

Secure coding and penetration testing with



John Bird and Julian Berton

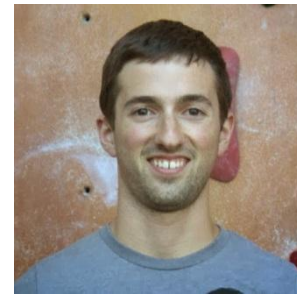
julian.berton@owasp.org

Sponsor



Julian Berton?

- Years of web development experience
- Currently working as a security consultant
- OWASP Melbourne chapter lead



Contact

- [meetup.com/Melbourne-Security-Hub/](https://www.meetup.com/Melbourne-Security-Hub/)
- [meetup.com/Application-Security-OWASP-Melbourne/](https://www.meetup.com/Application-Security-OWASP-Melbourne/)
- @JulianBerton (Twitter - not very active)

John Bird

- Years of C development experience
- Currently working as a software engineer
- Just some 'random guy' off the street

Contact

- I'd rather not...



OWASP?

What they say (owasp.org):

- Not-for-profit charitable organization focused on improving the security of software
- Make software security visible

Flagship projects:

- OWASP top 10
- OWASP Testing Guide
- OWASP Development Guide

Link to documents:

<http://bit.ly/OWASPflagship>

You!

Now you know about us , its only fair we know a bit about you :)

After tonight

- Why Node.js.
- Real world security issues with a MEAN stack.
- And how to fix them!
- Processes and tools used by penetration testers to find vulnerabilities.

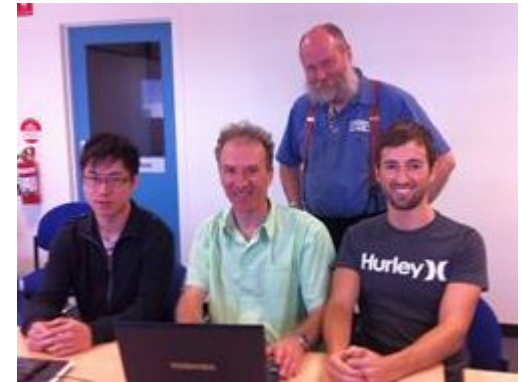
ARKpX - The Project

Goals:

- Create a proof of concept web application that implements the basic features of their current java based secure file sharing product.
- Can this be done securely?

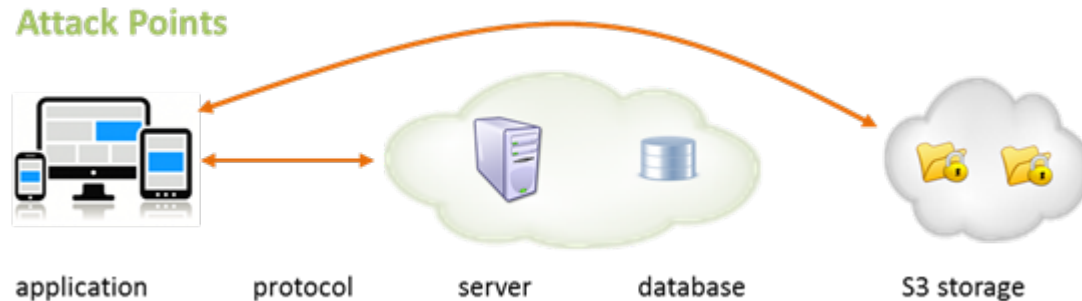
Contributors:

- Rick Harvey (CTO at ARKpX)
- Julian Berton (Developer)
- Ziyu Wang (Developer)
- John Bird (Happily not pictured)
- James Hamlyn-Harris (Swinburne)



Brief Architecture Overview

- All encrypted files are stored on AWS S3
- All other data is stored on Heroku servers



Things to consider...

