

Open Source Software Assessment

Authors:

Mrunmayi Churi

Shreyas Damle

Table of Contents

1. Introduction
 - 1.1 Overview
 - 1.2 Scope
 - 1.3 Project Team
2. Code structure
3. Attack vectors
4. Vulnerability Details
 - 4.1 Test methodology
 - 4.2 White Box testing
 - 4.3 Black Box testing
 - 4.4 Design and Security flaws
5. Summary and Future Work
6. References

1. Introduction

1.1 Overview

Free Code Camp is an open source boot camp style website.

Language: JavaScript

This document summarises the findings, analysis and recommendations from the assessment. The goal of this project is to utilise active and passive exploitation techniques in order to evaluate the security of the application and identify application level vulnerabilities.

1.2 Scope

We performed a Web Application Security Assessment of www.freecodecamp.com. The following application URL is defined to be in scope:

www.freecodecamp.com

1.3 Project Team

The project involved contributions from the following team members.

Name	Net id
Mrunmayi Churi	mrc596
Shreyas Damle	sd2917

2. Code Structure

The web application has been developed using MEAN stack. From client to server to database, MEAN is full stack JavaScript. The term MEAN stack refers to a collection of JavaScript based technologies used to develop web applications. It consists of the following:

MongoDB: It is a schemaless NoSQL database system. MongoDB saves data in binary JSON format which makes it easier to pass data between client and server.

Angular.js: The 'A' in MEAN Stack, it is a JavaScript MVC framework developed by Google and is used in front end development. It runs in browser's javascript engine.

Express: It is a lightweight framework used to build web applications in Node.

Node.js: It is a server side JavaScript execution environment for event driven and network applications.

```
[SHREYASs-MacBook-Pro:FreeCodeCamp Shreyas$ tree -L 1
.
├── CONTRIBUTING.md
├── LICENSE.md
├── README.md
├── bower.json
├── client
├── common
├── config
├── gulpfile.js
├── node_modules
├── package.json
├── pm2Start.js
├── public
├── seed
├── server
├── test
├── webpack.config.js
└── webpack.config.node.js

8 directories, 9 files
```

3. Attack vectors

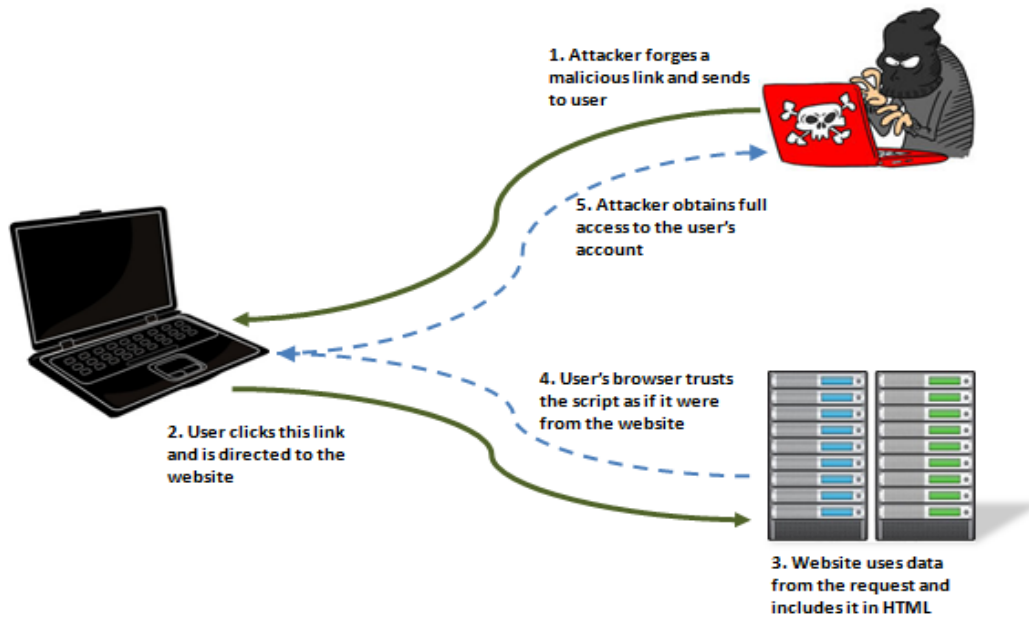
3.1 JavaScript

JavaScript's versatility and sustained performance has resulted in it playing a strategic role in the web development community. However, since its release, there have been several JavaScript security issues that have gained widespread attention. Node.js, which allows JavaScript to be run outside the browser is becoming a platform of choice for building fast and scalable full stack JavaScript applications.

