SECURITY ASSESSMENT

Udajuicer`s Web-Application

Submitted to: The Development Team
Security Analyst: Haneen Alamoudi

Date of Testing: 1/24/2023
Date of Report Delivery: 1/30/2023

Table of Contents

Contents

Security Engagement Summary

Engagement Overview

The Development Team has requested the Information Security Department to perform a vulnerability assessment of a legacy web-application to help them understand what security risk the web-application is posing to the organization, and what mitigations are possible to increase the security posture and reduce the risk to the organization. This assessment is completed weekly and on request in case changes was preformed to the systems.

Scope

The scope of this engagement is to perform a vulnerability assessment of Udajuicer`s legacy web-application (http://127.0.0.1:3000/). The web-application is the core of the business but it`s obsolete and outdated which may expose the organization to multiple risk that may affect the organization's data integrity and continuity.

Executive Risk Analysis

After completing the assessment, the overall risk for the report is High given the importance of the web application to the business continuity and overall reputation. Moreover, multiple vulnerabilities were identified when preforming the assessment, including one high, two medium, and one low vulnerability which can be summaries in the table below.

Vulnerability	Risk	Impact
Cloud Metadata Potentially Exposed	High	It would allow an attacker to completely compromise the system.
Content Security Policy (CSP) Header Not Set	Medium	Data theft, site defacement, and distribution of malware.
Cross-Domain Misconfiguration	Medium	An attacker could access confidential data.
Cross-Domain JavaScript Source File Inclusion	Low	An attacker could insert malicious functionality into the program.

Executive Recommendation

The Scan shows multiple vulnerabilities, the following list shows the order in which the vulnerability should be mitigated and solved:

1. The highest risk is that the cloud metadata is potentially exposed which may allow an attacker to completely compromise the system, it could be mitigated by correctly configuring the application, the periodical review of the configuration and establishing a change management process.
2. CSP Header Not Set vulnerability can be mitigated by configuring all servers to set the Content-Security-Policy header.
3. Cross-Domain Misconfiguration can be mitigated by configuring the "Access-Control-Allow-Origin" HTTP header, or removing all CORS headers entirely.
4. Cross-Domain JavaScript Source File Inclusion can be mitigated by ensuring JavaScript source files are loaded from only trusted sources.