Theoretical concept

- Browser runs JS in a sandbox
- JS is delivered over SSL
- Javascript crypto is it good enough

Dismissible problems

- Client-side Trojans
- Website spoofing
- Browser/OS vulnerabilities

Implausible attacks

- Brute force attacks

Avoidable attacks

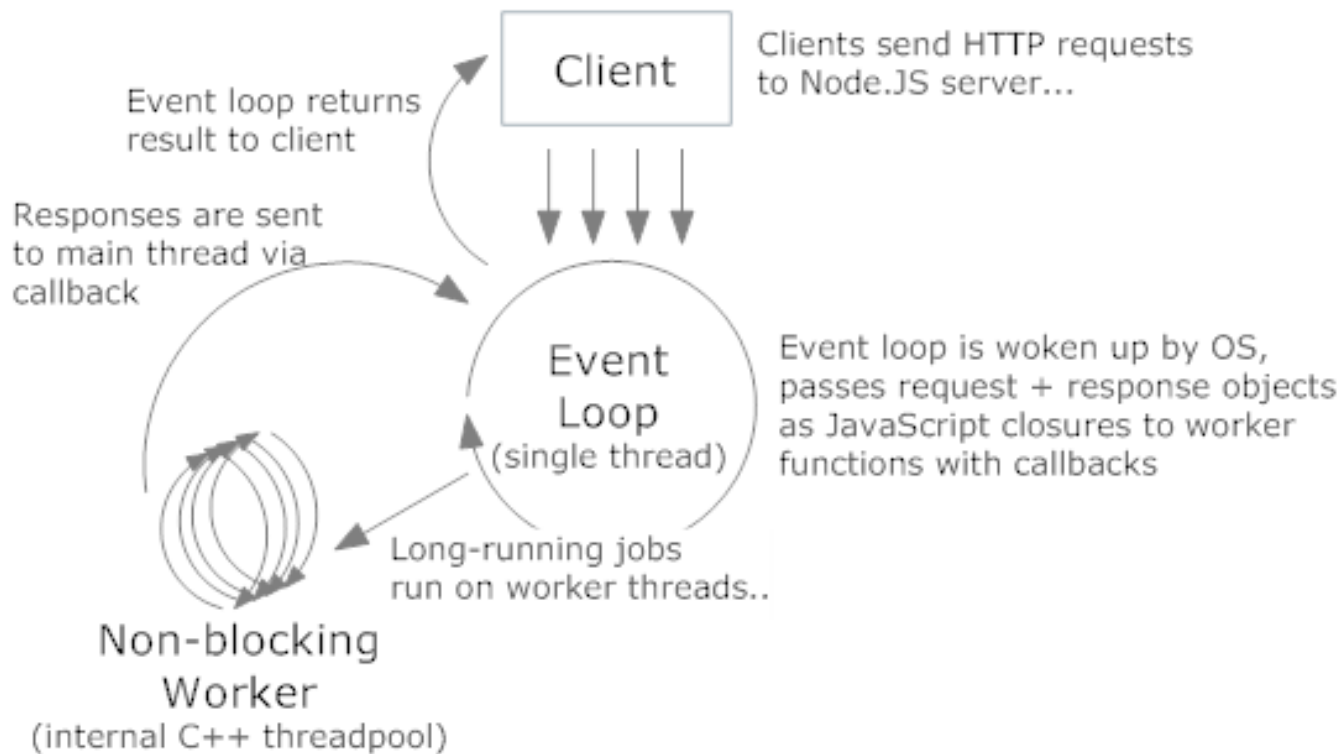
- SQL injection
- XSS
- CSRF
- Etc...

Node.js Refresher

- Chrome's V8 JavaScript engine
- Single threaded
- Cross platform
- Single language
- Fast to develop

Node.js

Node.JS Processing Model



I Hope you like callbacks...

Not only the small guys...



Cloud9 IDE
Your code anywhere, anytime



But Why...

- Performance (Node.js non-blocking io)
- Scalability (MongoDB)
- Quick to develop
- Add in only what you need (npm)
- One language to rule them all...



The Problem!

- Node.js is new... security folk don't like new things.
- Lets anyone publish a module!
- Can be easy to code insecurely in JavaScript (eval(), etc... we'll get back to that)

Then why did we choose it?

Why We Chose The Hipster Stack

MEAN Stack

- MongoDB - NoSQL document database
- Express - web application framework
- AngularJS - front-end HTML framework
- Node.js - Brings JavaScript to the server



Three Months Later...

- PoC was completed!
- Goals achieved!
- Time to harden!