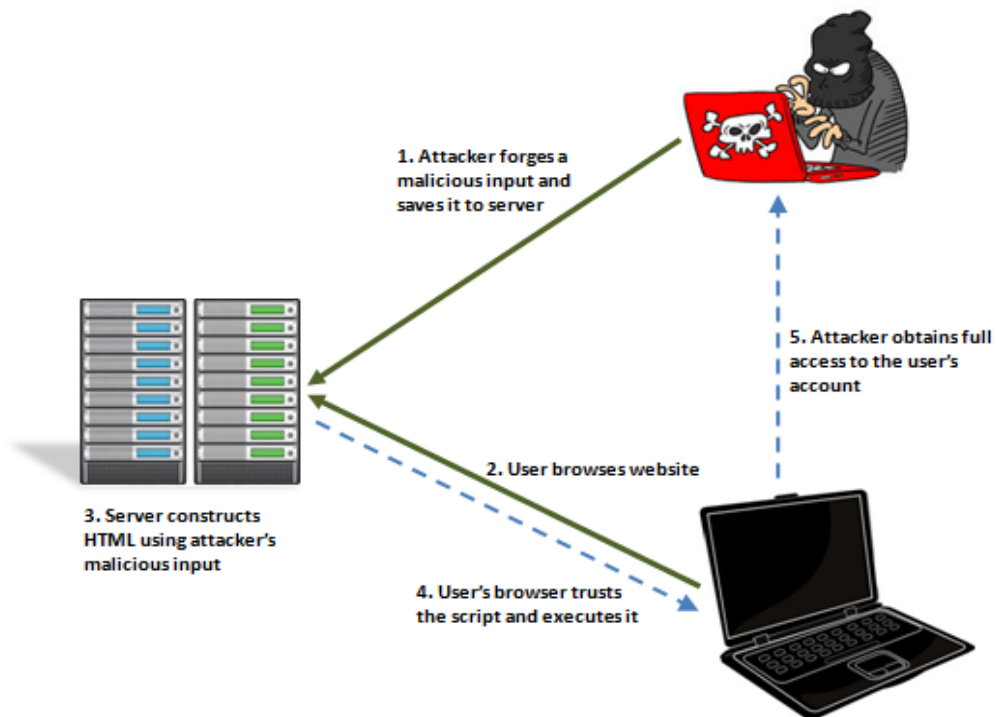
3.1.1 Cross-Site Scripting (XSS)

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site.