Significant Vulnerability Summary

High Risk Vulnerabilities

- Cloud Metadata Potentially Exposed

Medium Risk Vulnerabilities

- Content Security Policy (CSP) Header Not Set
- Cross-Domain Misconfiguration

Low Risk Vulnerabilities

- Cross-Domain JavaScript Source File Inclusion

Significant Vulnerability Detail

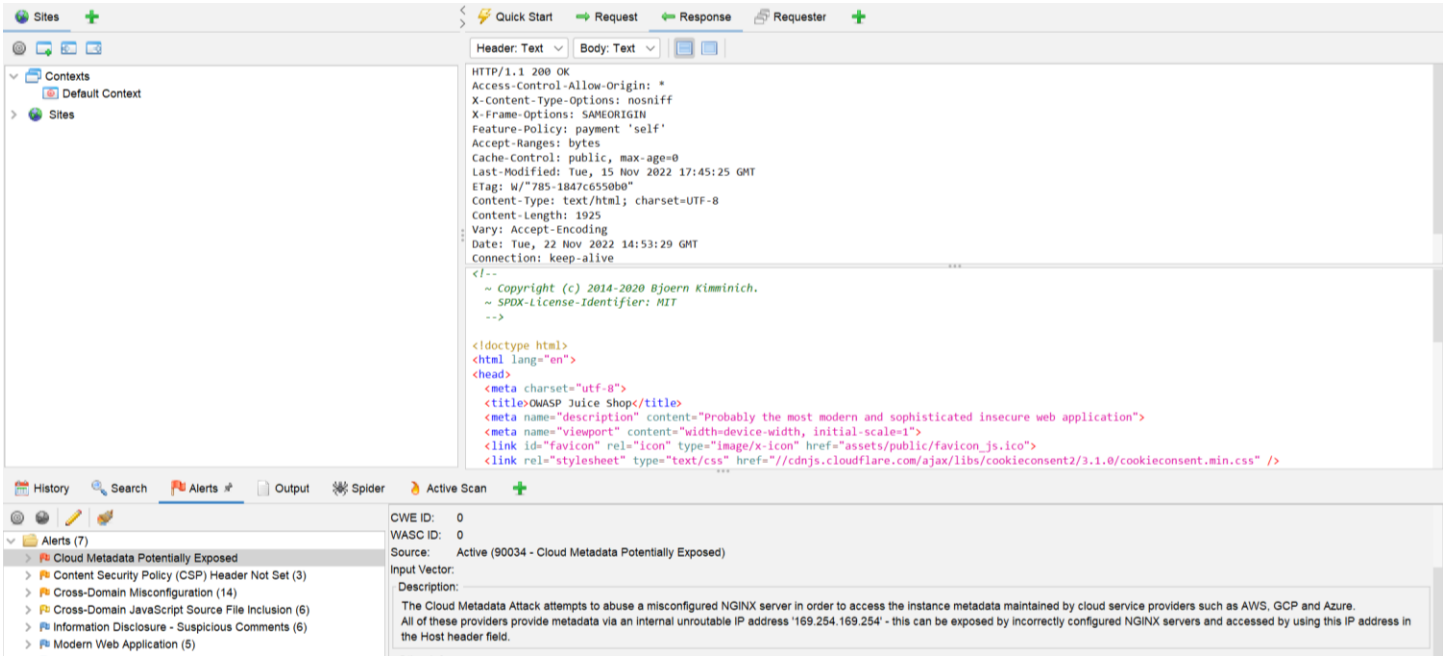
Cloud Metadata Potentially Exposed

HIGH

Vulnerability Details

The Cloud Metadata Attack attempts to abuse a misconfigured NGINX server in order to access the instance metadata maintained by cloud service providers such as AWS, GCP and Azure. All of these providers provide metadata via an internal unroutable IP address '169.254.169.254' - this can be exposed by incorrectly configured NGINX servers and accessed by using this IP address in the Host header field.

Evidence:



The probability of exploit/attack:

The meta data returned can include information that would allow an attacker to completely compromise the system.

Remediation:

The web application shall not trust any user data in NGINX configs. In this case it is probably the use of the $host variable which is set from the 'Host' header and can be controlled by an attacker. Also, the following step can be used to mitigate this vulnerability:

- A repeatable hardening process that makes it fast and easy to deploy another environment that is properly locked down. This process should be automated to minimize the effort required to setup a new secure environment.
- A minimal platform without any unnecessary features, components, documentation, and samples.
- A task to review and update the configurations appropriate to all security notes, updates and patches.
- A segmented application architecture that provides effective, secure separation between components.
- Sending security directives to clients, e.g. Security Headers.
- An automated process to verify the effectiveness of the configurations and settings in all environments