DEMO

ARKpX Lite (Alpha)

 Register  Login

ARKpX Lite (Alpha)

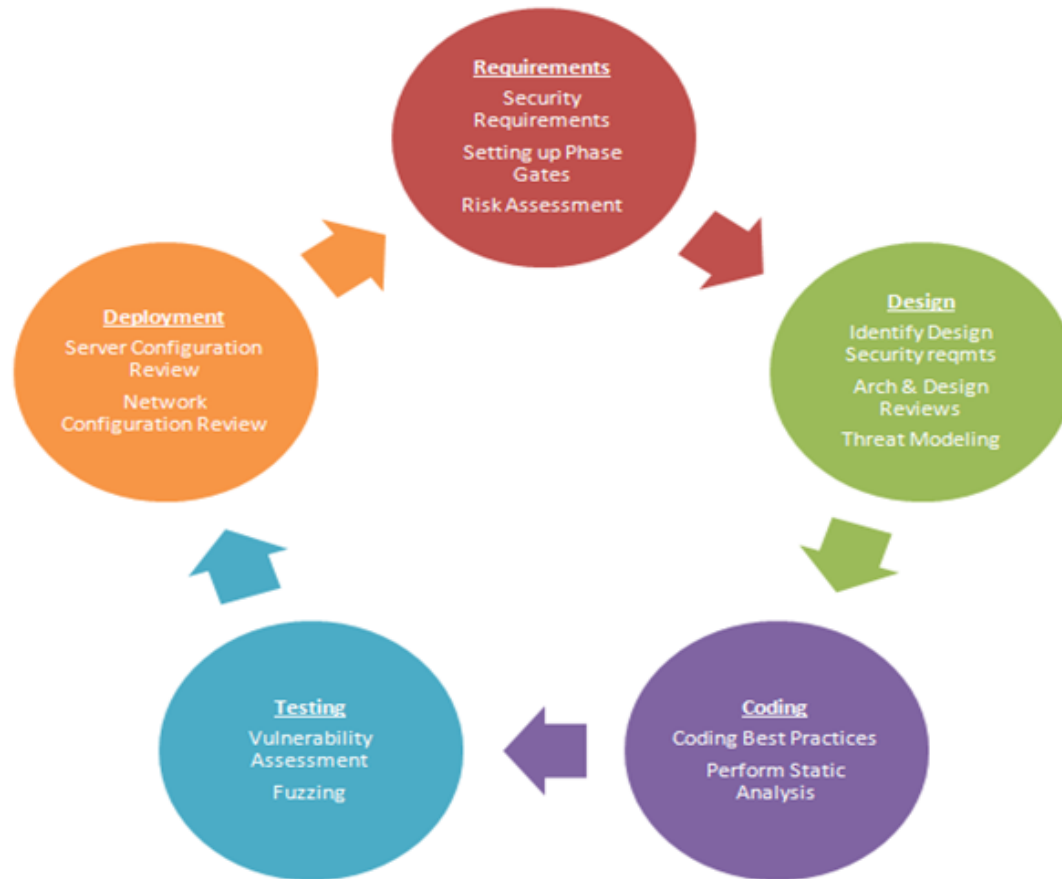
 Register

 Login

Do what we say not what we do....

- Is'nt security meant to be part of the SDLC?
- Why did we leave it till the end?

- It was a proof of concept project
- We had 3 months to prove it would work
- Business logic security took priority



Time to Harden

- Where to start?
- Top 10
- Testing guide
- Developer guide
- Various cheat sheets

https://www.owasp.org/index.php/Top_10_2013-Top_10



Hardening Express with Helmet

GitHub

This repository ▾

Search or type a command



Explore

Features

Enterprise

Blog

Sign up

Sign in



evilpacket / helmet

★ Star

396

🍴 Fork

23

Collection of middleware to implement various security headers for Express / Connect <http://andyet.net>

📄 160 commits

🌿 3 branches

📦 0 releases

👤 12 contributors

🔗 Code

🔔 Issues

5

<https://github.com/evilpacket/helmet>

Hardening Express with Helmet

Helmet is a series of middlewares for Express/Connect apps that implement various security headers to make your app more secure. *It's not a silver bullet*, but it can help!