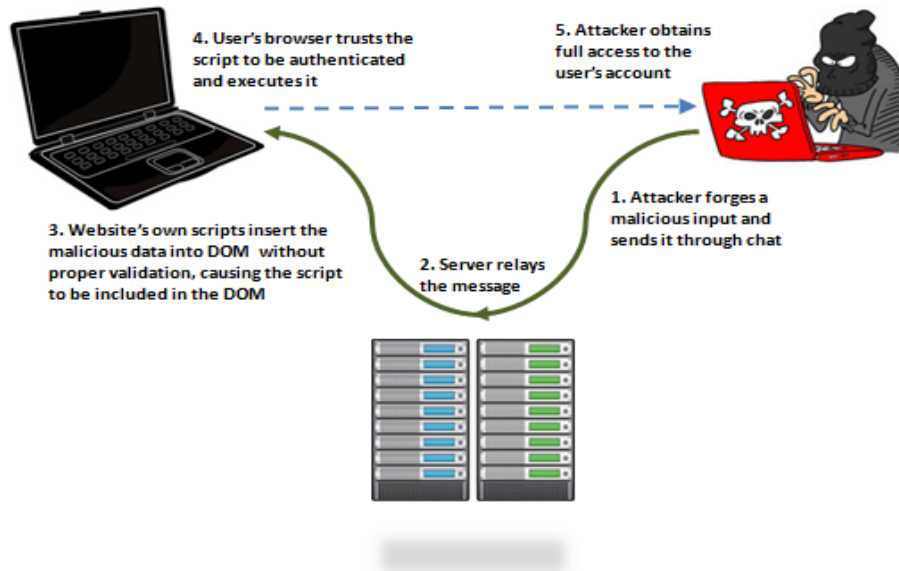
3.1.1.1 Reflected XSS



3.1.1.2 Stored XSS

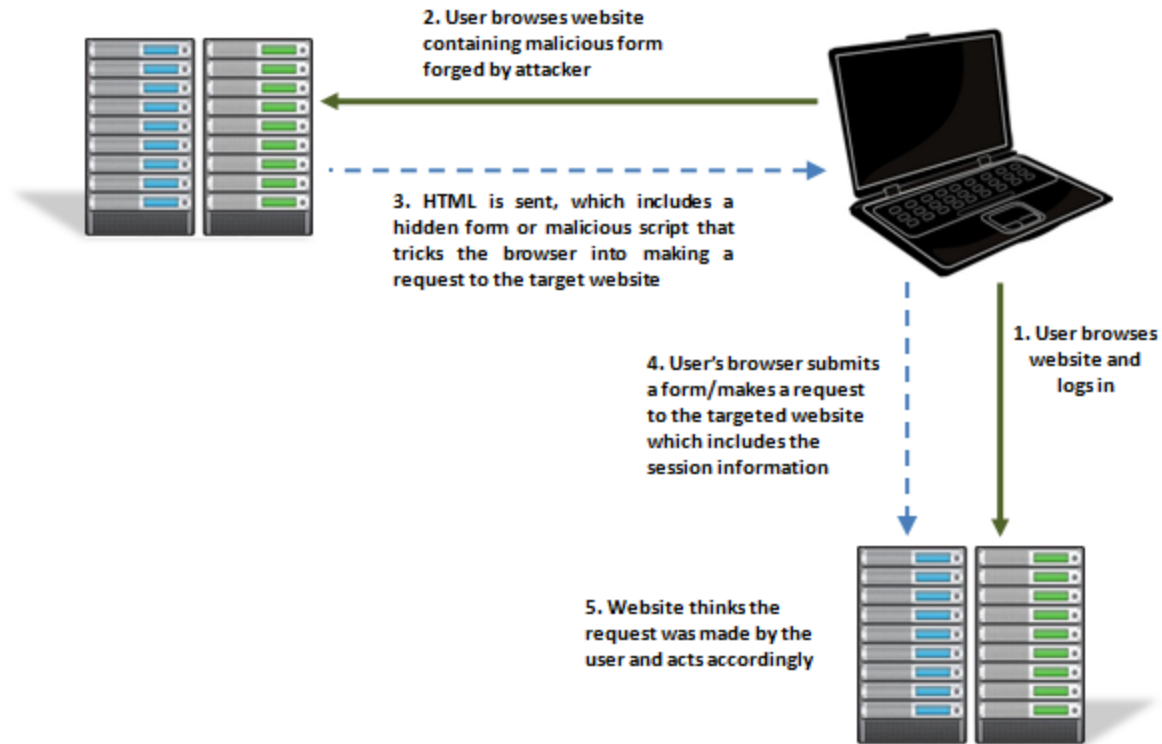


3.1.1.3 DOM XSS



3.1.2 Cross-Site Request Forgery

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. An attacker can use social engineering (such as sending a link via email or chat) and trick unsuspecting users of a web application into executing actions that the attacker wants them to execute. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.



3.1.3 Code Injection

Injection is an attack vector where the attacker introduces, or injects, malicious code into the application to trick the program into executing it. In Node.js commands executed through the **child_process** module, using **exec**, **execFile**, **spawn**, or **fork** can execute scripts on the operating system and can become a possible attack vector for code injection if the commands are incorrectly constructed with user input.

Code injections target the application's functionality which is created and interpreted during runtime based on user input. These can be possible attack points in any application. In Node.js there is an interpreter function to pay extra attention for `eval()`. The attack can be executed by passing JavaScript code in the `eval` function.

Consider the following express code:

```
var express = require('express');
var bodyParser = require('body-parser');
var app = express();
app.use(bodyParser.urlencoded());
app.post('/run', function (req, res {
  eval(req.body.cmd); });
app.listen(80);
```

In the above code, inside the `eval()`, instead of making valid HTTP request, attacker can pass arbitrary JavaScript code like:

```
cmd = while(1)console.log("Infinite loop")
```

This will cause a DoS attack, as application will get stuck inside the infinite loop. This can be more dangerous, if attacker executes any malicious code which can change the functionality of the server code.

Recommendation: Use `Json.parse()`

4. Vulnerability Details

4.1 Test methodology

The following gives a high level description of the methodology followed for performing the assessment



4.2 White box testing

4.2.1 Node Security Project

nsp

We use npm to manage our application dependencies, which is a powerful and convenient way. But these packages may contain critical security vulnerabilities that could affect our application. To ensure this, we checked the production application using nsp command line tool that checks the Node Security Project vulnerability database to determine if the packages used contain any known vulnerabilities.

Vulnerabilities found:

a. Sanitization bypass using HTML Entities

marked is an application that is meant to parse and compile markdown. Due to the way that marked parses input, specifically HTML entities, it's possible to bypass marked's content injection protection (`sanitize: true`) to inject a `javascript:` URL. This flaw exists because `&#xNNanything;` gets parsed to what it could and leaves the rest behind, resulting in just `anything;` being left.


```
SHREYASs-MacBook-Pro:FreeCodeCamp Shreyas$ nsp check
(+) 4 vulnerabilities found
```

	Sanitization bypass using HTML Entities
Name	marked
Installed	0.3.5
Vulnerable	All
Patched	None
Path	freecodecamp@0.1.0 > gulp-notify@2.2.0 > node-notifier@4.5.0 > cli-usage@0.1.2 > marked@0.3.5
More Info	https://nodesecurity.io/advisories/101

CVE-PENDING

Recommendation: There are a couple of options to mitigate the flaw. One of them is to switch to another markdown library such as *remarkable*.

b. Regular Expression Denial of Service

"The Regular expression Denial of Service (ReDoS) is a Denial of Service attack, that exploits the fact that most Regular Expression implementations may reach extreme situations that cause them to work very slowly (exponentially related to input size). An attacker can then cause a program using a Regular Expression to enter these extreme situations and then hang for a very long time."

uglify-js is vulnerable to regular expression denial of service (ReDoS) when certain types of input is passed into `.parse()`.

	Regular Expression Denial of Service
Name	uglify-js
Installed	2.2.5
Vulnerable	<2.6.0
Patched	>=2.6.0
Path	freecodecamp@0.1.0 > jade@1.11.0 > transformers@2.1.0 > uglify-js@2.2.5
More Info	https://nodesecurity.io/advisories/48

CVE-2015-8858

Recommendation: Update to version 2.6.0 or later

ms is vulnerable to regular expression denial of service (ReDoS) when extremely long version strings are parsed.