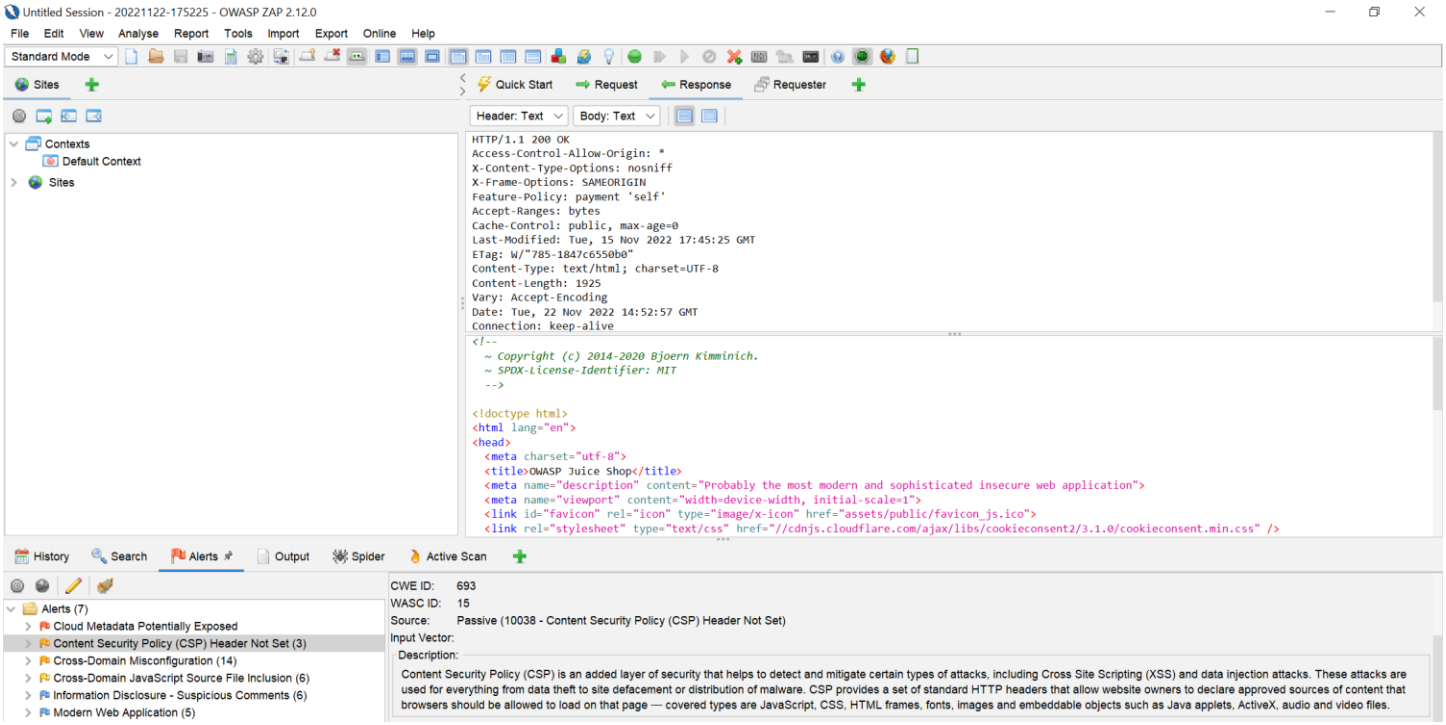
Content Security Policy (CSP) Header Not Set

MEDIUM

Vulnerability Details

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

Evidence:



The probability of exploit/attack:

This Vulnerability will could impact the business in various ways including the following:

- Data theft which may affect Udajuicer integrity, confidentiality, and reputation.
- Site defacement
- Distribution of malware which will impact the business-continuity.

Remediation:

Configure all servers to set the Content-Security-Policy header, to achieve optimal browser support: "Content-Security-Policy" for Chrome 25+, Firefox 23+ and Safari 7+, "X-Content-Security-Policy" for Firefox 4.0+ and Internet Explorer 10+, and "X-WebKit-CSP" for Chrome 14+ and Safari 6+.

