Probely

SECURITY REPORT

Juiceshop - http://juiceshop1234.armorcode.ai:3000

Report generated on Nov. 22, 2022 at 13:18 UTC

Summary

This section contains the scan summary

TARGET http://juiceshop1234.armorcode.ai:3000

Report generated on Nov. 22, 2022 at 13:18 UTC

STARTED	ENDED	DURATION	SCAN PROFILE
Sep. 20, 2022, 07:44 UTC	Sep. 20, 2022, 07:45 UTC	1 minute	Lightning

NUMBER OF FINDINGS

	CURRENT SCAN	FROM LAST SCAN	PENDING FIX
HIGH	0	= 0	0
MEDIUM	1	= 0	1
LOW	2	= 0	2

TOP 5

Missing Content Security Policy header	1
Referrer policy not defined	1
Unencrypted communications	1

Settings

This section contains the summary of settings that were used during this scan

✔ BASIC AUTH

USERNAME	PASSWORD
*********.com	********

✔ SCAN PROFILE

Lightning

Security posture scan. Fast scan that checks vulnerabilities related to SSL/TLS, HTTP security headers, and cookies attributes.

Technical Summary

The following table summarizes the findings, ordered by their severity

#	SEVERITY	VULNERABILITY	STATE
2	MEDIUM	**Unencrypted communications** http://juiceshop1234.armorcode.ai:3000	**NOT FIXED**
1	LOW	**Referrer policy not defined** http://juiceshop1234.armorcode.ai:3000	**NOT FIXED**
3	LOW	**Missing Content Security Policy header** http://juiceshop1234.armorcode.ai:3000	**NOT FIXED**

Exhaustive Test List

The following pages contain the list of vulnerabilities we tested in this scan, taking into consideration the chosen profile

- Cookie without HttpOnly flag
- Missing clickjacking protection
- SSL cookie without Secure flag
- Unencrypted communications
- HSTS header not enforced
- Certificate with insufficient key size or usage, or insecure signature algorithm
- Expired TLS certificate
- Insecure SSL protocol version 3 supported
- Deprecated TLS protocol version 1.0 supported
- Deprecated TLS protocol version 1.1 supported
- Secure TLS protocol version 1.2 not supported
- Weak cipher suites enabled
- Server Cipher Order not configured
- Untrusted TLS certificate
- Heartbleed
- Secure Renegotiation is not supported
- TLS Downgrade attack prevention not supported
- Certificate without revocation information
- HSTS header set in HTTP
- HSTS header with low duration and no subdomain protection
- HSTS header with low duration
- HSTS header does not protect subdomains
- Insecure SSL protocol version 2 supported
- Browser XSS protection disabled
- Browser content sniffing allowed
- Referrer policy not defined
- Insecure referrer policy
- Potential DoS on TLS Client Renegotiation
- TLS certificate about to expire
- Invalid referrer policy
- Missing Content Security Policy header
- Insecure Content Security Policy

Detailed Finding Descriptions

This section contains the findings in more detail, ordered by severity

2 Unencrypted communications

MEDIUM

CVSS SCORE
7.4

CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:N

METHOD	PATH
GET	http://juiceshop1234.armorcode.ai:3000

DESCRIPTION

The application allows clients to connect to it through an unencrypted connection, meaning that an attacker that is strategically positioned between the victim's and the applications's traffic is able to eavesdrop all communications between them, accessing any information that is being transmitted, such as the victim's credentials. In addition, the attacker can modify the communications to deliver more powerful attacks, for instance, to ask the victim for more sensitive information that hasn't been asked in the original application page.

Such attacks are more likely to occur when the victim is using an insecure Wi-Fi connection, a typical scenario in the public Wi-Fi services.

Unencrypted connections may also trigger browser warnings about the insecurity of the connection, following the trend of raising awareness about privacy.

EVIDENCE

No evidence available.