	Regular Expression Denial of Service
Name	ms
Installed	0.7.0
Vulnerable	<=0.7.0
Patched	>0.7.0
Path	freecodecamp@0.1.0 > react-vimeo@0.1.1 > jsonp@0.2.0 > debug@2.1.3 > ms@0.7.0
More Info	https://nodesecurity.io/advisories/46

CVE-2015-8315

Recommendation: Update to version 0.7.1 or greater. An alternative would be to limit the input length of the user input before passing it into ms.

c. Incorrect Handling of Non-Boolean Comparisons During Minification

In Boolean algebra, DeMorgan's laws describe the relationships between conjunctions (&&), disjunctions (||) and negations (!). In Javascript form, they state that: `!(a && b) === (!a) || (!b)` and `!(a || b) === (!a) && (!b)`. However, the law does not hold true when one of the values is not a boolean. Vulnerable versions of uglify-js do not account for this restriction, and erroneously apply the laws to a statement if it can be reduced in length

by it. When minified with a vulnerable version of uglify-js, it will produce an insecure output, where a token will never expire.

	Incorrect Handling of Non-Boolean Comparisons During Minification
Name	uglify-js
Installed	2.2.5
Vulnerable	<= 2.4.23
Patched	>= 2.4.24
Path	freecodecamp@0.1.0 > jade@1.11.0 > transformers@2.1.0 > uglify-js@2.2.5
More Info	https://nodesecurity.io/advisories/39

CVE-2015-8857

Recommendation: Upgrade uglify-js to version 2.4.24 or later.

4.2.2 Retire.js

Retire.js is a command line scanner that helps identify dependencies with known vulnerabilities in the application. Its goal is to help detect use of version with known vulnerabilities. The following screenshots demonstrate the results of the scan performed on the application using this scanner.

```
SHREYASs-MacBook-Pro:FreeCodeCamp Shreyas$ retire -n --node --nodepath ~/Documents/Spring\ 2016/A
pplication\ Security\FreeCodeCamp/
Downloading https://raw.githubusercontent.com/RetireJS/retire.js/master/repository/npmrepository.
json ...
ms 0.6.2 has known vulnerabilities: severity: medium; summary: Regular expression denial of serv
ice; https://nodesecurity.io/advisories/46
freecodecamp 0.1.0
└─ browser-sync 2.11.1
  └─ socket.io 1.3.7
    └─ debug 2.1.0
      └─ ms 0.6.2
ms 0.6.2 has known vulnerabilities: severity: medium; summary: Regular expression denial of serv
ice; https://nodesecurity.io/advisories/46
freecodecamp 0.1.0
└─ browser-sync 2.11.1
  └─ socket.io 1.3.7
    └─ socket.io-adapter 0.3.1
      └─ debug 1.0.2
        └─ ms 0.6.2
ws 0.8.0 has known vulnerabilities: https://nodesecurity.io/advisories/67
freecodecamp 0.1.0
└─ browser-sync 2.11.1
  └─ socket.io 1.3.7
    └─ socket.io-client 1.3.7
      └─ engine.io-client 1.5.4
        └─ ws 0.8.0
ms 0.6.2 has known vulnerabilities: severity: medium; summary: Regular expression denial of serv
ice; https://nodesecurity.io/advisories/46
freecodecamp 0.1.0
└─ browser-sync 2.11.1
```

```

    ↳ ws 0.8.0
ms 0.6.2 has known vulnerabilities: severity: medium; summary: Regular expression denial of serv
ice; https://nodesecurity.io/advisories/46
freecodecamp 0.1.0
↳ browser-sync 2.11.1
    ↳ socket.io 1.3.7
        ↳ socket.io-client 1.3.7
            ↳ engine.io-client 1.5.4
                ↳ debug 1.0.4
                    ↳ ms 0.6.2
ws 0.8.0 has known vulnerabilities: https://nodesecurity.io/advisories/67
freecodecamp 0.1.0
↳ browser-sync 2.11.1
    ↳ socket.io 1.3.7
        ↳ engine.io 1.5.4
            ↳ ws 0.8.0
ms 0.6.2 has known vulnerabilities: severity: medium; summary: Regular expression denial of serv
ice; https://nodesecurity.io/advisories/46
freecodecamp 0.1.0
↳ browser-sync 2.11.1
    ↳ socket.io 1.3.7
        ↳ engine.io 1.5.4
            ↳ debug 1.0.3
                ↳ ms 0.6.2
connect 1.9.2 has known vulnerabilities: https://nodesecurity.io/advisories/methodOverride_Middl
eware_Reflected_Cross-Site_Scripting
freecodecamp 0.1.0
↳ browser-sync 2.11.1
    ↳ browser-sync-ui 0.5.18
        ↳ weinre 2.0.0-pre-I0Z7U90V
            ↳ express 2.5.11