Cross-Domain Misconfiguration

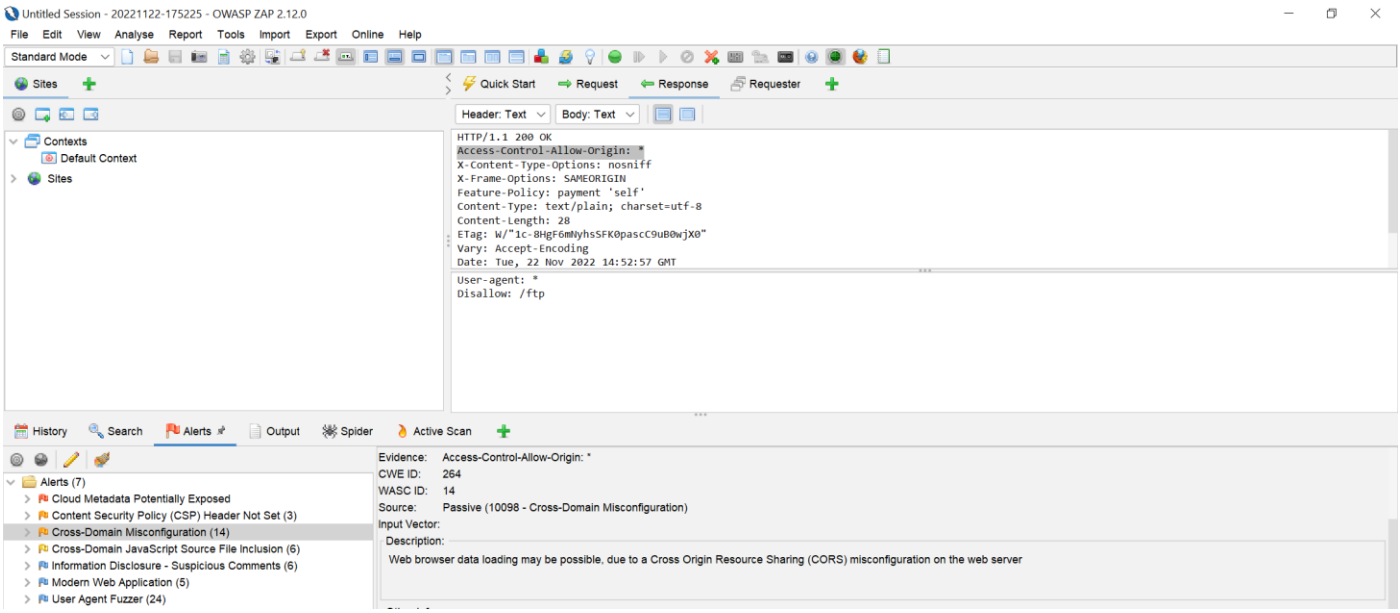
MEDIUM

Vulnerability Details

Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server as shown in the following configuration

```
Access-Control-Allow-Origin: *
```

Evidence:



The probability of exploit/attack:

- This misconfiguration could be used by an attacker to access data that is available in an unauthenticated manner.

Remediation:

- Ensure that sensitive data is not available in an unauthenticated manner.
- Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.

