



# **Information Security with Helmet.js / Quality Assurance Testing with Chai**

Securing Node.js Applications

**What is Helmet.js?**

# First what's an http header...

```
GET /tutorials/other/top-20-mysql-best-practices/ HTTP/1.1
Host: code.tutsplus.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5 (.NET CLR 3.5.30729)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: PHPSESSID=r2t5uvjq435r4q7ib3vtdjq12
0
Pragma: no-cache
Cache-Control: no-cache
```

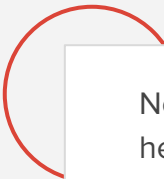
- When you type in a URL, your browser sends an http request

# After that request, your browser sends an http response... That looks like this


- Importance: when someone searches for your website, the http response from your website can leak important information.

```
HTTP/1.x 200 OK
Transfer-Encoding: chunked
Date: Sat, 28 Nov 2009 04:36:25 GMT
Server: LiteSpeed
Connection: close
X-Powered-By: W3 Total Cache/0.8
Pragma: public
Expires: Sat, 28 Nov 2009 05:36:25
GMT
Etag: "pub1259380237;gz"
Cache-Control: max-age=3600, public
Content-Type: text/html;
charset=UTF-8
Last-Modified: Sat, 28 Nov 2009
03:50:37 GMT
X-Pingback:
https://code.tutsplus.com/xmlrpc.php
Content-Encoding: gzip
Vary: Accept-Encoding, Cookie,
User-Agent
```

# Helmet.js is...

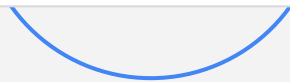


Node.js module that helps secure HTTP headers



HTTP headers provide important metadata about the HTTP, request and response, so the client (browser) & server can send additional information in one transaction.

HTTP headers can leak sensitive information about your applications.



Hackers can exploit vulnerabilities in Express/Node





**How do we  
set up  
Helmet.js?**

## npm install helmet

```
PS C:\Users\lparis\desktop\fcc> npm install helmet@3.23.3
npm WARN saveError ENOENT: no such file or directory, open 'C:\Users\lparis\desktop\fcc\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\lparis\desktop\fcc\package.json'
npm WARN fcc No description
npm WARN fcc No repository field.
npm WARN fcc No README data
npm WARN fcc No license field.

+ helmet@3.23.3
updated 1 package and audited 24 packages in 0.319s
found 0 vulnerabilities

PS C:\Users\lparis\desktop\fcc> |
```

```
1  const express = require('express');
2  const app = express();
3  |  const helmet = require('helmet');
4
5  |  app.use(helmet());
6
7  |  app.use(helmet.hidePoweredBy());
8
```

# Security



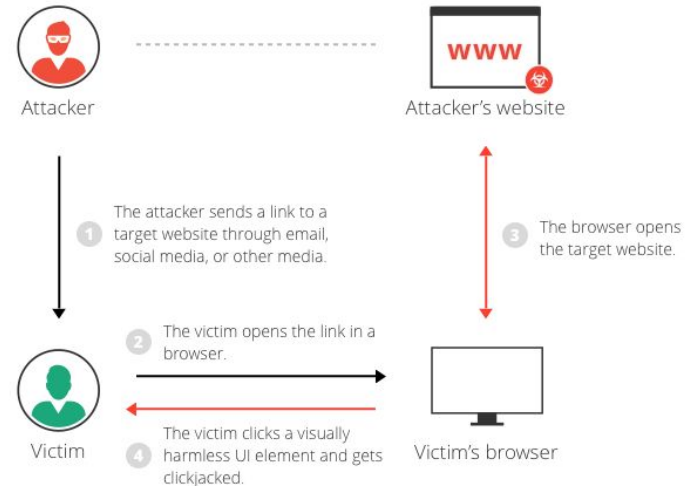
- **helmet.hidePoweredBy()**
- Express leaks information through the X-Powered-By header, which informs the browser which server you're using.  
**'X-Powered-By: Express'**
  - X-Powered-By header: HTTP response header.
    - It can be *disabled* and/or *manipulated* by the server!
- Hackers cross reference that information with publicly disclosed known vulnerabilities. (Easy exploits).
- **helmet.hidePoweredBy90** removes the X-Powered-By header.



# Clickjacking

- **Clickjacking** is a technique of tricking the user into interacting with a page different from what the user thinks it is.
  - `<iframe>` placed in html, used to insert content from another source.
  - Ex: Advertisements
- **Frameguard:** Sets a header instructing the browser to keep an eye out for `<iframe>`. Helps eliminate clickjacking attacks.

```
app.use(helmet.frameguard({ action: 'deny' }));
```



## MIME Type: noSniff

- **MIME sniffing** is a technique used by certain web browsers (Primarily *Internet Explorer*) to examine a websites file format.
  - `helmet.noSniff()`
  - Multipurpose Internet Mail Extension
- **X-Content-Type-Options:** A marker used to identify the MIME types advertised in the content-type.
  - Content-type: a header that tells the client what the content type of returned content actually is.



# PreFetching, Caching, Bcrypt



- DNS **prefetch**: boosts performance by fetching instructions/ data from their original storage in a slower memory to a faster local memory before it's needed.
  - Resolves a website IP address before the user clicks on its link.
  - `dnsPrefetchControl()` - stops DNS prefetch
- Using **noCache** helps boost performance and is useful in development.
- **Bcrypt** is a password-hashing function, which allows us to build secure and unique passwords.
- <https://helmetjs.github.io/>



# What is Chai?

and the browser that can be delightfully paired with any javascript testing framework.

## Download Chai

3.5.0 / 2016-01-28

for Node

Another platform?

Browser

Rails

The **chai** package is available on npm.



### Getting Started

Learn how to install and use Chai through a series of guided walkthroughs.



### API Documentation

Explore the BDD & TDD language specifications for all available assertions.

- Chai is a Javascript testing library that helps you confirm that your website/program still behaves the way you expect it to after you make changes to your code.
- You can write tests to describe your program's requirements and see if your program meets them.
- <https://www.npmjs.com/package/chai>
- <https://www.chaijs.com/>

# How can we use Chai?

- npm install chai
- Styles - assert, expect, or should.
  - Assertion: assertions are particular to certain values. Assertions will fail if the expected value does not match the actual value. Uses dot notation to test!
  - Expect: Allows you to include random/your choice messages to prepend any failed assertions. Typically used with non-descript topics (booleans or numbers).
  - Should: Allows for chainable assertions, similar to *expect*. Although it extends each object with a *should* property. (Issues with internet explorer).

```
1  var chai = require('chai');  
2  var assert = chai.assert;
```