```

```

    ↳ connect 1.9.2
qs 0.4.2 has known vulnerabilities: severity: low; advisory: qs_dos_extended_event_loop_blocking
; https://nodesecurity.io/advisories/qs_dos_extended_event_loop_blocking
freecodecamp 0.1.0
↳ browser-sync 2.11.1
    ↳ browser-sync-ui 0.5.18
        ↳ weinre 2.0.0-pre-I0Z7U90V
            ↳ express 2.5.11
                ↳ qs 0.4.2
qs 0.4.2 has known vulnerabilities: severity: low; advisory: qs_dos_extended_event_loop_blocking
; https://nodesecurity.io/advisories/qs_dos_extended_event_loop_blocking
freecodecamp 0.1.0
↳ browser-sync 2.11.1
    ↳ browser-sync-ui 0.5.18
        ↳ weinre 2.0.0-pre-I0Z7U90V
            ↳ express 2.5.11
                ↳ connect 1.9.2
                    ↳ qs 0.4.2
qs 0.6.6 has known vulnerabilities: severity: low; advisory: qs_dos_extended_event_loop_blocking
; https://nodesecurity.io/advisories/qs_dos_extended_event_loop_blocking
freecodecamp 0.1.0
↳ loopback-testing 1.2.0
    ↳ supertest 0.13.0
        ↳ superagent 0.18.0
            ↳ qs 0.6.6
ms 0.6.2 has known vulnerabilities: severity: medium; summary: Regular expression denial of serv
ice; https://nodesecurity.io/advisories/46
freecodecamp 0.1.0
↳ loopback-testing 1.2.0
    ↳ mocha 1.21.5
        ↳ debug 2.0.0

```

```

└─ superagent 0.18.0
└─ qs 0.6.6
ms 0.6.2 has known vulnerabilities: severity: medium; summary: Regular expression denial of serv
ice; https://nodesecurity.io/advisories/46
freecodecamp 0.1.0
└─ loopback-testing 1.2.0
└─ mocha 1.21.5
└─ debug 2.0.0
└─ ms 0.6.2
ms 0.7.0 has known vulnerabilities: severity: medium; summary: Regular expression denial of serv
ice; https://nodesecurity.io/advisories/46
freecodecamp 0.1.0
└─ react-vimeo 0.1.0
└─ jsonp 0.2.0
└─ debug 2.1.3
└─ ms 0.7.0
uglify-js 2.2.5 has known vulnerabilities: https://github.com/mishoo/UglifyJS2/issues/751 https:
//github.com/tmcw/mdast-uglify-bug https://nodesecurity.io/advisories/48
freecodecamp 0.1.0
└─ jade 1.11.0
└─ transformers 2.1.0
└─ uglify-js 2.2.5
marked 0.3.5 has known vulnerabilities: severity: medium; advisory: Cross-Site Scripting (XSS) D
ue To Sanitization Bypass Using HTML Entities; https://srcclr.com/security/cross-site-scripting-x
ss-due-to/javascript/s-2309
freecodecamp 0.1.0
└─ gulp-notify 2.2.0
└─ node-notifier 4.5.0
└─ cli-usage 0.1.2
└─ marked 0.3.5
SHREYASs-MacBook-Pro:FreeCodeCamp Shreyas$ █

```

4.3 Black box testing

4.3.1 OWASP Zed Attack Proxy (ZAP)

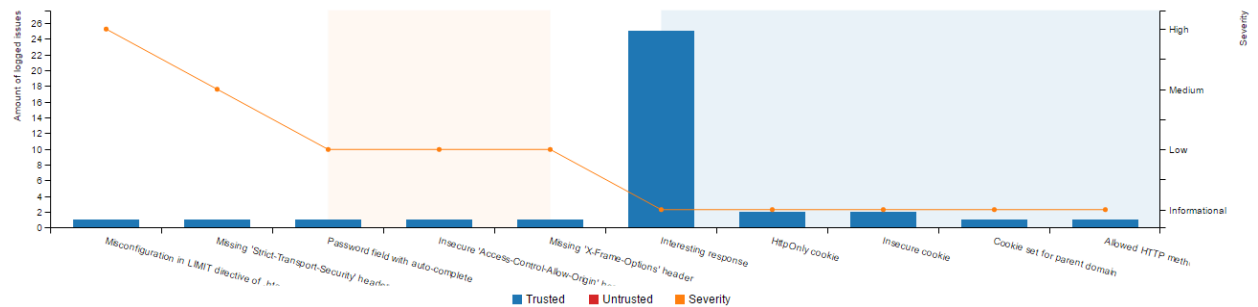
OWASP ZAP is an open-source web application security scanner. When used as a proxy server it allows the user to manipulate all of the traffic that passes through it, including traffic using https.