Cross-Domain JavaScript Source File Inclusion

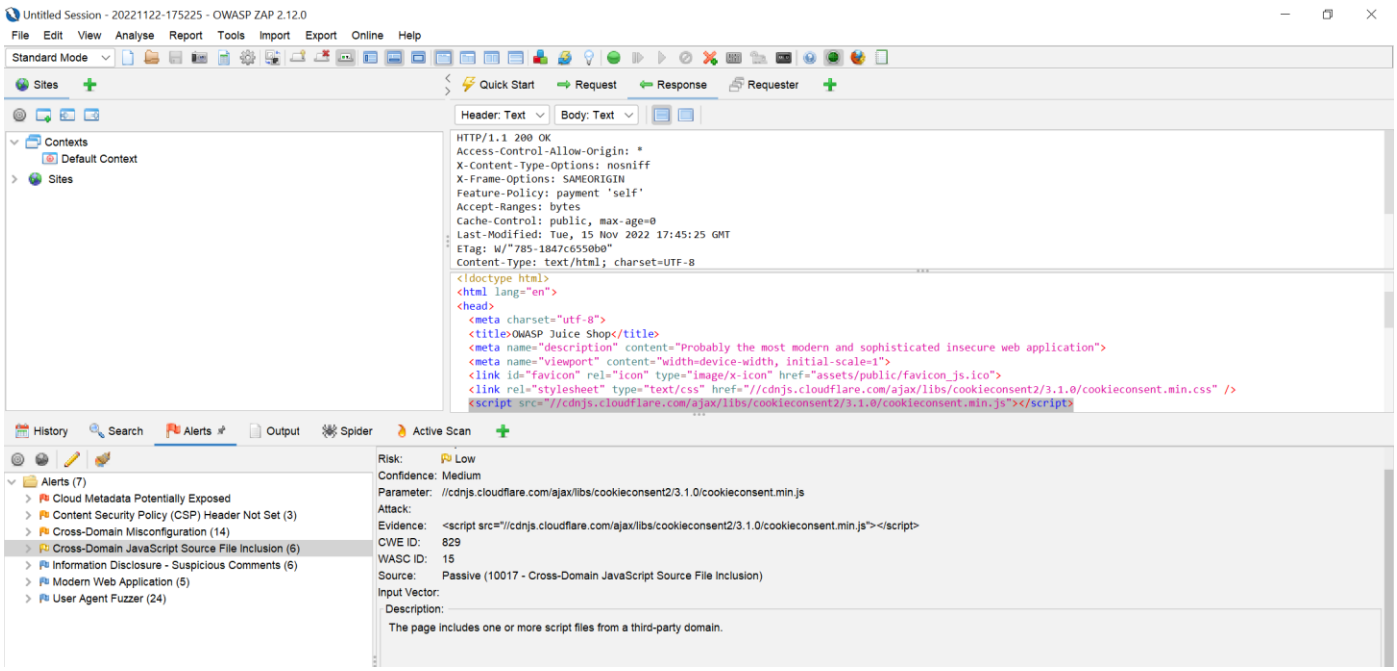
LOW

Vulnerability Details

The configuration includes one script files from a third-party domain.

```
<script
src="//cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookiec
onsent.min.js"></script>
```

Evidence:



The probability of exploit/attack:

An attacker could insert malicious functionality into the program by causing the program to download code that the attacker has placed into the untrusted control sphere, such as a malicious web site.

Remediation:

- Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.

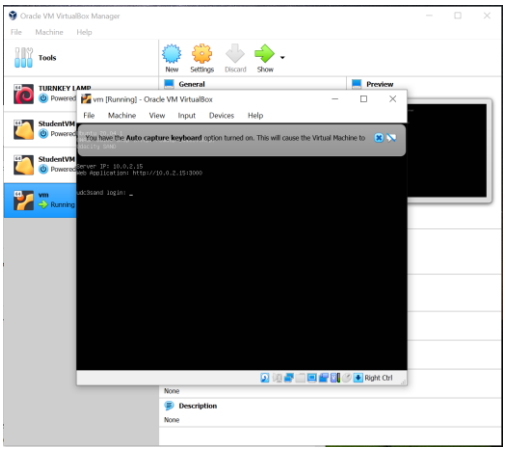
Methodology

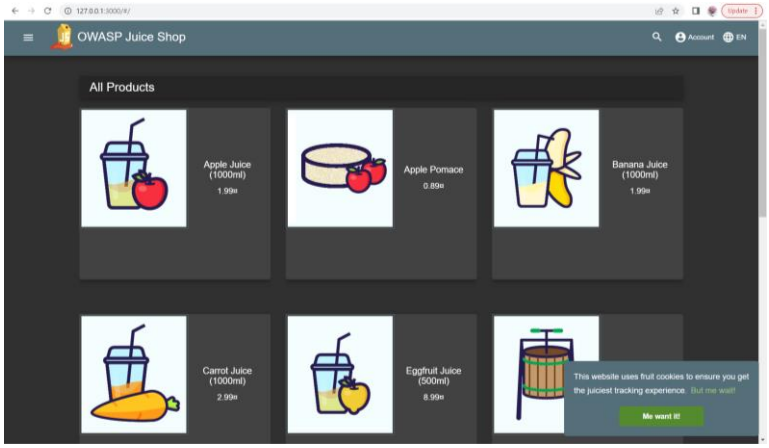
Assessment Toolset Selection

Since the assessment is done to a web-application, the assessment was completed using OWASP ZAP.

