query string inputs with square brackets.
- Alas, we have: `$collection->find(array ('screen_name' => array ('$ne':'PandoDaily')));`
- In Mongo, `$ne` is the not equals operator

What About `search_hackme.php`?

1. Want everything by @CBSNews? Use this for input (searchbox): `function() {return this.screen_name=='CBSNews';}` as the input
2. Want everything that has the word “Apple” in it? Use this for input (searchbox): `function() {return this.text.match(/Apple/);}`

A Heterogeneous Problem

- RTFM for each database system
- Different for each system:
  - Terminologies and analogies
  - Methods of granting permissions and user control
  - Flavors of query types, including: Cassandra Query Language (CQL), command-based queries, JavaScript
  - Flavors of query results, including: JSON, BSON (Binary JSON)

Vendor-Specific Items

- MongoDB:
  - `mongod` is bind to all interfaces
  - The `run()` command can act as a shell
  - Easy information gathering by simply looking at the `startup_log` in the `local` collection (shows pid, OS details, paths)
  - `mongosniff` tool comes with mongo installation for "tracing/sniffing view into database activity in real time"
- CouchDB:
  - HTTP document REST API exposed by default

Old Security Matters

- Really important:
  - Architecture
    - Since many NoSQL databases have weak security, more controls may be necessary
  - Configuration
  - Validation becomes even more important
    - No longer are we just validating input strings but also results and JavaScript functions

The Takeaways

1. No longer a one-size-fits-all game
2. Plenty of new attack vectors, contrary to the idea that SQL injection is practically gone thus eliminating many concerns
3. Technologies being deployed naively
4. Database vendors have left security largely to the developers
5. The reports of the death of database administrators are greatly exaggerated

References

- Chow, M. "JavaScript Pitfalls" SOURCE Boston Conference 2013
- Okman et al "Security Issues in NoSQL Databases" http://jmiller.uaa.alaska.edu/cse465-fall2012/papers/okman2011.pdf
- Sullivan, B. "Server-Side JavaScript Injection" Black Hat USA 2011 http://media.blackhat.com/bh-us-11/Sullivan/BH_US_11_Sullivan_Server_Side_WP.pdf
- Urbinsky, W. "NoSQL, No Security?" AppSec USA 2012, Austin, Texas. http://www.slideshare.net/wurbanski/nosql-no-security
- http://www.slideshare.net/gavinholt/no-sql-no-security-20074309
- http://blogs.adobe.com/asset/files/2011/04/NoSQL-But-Even-Less-Security.pdf
- http://blog.astyran.sg/2011/11/there-is-no-security-in-nosql.html
- http://www.darkreading.com/database/does-nosql-mean-no-security/232400214
- https://securosis.com/blog/nosql-and-no-security
- http://blog.spiderlabs.com/2013/03/mongodb-security-weaknesses-in-a-typical-nosql-database.html
- http://jkb.netii.net/index.php/pub/sinosqldb/cassandra-security