Medium (Medium)	X-Frame-Options Header Not Set
Description	X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
URL	http://www.freecodecamp.com/auth/github/callback?code=cd7f199961ad763579ac
URL	http://www.freecodecamp.com/robots.txt
URL	http://www.freecodecamp.com
URL	http://www.freecodecamp.com/sitemap.xml
URL	http://www.freecodecamp.com/auth
URL	http://www.freecodecamp.com/auth/github
URL	http://www.freecodecamp.com/news/feed
URL	http://www.freecodecamp.com/challenges/understanding-uninitialized-variables
URL	http://www.freecodecamp.com/coding-bootcamp-cost-calculator
URL	http://www.freecodecamp.com/news
Instances	10

Recommendation: Ensure that the the X-Frame-Options HTTP header is set on all web pages returned by the website

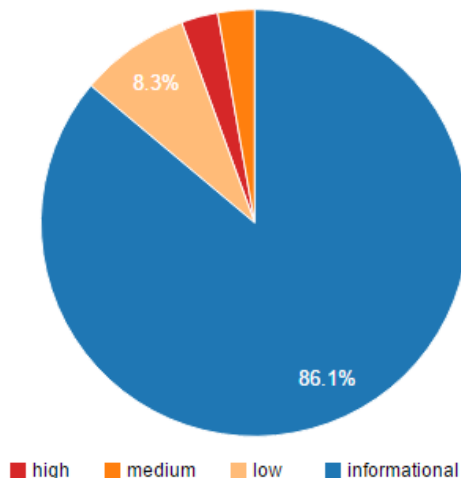
4.3.2 Arachni Scanner

Arachni is a modular, high-performance Ruby framework used to evaluate the security of modern web applications.

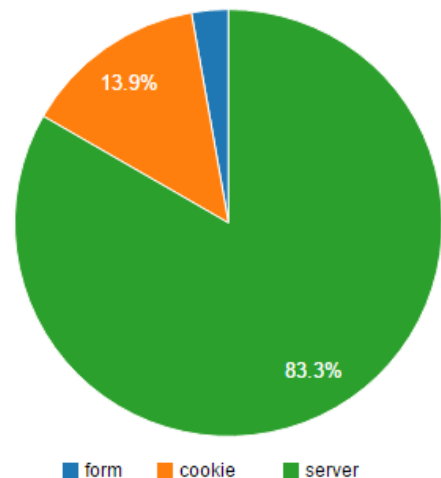


The graph above shows issues arranged according to type, trust and severity. The pie charts demonstrate the severities of issues based on possible impact classifying them into high, medium, low and informational and also illustrate the elements which contain issues classifying them into form, cookie and server.

Severities of issues based on possible impact



Elements with issues, by type



4.3.2.1 High Severity

Misconfiguration in LIMIT directive of .htaccess file

A number of HTTP methods can be used on a webserver (for example GET, POST, etc.). Each of these methods performs a different function, and each has an associated level of risk when their use is permitted on the webserver. The `<Limit>` directive within the `.htaccess` file allows administrators to define which of the methods they would like to block. This is a blacklisting approach and a server administrator may accidentally miss adding certain HTTP methods to be blocked, thus increasing the level of risk to the application and/or server.

Affected page: <https://www.freecodecamp.com/api/users>

Recommendation:

The preferred configuration is to prevent the use of unauthorised HTTP methods by utilising the `<LimitExcept>` directive. This directive uses a whitelisting approach to permit HTTP methods while blocking all others not listed in the directive, and will therefore block any method tampering attempts. The only HTTP methods required for most scenarios are GET and POST. An example of permitting these HTTP methods is : `<LimitExcept POST GET> require valid-user </LimitExcept>`

4.3.2.2 Medium Severity

Missing 'Strict-Transport-Security' header

HTTP Strict Transport Security (HSTS) is an optional response header that can be configured on the server to instruct the browser to only communicate via HTTPS. This will be enforced by the browser even if the user requests a HTTP resource on the same server. Cyber-criminals will often attempt to compromise sensitive information passed from the client to the server using HTTP. This can be conducted via various Man-in-The-Middle (MiTM) attacks or through network packet captures. This application is using HTTPS however does not use the HSTS header.

Affected page: <https://www.freecodecamp.com/>

Recommendation:

The *Strict-Transport-Security* header should be configured on the server. One of the options for this header is max-age, which is a representation (in milliseconds) determining the time in which the client's browser will adhere to the header policy.

4.3.2.3 Low Severity

Password field with auto-complete

When *autocomplete* enabled (default), the browser is allowed to cache previously entered form values. This could allow an attacker with access to the victim's computer the ability to have the victim's credentials automatically entered as the attacker visits the affected page.

Affected page: <https://www.freecodecamp.com/email-signin>

Recommendation: Disable the *autocomplete* attribute on the `<form>` HTML tag. This will disable *autocomplete* for all inputs within that form. An example of disabling *autocomplete* within the form tag is `<form autocomplete=off>`.

Missing 'X-Frame-Options' header

The server didn't return an X-Frame-Options header which means that this website could be at risk of a clickjacking attack. The X-Frame-Options HTTP response header can be used to indicate whether or not a browser should be allowed to render a page inside a frame or iframe.

Affected page: <https://www.freecodecamp.com/>

Recommendation: Configure the web server to include an X-Frame-Options header.

4.3.3 Manual testing

We performed manual and static analysis of code and found potential issue in the program calculator.js. While analysing source and sinks, we observed active sink in the code, which can be potential cause for the DOM based XSS. Sinks are the points in the flow of data at which the untrusted input gets outputted on the page or executed by JavaScript within the page.