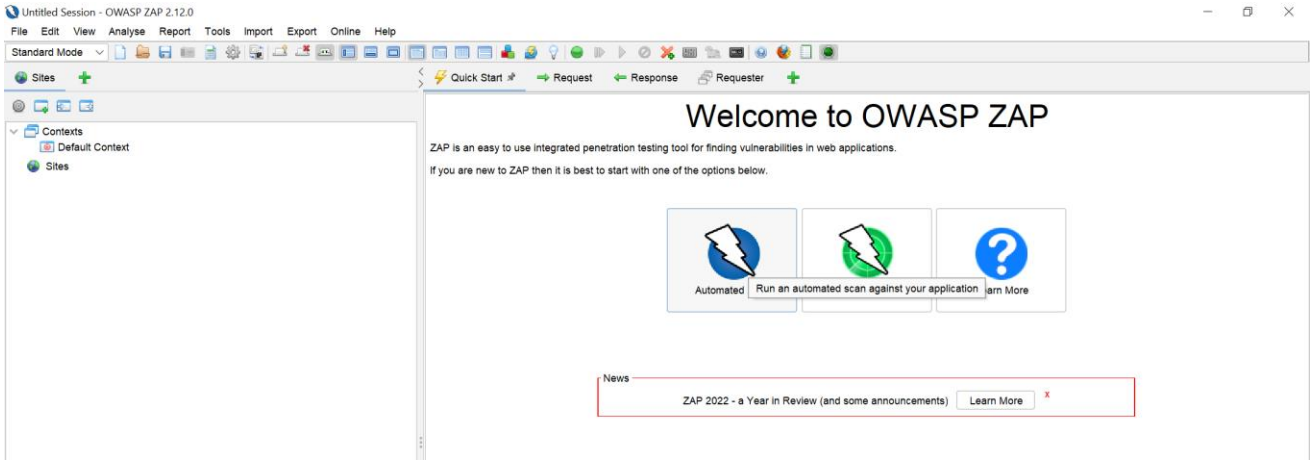
Assessment Methodology Detail

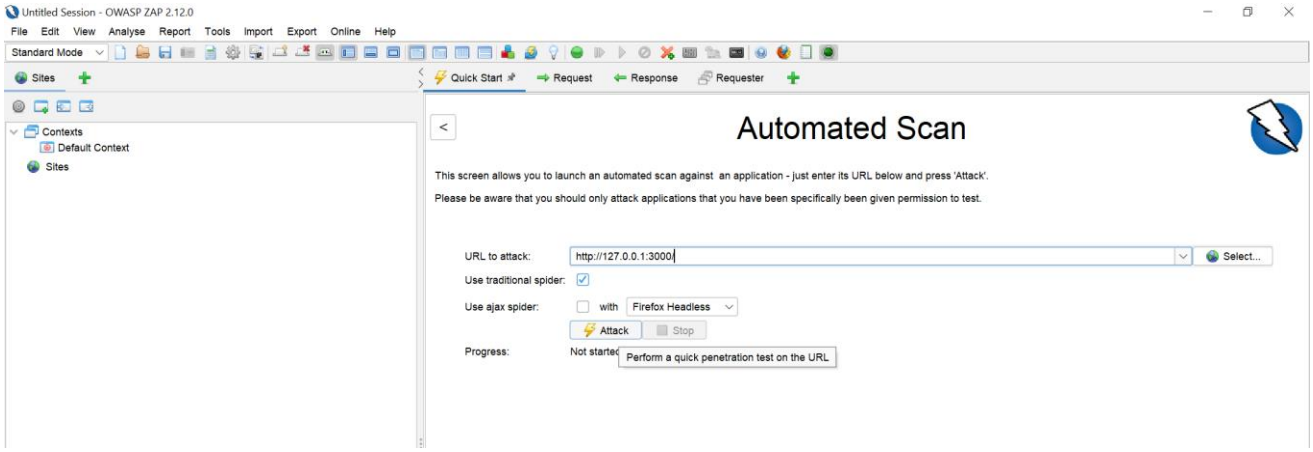
Using Oracle VM, first the snapshot of the web-application VM was started, and the web application is working properly.