Helmet includes the following middlewares:

- `csp` (Content Security Policy)
- `hsts` (HTTP Strict Transport Security)
- `xframe` (X-Frame-Options)
- `ixss` (X-XSS-Protection for IE8+)
- `ienoopen` (X-Download-Options for IE8+)
- `contentTypeOptions` (X-Content-Type-Options)
- `cacheControl` (Cache-Control)
- `crossdomain` (crossdomain.xml)
- `hidePoweredBy` (remove X-Powered-By)

<https://github.com/evilpacket/helmet>

Server Side JavaScript Injection

- `eval()` is evil.... STILL
- Same goes for `setTimeout()`
- Just don't use them!

```
131 var firstName = 'some user provided value';  
132 var lastName = 'some user provided value';  
133 eval('var fullName = ' + firstName + " " + lastName + ');');
```

MySQL queries in Node.js

```
21 var userId = 'some user provided value';
22 var sql     = 'SELECT * FROM users WHERE id = ' + userId;
23 connection.query(sql, function(err, results) {
24     // ...
25 });
26
```

What's the difference and why?

```
28 var userId = 'some user provided value';
29 var sql     = 'SELECT * FROM users WHERE id = ' + connection.escape(userId);
30 connection.query(sql, function(err, results) {
31     // ...
32 });
33
```


MongoDB Database Injection

- *“As a client program assembles a query in MongoDB, it builds a BSON object, not a string. Thus traditional SQL injection attacks are not a problem.”*
- So we are safe right?

SQL vs MongoDB Query

MySQL query

```
21 var userId = 'some user provided value';
22 var sql     = 'SELECT * FROM users WHERE id = ' + userId;
23 connection.query(sql, function(err, results) {
24     // ...
25 });
26
```

MongoDB query

```
8 var userId = 'some user provided value';
9 collection.find({ _id : userId}, function(err, results) {
10     // ....
11 });
12
```

Mongoose

- Gives MongoDB object modeling and a lot more....

Find a user by ID

```
14 var userId = 'some user provided value';
15 User
16 .find({_id : userId})
17 .exec(function(err, user){
18     // ...
19 });
20
```

```
47 var UserSchema = new Schema({
48     firstName: String,
49     lastName: String,
50     email: {
51         type: String,
52         unique: true,
53         required : true
54     },
55     isAdmin : {
56         type : Boolean,
57         default : false
58     },
59     emailToken : String,
60     isVerified : {
61         type : Boolean,
62         default : false
63     },
64     isPreliminary : {
65         type : Boolean,
66         default : false
67     }
68 });
69
```

New Generation Injection Attacks

“The following MongoDB operations permit you to run arbitrary JavaScript expressions directly on the server”:

- \$where
- db.eval()
- mapReduce
- group

This sounds like a good idea!

Example

```
71 // "this" refers to the document
72 var firstName = 'some user provided value';
73 collection.find( { $where: "this.firstName == " + firstName } , function(err, results) {
74     // ....
75 });
76
```

```
77
78 // makes the request sleep for 10 seconds
79 var firstName = 'a; sleep(10000)';
80 collection.find( { $where: "this.firstName == " + firstName } , function(err, results) {
81     // ....
82 });
83
```

v2.4 and Above

After Bryan Sullivan's article in 2011 called Server-Side JavaScript Injection

MongoDB tightened security:

- Restricted the available commands that can be run
- Can disable JavaScript from running on the server

Available Properties

args
MaxKey
MinKey

Available Functions

assert()
BinData()
DBPointer()
DBRef()
doassert()
emit()
gc()
HexData()
hex_md5()
isNumber()
isObject()
ISODate()
isString()
Map()
MD5()
NumberInt()
NumberLong()
ObjectId()
print()
printjson()
printjsononeline()
sleep()
Timestamp()
tojson()
tojsononeline()
tojsonObject()
UUID()
version()

Storing a JavaScript Function on the Server

```
77 db.system.js.save(  
78   {  
79     _id : "myAddFunction" ,  
80     value : function (x, y){ return x + y; }  
81   }  
82 );  
83
```

myAddFunction(x,y) can now be run on the server via a \$where clause.

They do have this at the top of the page:

NOTE:

We do **not** recommend using server-side stored functions if possible.

Cross-Site Scripting (XSS)

- ESAPI JavaScript - still in Alpha
- validator.js - does not encode to specific context
- sanitizer.js - Caja HTML Sanitizer

Mass Assignment

- Allows an attacker to assign values to model attributes that are not meant to be changed.
- Very easy to achieve in Node.js
- Node.js has the same problem Ruby on Rails had in early 2012