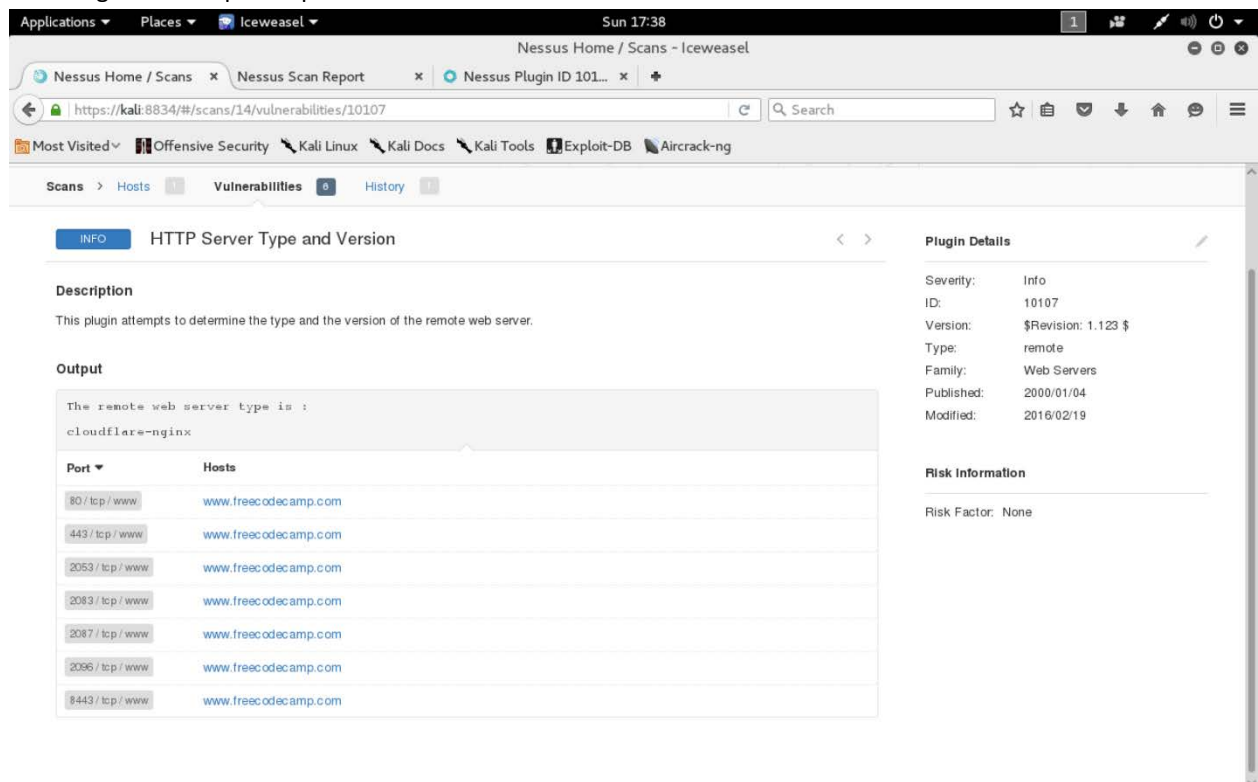
```
var selection = svg.selectAll(".series")
  .data(bootcamps)
  .enter().append("g")
  .attr("class", "series")
  .attr("transform", function (d) {
    return "translate(0," + y0Scale(d.name) + ")";
  });
var rect = selection.selectAll("rect")
  .data(function (d) {
    return d.mapping;
  })
```

In the above code snippet, *selectAll* function has the active sink and it may vulnerable to XSS attacks.

4.4 Design and Security flaws

4.4.1 Server type and version

We used Nessus to check server security and open ports on the server. Here, type and version of the server is visible and the attacker may use this information by exploiting known vulnerabilities against this. The following is the sample output of the nessus scan:



The screenshot shows the Nessus web interface displaying the results of a scan for 'HTTP Server Type and Version'. The interface includes a top navigation bar with 'Applications', 'Places', and 'Iceweasel'. The main content area shows the scan results for the 'HTTP Server Type and Version' plugin. The 'Description' section states: 'This plugin attempts to determine the type and the version of the remote web server.' The 'Output' section shows the remote web server type as 'cloudflare-nginx'. The 'Plugin Details' section on the right lists: Severity: Info, ID: 10107, Version: \$Revision: 1.123 \$, Type: remote, Family: Web Servers, Published: 2000/01/04, Modified: 2016/02/19. The 'Risk Information' section shows a Risk Factor of None. A table of open ports is also displayed, showing ports 80, 443, 2053, 2083, 2087, 2096, and 8443, all of which are open to www.freecodecamp.com.