REQUEST

```
GET / HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; +https://probely.com/sos) ProbelyMRKT/0.1.0
Accept: */*
Accept-Encoding: gzip, deflate
Host: juiceshop1234.armorcode.ai:3000
```

RESPONSE

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Last-Modified: Tue, 20 Sep 2022 06:54:22 GMT
ETag: W/"7c3-18359ad24cf"
Content-Type: text/html; charset=UTF-8
Vary: Accept-Encoding
Content-Encoding: gzip
Date: Tue, 20 Sep 2022 09:21:14 GMT
Connection: keep-alive
Keep-Alive: timeout=5
```

```
Transfer-Encoding: chunked
<!--
  ~ Copyright (c) 2014-2022 Bjoern Kimminich & the OWASP Juice Shop
contributors.
  ~ 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/cookieconse
nt.min.js"></script>
  <script
src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
  <script>
```

1

Referrer policy not defined

LOW

CVSS SCORE
3.1

CVSS:3.0/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N

METHOD	PATH
GET	http://juiceshop1234.armorcode.ai:3000

DESCRIPTION

The application does not prevent browsers from sending sensitive information to third party sites in the **referer** header.

Without a referrer policy, every time a user clicks a link that takes him to another origin (domain), the browser will add a **referer** header with the URL from which he is coming from. That URL may contain sensitive information, such as password recovery tokens or personal information, and it will be visible that other origin. For instance, if the user is at `example.com/password_recovery?unique_token=14f748d89d` and clicks a link to `example-analytics.com`, that origin will receive the complete password recovery URL in the headers and might be able to set the users password. The same happens for requests made automatically by the application, such as XHR ones.

Applications should set a secure referrer policy that prevents sensitive data from being sent to third party sites.

EVIDENCE

Response headers, missing the `Referrer-Policy` header:

```
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Last-Modified: Tue, 20 Sep 2022 06:54:22 GMT
ETag: W/"7c3-18359ad24cf"
Content-Type: text/html; charset=UTF-8
Vary: Accept-Encoding
Content-Encoding: gzip
Date: Tue, 20 Sep 2022 09:21:14 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Transfer-Encoding: chunked
```

REQUEST

```
GET / HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; +https://probely.com/sos) ProbelyMRKT/0.1.0
Accept: */*
Accept-Encoding: gzip, deflate
Host: juiceshop1234.armorcode.ai:3000
```

RESPONSE

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Last-Modified: Tue, 20 Sep 2022 06:54:22 GMT
```

```
ETag: W/"7c3-18359ad24cf"
Content-Type: text/html; charset=UTF-8
Vary: Accept-Encoding
Content-Encoding: gzip
Date: Tue, 20 Sep 2022 09:21:14 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Transfer-Encoding: chunked
<!--
  ~ Copyright (c) 2014-2022 Bjoern Kimminich & the OWASP Juice Shop
contributors.
  ~ 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/cookieconse
nt.min.js"></script>
  <script
src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
  <script>
```

3 Missing Content Security Policy header

LOW

CVSS SCORE
3.7

CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N

METHOD	PATH
GET	http://juiceshop1234.armorcode.ai:3000

DESCRIPTION

The Content Security Policy (CSP) is an HTTP header through which site owners define a set of security rules that the browser must follow when rendering their site. The most common usage is to define a list of approved sources of content that the browser can lo ad. This can be used to effectively mitigate Cross-Site Scripting (XSS) and Clickjacking attacks.

CSP is flexible enough for you to define from where the browser can load JavaScript, Stylesheets, images, or fonts, among other op tions. It can also be used in report mode only, a recommended approach before deploying strict rules in a live environment. Howev er, please note that report mode does not protect you, it just logs policy violations.