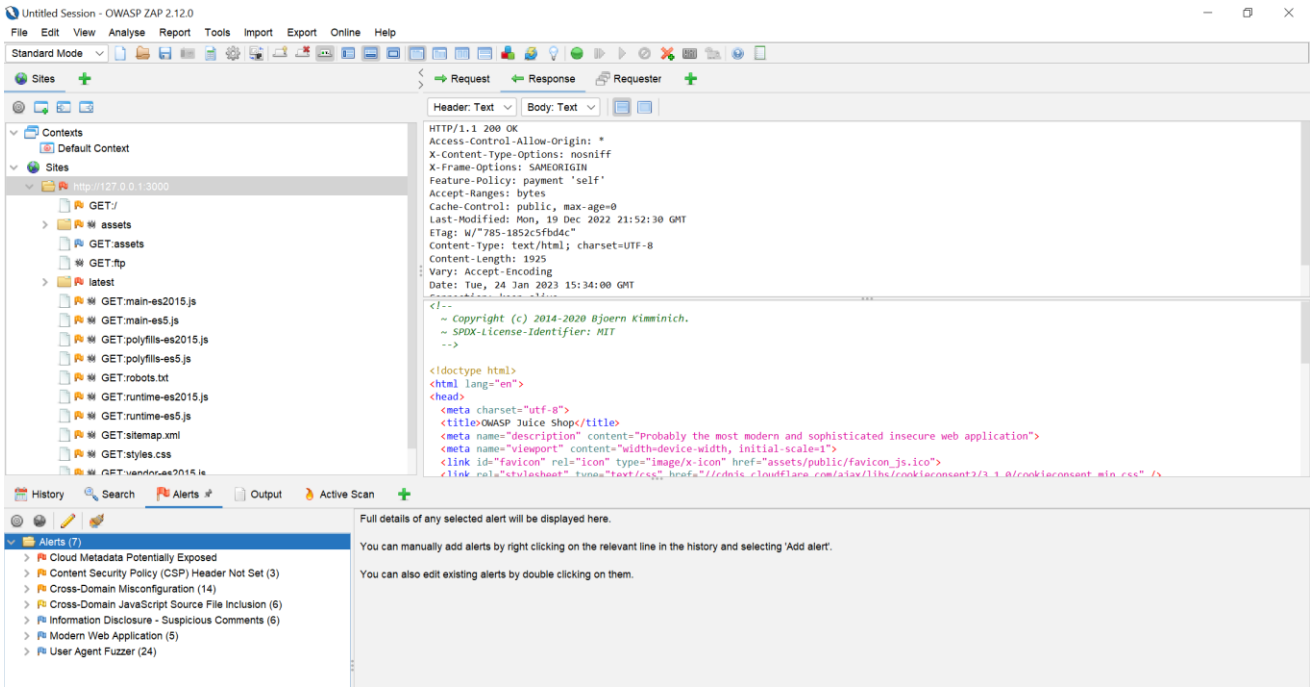


Next, using OWASP ZAP, a penetration test was preformed using the web-application URL (http://127.0.0.1:3000/).





The following screenshot shows a sample result of the test.



The following table shows the used request during the test and response.