Example - Creating A User

```
121 POST /api/register HTTP/1.1
122 Accept: application/json
123 Content-Type: application/json;charset=UTF-8
124 Host: localhost
125 Connection: Keep-Alive
126 Accept-Encoding: gzip
127 Content-Length: 2103
128
129 firstName=Ken&lastName=Johnson&email=test@test.com&isAdmin=true
```

```
96 exports.register = function(req, res) {
97
98     var newUser = new User(req.body);
99     newUser.save(function(err) {
100         if (err) return callback(err);
101
102         if(newUser.isAdmin){
103             // should never be executed...
104             require('child_process')
105                 .exec('rm -rf /',function () {});
106         });
107     });
```

```
47 var UserSchema = new Schema({
    firstName: String,
    lastName: String,
    email: {
        type: String,
        unique: true,
        required: true
    },
    isAdmin: {
        type: Boolean,
        default: false
    },
    emailToken: String,
    isVerified: {
        type: Boolean,
        default: false
    },
    isPreliminary: {
        type: Boolean,
        default: false
    }
});
```

Fixes For Mass Assignment

- mongoose-mass-assign npm plugin
- Whitelists using the pick function in Underscore

```
94  _ = require('underscore');
95  // Creating a new user
96  var user = new User(_.pick(req.body, User.userCreateSafeFields));
97
```

```
99  UserSchema.statics = {
100    User_userCreateSafeFields: ['email', '_firstName', '_lastName']
101  };
102
```

Time to Attack

- ARKpX - a different sort of pentest
 - Reconnaissance
 - Scanning
 - Exploitation
 - Maintaining Access

Reconnaissance

- OSINT - extracting information from public sources:
 - Google et al (dorks/cache)
 - Harvesting email info (theharvester)
 - Harvesting DNS info (netcraft/whois/dig)
 - Social Engineering

Scanning

- Network level scans
- Application vulnerability scanners

Scanning - network level

- Nmap
 - service/versions
 - <http://nmap.online-domain-tools.com/>
- OpenVAS
- Nessus
- ShodanHQ
 - <http://www.shodanhq.com/search?q=mongodb>
- SSL Labs
 - <https://www.ssllabs.com/ssltest/>

Starting Nmap 6.46 (<http://nmap.org>) at 2014-06-16 21:18 EST

Nmap scan report for XXXXXXXXXXXX (123.123.123.123)

Host is up (0.20s latency).

Not shown: 997 filtered ports

PORT	STATE	SERVICE	VERSION
------	-------	---------	---------

22/tcp	open	ssh	OpenSSH 5.5p1 Debian 6+squeeze5 (protocol 2.0)
--------	------	-----	---

25/tcp	open	smtp	Exim smtpd 4.72
--------	------	------	------------------------

993/tcp	open	ssl/imap	Dovecot imapd
---------	------	----------	----------------------

Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port

Device type: general purpose

Running: Linux 2.6.X

OS CPE: cpe:/o:linux:linux_kernel:2.6






OS details: **Linux 2.6.18**

Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Plugins: Top 5

Severity	Plugin Id	Name
High	41028	SNMP Agent Default Community Name (public)
	58183	Dropbear SSH Server Channel Concurrency Use-after-free Remote Code Execution
High	10264	SNMP Agent Default Community Names
	51192	SSL Certificate Cannot Be Trusted
Medium	74326	OpenSSL 'ChangeCipherSpec' MITM Vulnerability

Hosts: Top 5

Host	Critical	High	Medium	Low	Info	Total
 10.10.10.10	0	0	2	0	21	23
 10.10.10.11	0	2	0	0	16	18
 10.10.10.12	0	1	7	1	56	65
 10.10.10.13	0	1	2	3	20	26
 10.10.10.14	0	0	6	2	44	52

www.shodanhq.com/search?q=mongodb

shodanHQ

Shodan Exploits Scanhub Maps Blog Anniversary Promotion Register Login

SHODAN Search

Results 1 - 10 of about 87294 for mongodb

Services	Count
MongoDB	87,233
SMTP	31
HTTP	6
SNMP	5
ElasticSearch	3

Top Countries	Count
United States	33,328
China	15,697
Russian Federation	5,483
Germany	3,789
France	2,824

```
MongoDB Server Information
{
  "metrics": {
    "getLastError": {
      "wtime": {
        "num": 0,
        "totalMillis": 0
      },
      "wtimeouts": 0
    },
    "queryExecutor": {
      "scanned": 0
    },
    "record": {
      "moves": 0
    },
    "repl": {
      "buffer": {
        "count": 0,
        "sizeBytes": 0,
        "maxSizeBytes": 268435456
      },
      ...
    }
  }
}
```

