Question: What Operating System does the website utilise?

Answer :

Operating System

Name:	FreeBSD 8.2-RELEASE
Accuracy:	85%

Ports used

Port-Protocol-State: 21 - tcp - open

OS Classes

Type	Vendor	OS Family	OS Generation	Accuracy
general purpose	FreeBSD	FreeBSD	8.X	85%

Question: What web server software is it running?

Answer:

Software / Version	Category
Apache	Web Servers
Twitter Bootstrap	Web Frameworks
jQuery 1.8.3	JavaScript Frameworks

Question: Is it running a CMS (WordPress, Drupal, etc.?)

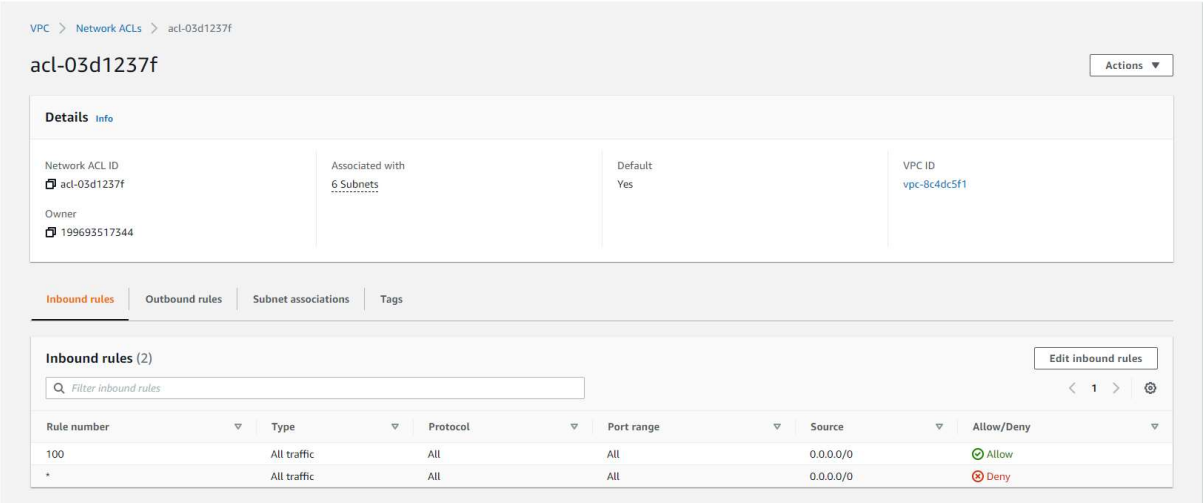
Answer :

FINGERPRINTED CMS & VULNERABILITIES

No CMS were fingerprinted on the website. Information

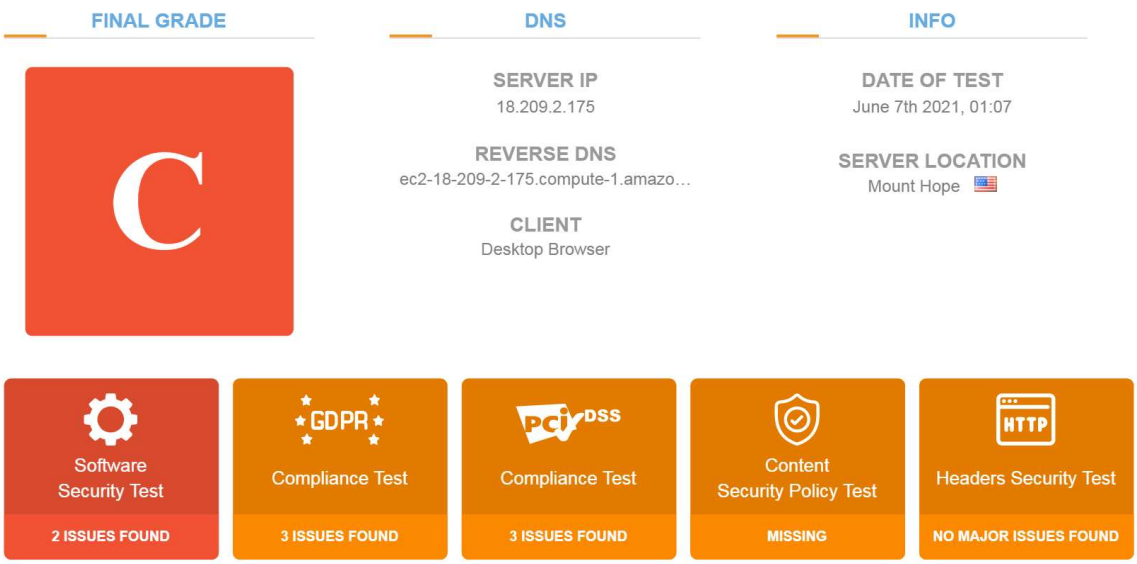
Question: What protection does it have (CDN, Proxy, Firewall?)