Port	Hosts
80 / tcp / www	www.freecodecamp.com
443 / tcp / www	www.freecodecamp.com
2053 / tcp / www	www.freecodecamp.com
2083 / tcp / www	www.freecodecamp.com
2087 / tcp / www	www.freecodecamp.com
2096 / tcp / www	www.freecodecamp.com
8443 / tcp / www	www.freecodecamp.com

The screenshot shows the Nessus Home interface in a web browser. The browser tabs include 'Nessus Home / Scans', 'Nessus Scan Report', and 'Nessus Plugin ID 101...'. The address bar shows the URL 'https://kali:8834/#/scans/14/vulnerabilities/43111'. The main content area displays the details for the 'HTTP Methods Allowed (per directory)' plugin. The 'Description' section explains that the plugin tests for various HTTP methods. The 'Output' section shows a list of allowed methods: ACL, BCOPY, BDELETE, BMOVE, BPROPFIND, BPROPPATCH, CHECKIN, CHECKOUT, CONNECT, COPY, DEBUG, DELETE, GET, HEAD, INDEX, LABEL, LOCK, MERGE, MHEAD, MLOCK, MMOVE, MPUT, MPOST, MPROPFIND, MPROPPATCH, MPUT, MREPORT, MRPC_IN_DATA, MRPC_OUT_DATA, MSEARCH, MSUBSCRIBE, MUNCHECKOUT, M_UNLOCK, M_UNSUBSCRIBE, M_UPDATE. The 'Plugin Details' sidebar on the right shows the severity as 'Info', ID as '43111', version as '\$Revision: 1.7\$', type as 'remote', family as 'Web Servers', published date as '2009/12/10', and modified date as '2013/05/09'. The 'Risk Information' section shows a risk factor of 'None'.

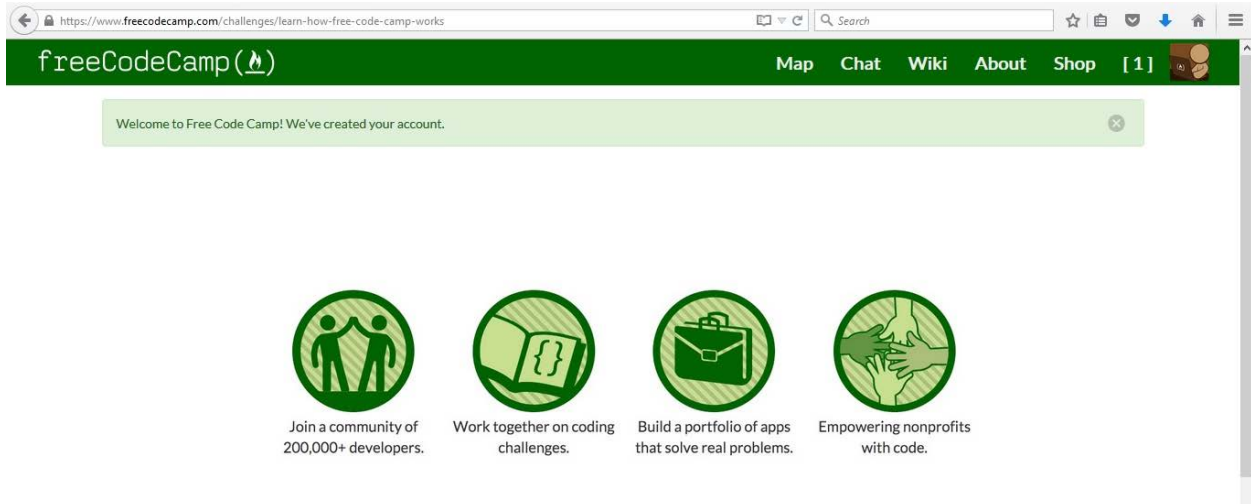
Recommendation: To stop outputting Nginx version, remove the Server header completely, compile Nginx with the Headers More module in, as the header is hard coded in the Nginx source, and this module allows changing any http headers.

more_clear_headers Server;

4.4.2 No email verification

The screenshot shows the freeCodeCamp website's email sign-up page. The browser address bar shows the URL 'https://www.freecodecamp.com/email-signup'. The page has a green header with the freeCodeCamp logo and navigation links: 'Map', 'Chat', 'Wiki', 'About', 'Shop', and 'Sign in'.

Sign up with an email address here:



An attacker can write a script to create multiple new accounts on the website which could result in a DOS attack by filling database (disk).

Recommendation: Enable verification of the account through email.

5. Summary and future work

Vulnerability assessment is a critical step to determine the threats that any organization can face and to develop a comprehensive plan to mitigate them. After assessing the application, we found that www.freecodecamp.com is in decent shape regarding the security design; however, there is always room for improvement. We plan to notify the website about the vulnerabilities and are also interested in contributing to the project. Although they have addressed most of the security issues presented in this document, the current version of their website may still be vulnerable to some attacks. A successful comprehensive vulnerability assessment program and publishing basic security testing tutorials on their own website will help them in developing a more secure learning environment.

6. References

- [1] <https://github.com/FreeCodeCamp/FreeCodeCamp>
- [2] <https://github.com/Arachni/arachni>
- [3] <https://www.owasp.org/index.php/ZAPpingTheTop10>
- [4] <https://nodesecurity.io/tools>
- [5] https://www.owasp.org/index.php/Regular_expression_Denial_of_Service_-_ReDoS
- [6] <https://github.com/RetireJS/retire.js>
- [7] <https://www.toptal.com/nodejs/top-10-common-nodejs-developer-mistakes>
- [8] <http://scottksmith.com/blog/2015/06/08/secure-node-apps-against-owasp-top-10-injection/>
- [9] https://www.owasp.org/index.php/Projects/OWASP_Node_js_Goat_Project