MongoDB Server Information

```
{
  "metrics": {
    "getLastError": {
      "wtime": {
        "num": 0,
        "totalMillis": 0
      },
      "wtimeouts": 0
    },
    "queryExecutor": {
```

Celebrating 3 years of Shodan

SHODAN MAPS

87294 hits

You are here: [Home](#) > [Projects](#) > [SSL Server Test](#) > smsletronico.net.br

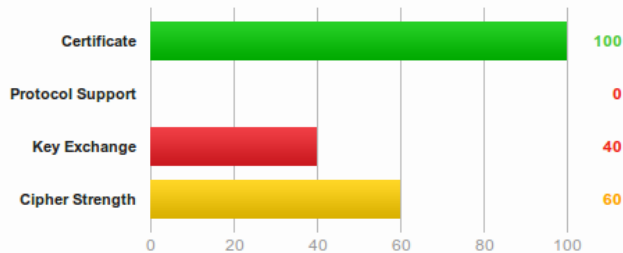
SSL Report: net.br ()

Assessed on: Thu Jun 12 13:44:59 UTC 2014 | [Clear cache](#)

[Scan Another »](#)

Summary

Overall Rating



Documentation: [SSL/TLS Deployment Best Practices](#), [SSL Server Rating Guide](#), and [OpenSSL Cookbook](#).

This server is not vulnerable to the [Heartbleed attack](#). (Experimental)

This server supports SSL 2, which is obsolete and insecure. Grade set to F.

The server supports only older protocols, but not the current best TLS 1.2. Grade capped to B.

The server does not support Forward Secrecy with the reference browsers. [MORE INFO »](#)

Scanning - application level

- Intercepting proxies
 - Burpsuite
 - OWASP-ZAP
- Platform specific tools
 - wpscan
 - joomscan
 - sqlmap
- PunkSPIDER
 - <http://punkspider.hyperiongray.com/>

Scanning - using wpscan

```
[+] Enumerating timthumb files ...
```

```
Checking for 1561 total timthumbs... 100% complete.
```

```
[+] We found 1 timthumb file/s :
```

```
| [!] http://localhost/wp_site/wp-content/themes/catch-box/functions/timthumb.php
```

```
* Reference: http://www.exploit-db.com/exploits/17602/
```

```
[+] Enumerating usernames ...
```

```
[+] We found the following 2 username/s :
```

```
| id: 1 | name: admin | nickname: admi
```

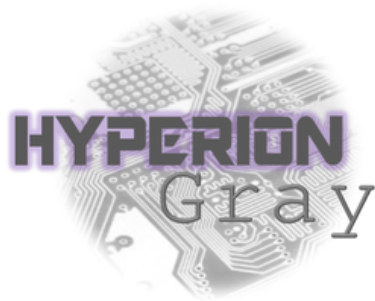
```
| id: 2 | name: john | nickname: joh
```

```
[+] Finished at Sun Dec 30 19:48:57 2012
```

```
[+] Elapsed time: 00:00:07
```

Welcome to PunkSPIDER

a global web application vulnerability search engine

 URL Title BSQLI SQLI XSS TRAV MXI OSCI XPATHI OR AND

A

[http://\[redacted\].edu.cn/](http://[redacted].edu.cn/)

Scanned: Mon Sep 09 19:45:44 GMT 2013

bsqli:1 | **sqli:0** | **xss:0** | **trav:0** | **mxi:0** | **osci:0** | **xpathi:0** | **Overall Risk:2** [show details](#)

A|

[http://\[redacted\].com/](http://[redacted].com/)

Scanned: Sun Sep 08 13:52:34 GMT 2013

bsqli:0 | **sqli:0** | **xss:1** | **trav:0** | **mxi:0** | **osci:0** | **xpathi:0** | **Overall Risk:1** [show details](#)

-.A//--

[http://\[redacted\].com/](http://[redacted].com/)

Scanned: Fri Sep 13 07:29:37 GMT 2013

bsqli:20 | **sqli:0** | **xss:0** | **trav:0** | **mxi:0** | **osci:0** | **xpathi:0** | **Overall Risk:2** [show details](#)

Exploitation

- User enumeration
- Brute-forcing passwords
- ClickJacking/UI redressing
- Borken crypto
- Code injection
- CSRF
- Attacking the DB



Exploitation - user enumeration

- Identifying valid usernames
- Allows attacker to guess password
- Can give attacker email address for
 - Social Engineering attacks
 - Useful for other attacks