Answer: The web application is hosted on a platform that is protected by a rule-based firewall.



Question: Where is it hosted?

Answer:



Question: Does it have any open ports?

Answer :

Open port and their network protocol

Nmap Output | Ports / Hosts | Topology | Host Details | Scans

Port	Protocol	State	Service	Version
21	tcp	open	ftp	
22	tcp	open	ssh	OpenSSH 7.4 (protocol 2.0)
80	tcp	open	tcpwrapped	

Question: Does the site have any known vulnerabilities?

Answer:

Findings

Communication is not secure

URL	Evidence
http://nismphp-env.eba-3mvd2kij.us-east-1.elasticbeanstalk.com/	Communication is made over unsecure, unencrypted HTTP.

Details

Risk description:
The communication between the web browser and the server is done using the HTTP protocol, which transmits data unencrypted over the network. Thus, an attacker who manages to intercept the communication at the network level, is able to read and modify the data transmitted (including passwords, secret tokens, credit card information and other sensitive data).

Recommendation:
We recommend you to reconfigure the web server to use HTTPS - which encrypts the communication between the web browser and the server.

Missing security header: Content-Security-Policy

URL	Evidence
http://nismphp-env.eba-3mvd2kij.us-east-1.elasticbeanstalk.com/	Response headers do not include the HTTP Content-Security-Policy security header

Details

Risk description:
The Content-Security-Policy (CSP) header activates a protection mechanism implemented in web browsers which prevents exploitation of Cross-Site Scripting vulnerabilities (XSS). If the target application is vulnerable to XSS, lack of this header makes it easily exploitable by attackers.

Recommendation:
Configure the Content-Security-Header to be sent with each HTTP response in order to apply the specific policies needed by the application.

Read more about CSP:
https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy

Classification:
CWE : CWE-693
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Software / Version	Category
Apache	Web Servers
Twitter Bootstrap	Web Frameworks
jQuery 1.8.3	JavaScript Frameworks

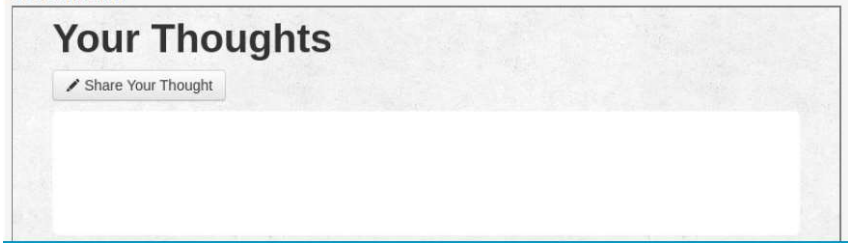
Details

Risk description:
An attacker could use this information to mount specific attacks against the identified software type and version.

Recommendation:
We recommend you to eliminate the information which permits the identification of software platform, technology, server and operating system: HTTP server headers, HTML meta information, etc.

More information about this issue:
https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/01-Information_Gathering/02-Fingerprint_Web_Server.html.

Screenshot:



Question: What versions of software is it using? Are these patched so that they are up to date?

Answer: jQuery 1.8.3 – JavaScrip Frameworks. jQuery 1.9.1 is recommended as remediation.

HIGH SEVERITY

Arbitrary Code Injection

Vulnerable module: xmlhttprequest
Introduced through: xmlhttprequest@1.4.2

Detailed paths

- **Introduced through**: jquery@1.8.3 › xmlhttprequest@1.4.2

  Remediation: Upgrade to jquery@1.9.1.

Overview

xmlhttprequest is a wrapper for the built-in http client to emulate the browser XMLHttpRequest object.

Affected versions of this package are vulnerable to Arbitrary Code Injection. Provided requests are sent synchronously (`async=False` on `xhr.open`), malicious user input flowing into `xhr.send` could result in arbitrary code being injected and run.

POC

```
const { XMLHttpRequest } = require("xmlhttprequest")

const xhr = new XMLHttpRequest()
xhr.open("POST", "http://localhost.invalid/", false /* use synchronize request */)
xhr.send("\\');require(\"fs\").writeFileSync(\"/tmp/aaaaa.txt\", \"poc-20210306\");req.end();//")
```

Arbitrary Code Injection vulnerability report