Request
GET http://127.0.0.1:3000/latest/meta-data/ HTTP/1.1 Host: 169.154.169.254 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:105.0) Gecko/20100101 Firefox/105.0 Pragma: no-cache Cache-Control: no-cache
Response
HTTP/1.1 200 OK Access-Control-Allow-Origin: * X-Content-Type-Options: nosniff X-Frame-Options: SAMEORIGIN Feature-Policy: payment 'self' Accept-Ranges: bytes Cache-Control: public, max-age=0 Last-Modified: Mon, 19 Dec 2022 21:52:30 GMT ETag: W/"785-1852c5fbd4c" Content-Type: text/html; charset=UTF-8 Content-Length: 1925 Vary: Accept-Encoding Date: Tue, 24 Jan 2023 15:33:27 GMT Connection: keep-alive <!-- ~ Copyright (c) 2014-2020 Bjoern Kimminich. ~ SPDX-License-Identifier: MIT --> <!doctype html> <html lang="en"> <head>

```
  <meta charset="utf-8">
  <title>OWASP Juice Shop</title>
  <meta name="description" content="Probably the most modern and sophisticated insecure
web application">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link id="favicon" rel="icon" type="image/x-icon" href="assets/public/favicon_js.ico">
  <link rel="stylesheet" type="text/css"
href="//cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.css" />
  <script
src="//cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.js"></scrip
t>
  <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
  <script>
    window.addEventListener("load", function(){
      window.cookieconsent.initialise({
        "palette": {
          "popup": { "background": "#546e7a", "text": "#ffffff" },
          "button": { "background": "#558b2f", "text": "#ffffff" }
        },
        "theme": "classic",
        "position": "bottom-right",
        "content": { "message": "This website uses fruit cookies to ensure you get the
juiciest tracking experience.", "dismiss": "Me want it!", "link": "But me wait!",
"href": "https://www.youtube.com/watch?v=9PnbKL3wuH4" }
      })});
  </script>
<link rel="stylesheet" href="styles.css"></head>
<body class="mat-app-background bluegrey-lightgreen-theme">
  <app-root></app-root>
<script src="runtime-es2015.js" type="module"></script><script src="runtime-es5.js"
nomodule defer></script><script src="polyfills-es5.js" nomodule defer></script><script
src="polyfills-es2015.js" type="module"></script><script src="vendor-es2015.js"
type="module"></script><script src="vendor-es5.js" nomodule defer></script><script
src="main-es2015.js" type="module"></script><script src="main-es5.js" nomodule
defer></script></body>
</html>
```

The test resulted in multiple alerts, each will be discussed in details as follow:

- Cloud Metadata Potentially Exposed

Attack	169.154.169.254
Remediation	

- Do not trust any user data in NGINX configs. In this case it is probably the use of the $host variable which is set from the 'Host' header and can be controlled by an attacker.
- A repeatable hardening process that makes it fast and easy to deploy another environment that is properly locked down. This process should be automated to minimize the effort required to setup a new secure environment.
- A minimal platform without any unnecessary features, components, documentation, and samples.
- A task to review and update the configurations appropriate to all security notes, updates and patches.
- A segmented application architecture that provides effective, secure separation between components.
- Sending security directives to clients, e.g. Security Headers.
- An automated process to verify the effectiveness of the configurations and settings in all environments

- Content Security Policy (CSP) Header Not Set

Remediation
• Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header, to achieve optimal browser support: "Content-Security-Policy" for Chrome 25+, Firefox 23+ and Safari 7+, "X-Content-Security-Policy" for Firefox 4.0+ and Internet Explorer 10+, and "X-WebKit-CSP" for Chrome 14+ and Safari 6+. • Do not trust any user data in NGINX configs. In this case it is probably the use of the $host variable which is set from the 'Host' header and can be controlled by an attacker. • A repeatable hardening process that makes it fast and easy to deploy another environment that is properly locked down. This process should be automated to minimize the effort required to setup a new secure environment. • A minimal platform without any unnecessary features, components, documentation, and samples. • A task to review and update the configurations appropriate to all security notes, updates and patches. • A segmented application architecture that provides effective, secure separation between components. • Sending security directives to clients, e.g. Security Headers. • An automated process to verify the effectiveness of the configurations and settings in all environments

- Cross-Domain Misconfiguration

Evidence
`Access-Control-Allow-Origin: *`
Remediation
Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance). Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.

- Cross-Domain JavaScript Source File Inclusion

Evidence
`<script src="//cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.js"></script>`
Remediation
Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.

This concluded the vulnerability assessment methodology portion of this report.