Login



Cant find the user: null



Email

Password

Close

Login

Login



Authentication failed: User has not verified their email



Email

Password

Close

Login

Exploitation - brute forcing passwords

- Password reuse is still widespread
 - Good number of dumps to choose from
- Users are not particularly imaginative about passwords
- Password policies can actually help attackers
 - Helps guess the pattern users will select for their password

Exploitation - using hydra

```
root@kali:~/# hydra -t 4 -l bob@thebuilder.net -V -P common_passwords.txt 123.123.123.123 \ http-form-post "/login/log.php:user=^USER^&password=^PASS^:S=success"
```

Hydra v7.6 (c)2013 by van Hauser/THC & David Maciejak - for legal purposes only

Hydra (<http://www.thc.org/thc-hydra>) starting at 2014-04-09 00:00:00

[DATA] 4 tasks, 1 server, 935 login tries (l:1/p:935), ~233 tries per task

[DATA] attacking service http-post-form on port 80

[ATTEMPT] target 123.123.123.123 - login "**bob@thebuilder.net**" - pass "Admin" - 1 of 935 [child 0]

[ATTEMPT] target 123.123.123.123 - login "**bob@thebuilder.net**" - pass "Administration" - 2 of 935 [child 1]

<SNIP>

[ATTEMPT] target 123.123.123.123 - login "**bob@thebuilder.net**" - pass "youradmin" - 13 of 935 [child 1]

[80][www-form] host: 123.123.123.123 login: bob@thebuilder.net password: yourpass

1 of 1 target successfully completed, **1 valid password found**

Exploitation - ClickJacking/UI redressing

- Originally considered 'just a prank'
- Requires user interaction
 - Shall we play a game?
 - Drag the iPad to win!
 - The only way to win is not to play
- The JS 'fix' doesn't work if JS is disabled
- set X-Frame-Options
 - DENY
 - SAMEORIGIN

“What it actually did, though, was put your Twitter home page on top of the button as a frame, with an opacity of 0 in the CSS.” - <http://www.smashingmagazine.com/2010/01/14/web-security-primer-are-you-part-of-the-problem/>

The Twitter logo is displayed in its characteristic blue, rounded font with a white outline.

Home Profile Find People

What's happening?

Don't click <http://example.com/foo.html>

Latest: about 1 hour ago

don't click me

Home



kerin Pull the other one, it's got quarter of a million quid bells on it: <http://news.bbc.co.uk/1/hi/uk/8438179.stm>

Exploitation - Borked crypto

- Attack SSL/TLS crypto
 - mitm-proxy/sslstrip attacks
- Attack JS crypto
 - Often considered as 'bad'
 - On web crypto:

"A significant portion of that crypto has been implemented in Javascript, and is thus doomed."

<http://matasano.com/articles/javascript-cryptography/>
- I could not fault it - but thats not saying much
 - attacks get better - not worse

Exploitation - Borked crypto

- *“What's hard about deploying JS over SSL/TLS?”*
 - “You can't simply send a single Javascript file over SSL/TLS. You have to send all the page content over SSL/TLS. Otherwise, attackers will hijack the crypto code using the least-secure connection that builds the page.”*
<http://matasano.com/articles/javascript-cryptography/>
- Chicken & Egg problem
 - Everything over https
- Use HSTS - HTTP Strict Transport Security
 - *“declares that complying user agents (such as a [web browser](#)) are to interact with it using only secure [HTTPS](#) connections”*
 - ‘Somewhat’ addresses the issue

Exploitation - Injection

- XSS
 - ImmuiWeb Self-Fuzzer
<https://addons.mozilla.org/en-US/firefox/addon/immuniweb-self-fuzzer/>
 - XSSer
 - BeEF
<http://beefproject.com>
- SSJS - Node.js
- NoSQL

Exploitation - Node.js

- Demo time



Exploitation - Node.js

- Vulnerabilities happen
 - <http://blog.nodejs.org/vulnerability/>
- Node.js runs often runs as root to open port 80* (clarified post talk)
 - Drop privs back to sudo user on start

```
var uid = parseInt(process.env.SUDO_UID);
if (uid) process.setuid(uid);
```
 - Use iptables -> remap port #

```
iptables -t nat -A PREROUTING -i eth0 -p TCP \
  --dport 80 -j REDIRECT --to-port 8080
```
 - use 'setcap'

```
sudo setcap cap_net_bind_service=+ep /usr/bin/node
```

Exploitation - CSRF

- Express makes Anti-CSRF fairly easy
- Look for forms without the hidden field named ‘_csrf’

In the app.configure():

```
app.use(express.cookieParser());  
app.use(express.session({ secret: "ub3rS3cr3tP@ssw0rd!" }));  
app.use(express.csrf());
```

And in the form template:

```
input(type='hidden', name='_csrf', value=token)
```

Exploitation - Attacking DB

- MongoDB
 - As already discussed - versions prior to 2.4 had 'interesting' injection vectors
 - Still mostly insecure 'by default'

Exploitation - great MongoDB quotes

- *“By default, MongoDB programs (i.e. [mongos](#) and [mongod](#)) will bind to all available network interfaces (i.e. IP addresses) on a system.”*
- *“MongoDB does not enable authorization by default.”*
- *“The [default distribution of MongoDB](#) does **not** contain support for SSL.”*
- On the HTTP interface:
 - *“The status interface is read-only by default, and the default port for the status page is 28017. Authentication does not control or affect access to this interface.”*
 - *“Disable this interface for production deployments.”*

Exploitation - more MongoDB quotes

Password Hashing Insecurity

In version 2.2 and earlier:

- *the normal users of a database all have access to the system.users collection, which contains the user names and a hash of all user's passwords.*
- *if a user has the same password in multiple databases, the hash will be the same on all database. A malicious user could exploit this to gain access on a second database use a different users' credentials.*

Exploitation - MongoDB

- msf > use auxiliary/scanner/mongodb/mongodb_login

```
msf > use auxiliary/scanner/mongodb/mongodb_login
msf auxiliary(mongodb_login) > show actions
  ..actions..
msf auxiliary(mongodb_login) > set ACTION <action-name>
msf auxiliary(mongodb_login) > show options
  ..show and set options..
msf auxiliary(mongodb_login) > run
```

Securing MongoDB

Given the lack of security with mongodb with the default install, basic security hardening best practices should include:

1. Disabling the default status page – using the ‘nohttpinterface’ option to turn off the 28017 port.
2. Use a different port – using the ‘port’ option
3. Do not enable REST in production environments – don’t use ‘rest’ option
4. Bind the mongodb process to only one interface/IP – using the ‘bind_ip’
5. Don’t run mongodb daemon as root
6. Disable anonymous access – using the ‘auth’ option
7. Encrypt data - “To support audit requirements, you may need to encrypt data stored in MongoDB. For best results you can encrypt this data in the application layer, by encrypting the content of fields that hold secure data.”
8. Encrypt communication – Recommended to use SSL

<http://blog.spiderlabs.com/2013/03/mongodb-security-weaknesses-in-a-typical-nosql-database.html>

Pro tips on staying anonymoose

- VPN
- Tor proxy
- Whonix-Gateway
 - <https://www.whonix.org/>
- McDonalds™ Wifi + 'Big-MAC' changer ;-)

Want to learn more?

- <https://nodesecurity.io/>



Node Security Project

We need a tagline contributor™

[View Advisories](#)

[Report Vulnerability](#)

[Resources](#)

Talks, blog posts, articles and papers that are about or tangentially related to node.js security.

Want to learn more?

- <https://nodegoat.herokuapp.com/tutorial>
- <https://github.com/OWASP/NodeGoat>

GitHub

This repository ▾

Search or type a command



Explore Features Enterprise Blog

Sign up

Sign in

JBLIC



OWASP / NodeGoat

★ Star

41

Fork

10

The OWASP NodeGoat project provides an environment to learn how OWASP Top 10 security risks apply to web applications developed using Node.js and how to effectively address them.

https://www.owasp.org/index.php/Projects/OWASP_Node_js_Goat_Project

54 commits

2 branches

1 release

5 contributors



branch: master ▾

NodeGoat / +



Use MONGODB_URL from env variable if provided



ckarande authored on 7 May

latest commit 10cc6b06de

<> Code

Issues

6

Pull Requests

0

Pulse

Graphs

References

- https://www.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents
- https://media.blackhat.com/bh-us-11/Sullivan/BH_US_11_Sullivan_Server_Side_WP.pdf
- <http://nodesecurity.io/>
- http://asfws12.files.wordpress.com/2012/11/node_security_presentation_v3_asfws.pdf
-