EVIDENCE

Response headers, missing the Content-Security-Policy header:

```
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Last-Modified: Tue, 20 Sep 2022 06:54:22 GMT
ETag: W/"7c3-18359ad24cf"
Content-Type: text/html; charset=UTF-8
Vary: Accept-Encoding
Content-Encoding: gzip
Date: Tue, 20 Sep 2022 09:21:14 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Transfer-Encoding: chunked
```

REQUEST

```
GET / HTTP/1.1
User-Agent: Mozilla/5.0 (compatible; +https://probely.com/sos) ProbelyMRKT/0.1.0
Accept: */*
Accept-Encoding: gzip, deflate
Host: juiceshop1234.armorcode.ai:3000
```

RESPONSE

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Accept-Ranges: bytes
Cache-Control: public, max-age=0
Last-Modified: Tue, 20 Sep 2022 06:54:22 GMT
ETag: W/"7c3-18359ad24cf"
Content-Type: text/html; charset=UTF-8
Vary: Accept-Encoding
```

```
Content-Encoding: gzip
Date: Tue, 20 Sep 2022 09:21:14 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Transfer-Encoding: chunked
<!--
  ~ Copyright (c) 2014-2022 Bjoern Kimminich & the OWASP Juice Shop
contributors.
  ~ 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/cookieconse
nt.min.js"></script>
  <script
src="//cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js"></script>
  <script>
```

Glossary

Term	Definition
Vulnerability	A type of security weakness that might occur in applications (e.g. Broken Authentication, Information Disclosure). Some vulnerabilities take their name not from the weakness itself, but from the attack that exploits it (e.g. SQL Injection, XSS, etc.).
Findings	An instance of a Vulnerability that was found in an application.

Severity Legend

To each finding is attributed a severity which sums up its overall risk

The severity is a compound metric that encompasses the likelihood of the finding being found and exploited by an attacker, the skill required to exploit it, and the impact of such exploitation. A finding that is easy to find, easy to exploit and the exploitation has high impact, will have a greater severity.

Different findings of the same type could have a different severity: we consider multiple factors to increase or decrease it, such as if the application has an authenticated area or not.

The following table describes the different severities:

Severity	Description	Examples
HIGH	These findings may have a direct impact in the application security, either clients or service owners, for instance by granting the attacker access to sensitive information.	SQL Injection OS Command Injection
MEDIUM	Medium findings usually don't have immediate impact alone, but combined with other findings may lead to a successful compromise of the application.	Cross-site Request Forgery Unencrypted Communications
LOW	Findings where either the exploit is not trivial, the impact is low, or the finding cannot be exploited by itself.	Directory Listing Clickjacking

Category Descriptions

The following pages contain descriptions of each vulnerability. For each vulnerability you will find a section explaining its impact, causes and prevention methods.
These descriptions are very generic, and whenever they are not enough to understand or fix a given finding, more information is provided for that finding in the Detailed Finding Descriptions section.

Unencrypted communications

Description

The application allows clients to connect to it through an unencrypted connection, meaning that an attacker that is strategically positioned between the victim's and the applications's traffic is able to eavesdrop all communications between them, accessing any information that is being transmitted, such as the victim's credentials. In addition, the attacker can modify the communications to deliver more powerful attacks, for instance, to ask the victim for more sensitive information that hasn't been asked in the original application page.

Such attacks are more likely to occur when the victim is using an insecure Wi-Fi connection, a typical scenario in the public Wi-Fi services.

Unencrypted connections may also trigger browser warnings about the insecurity of the connection, following the trend of raising awareness about privacy.

Fix

The application should be configured to use encryption in all communications. This normally means to configure the web server to use TLS with a suitable choice of ciphers, forcing all incoming requests in plaintext (without encryption) to be redirected to TLS endpoint of the application.

In HTTP this is achieved with a 301 Moved Permanently directive. It can be complemented with an HTTP Strict Transport Security header (HSTS) to force the browsers to make all requests in HTTPS even if the end user forgets to write HTTPS in the request.

Referrer policy not defined

Description

The application does not prevent browsers from sending sensitive information to third party sites in the **referer** header.

Without a referrer policy, every time a user clicks a link that takes him to another origin (domain), the browser will add a **referer** header with the URL from which he is coming from. That URL may contain sensitive information, such as password recovery tokens or personal information, and it will be visible that other origin. For instance, if the user is at `example.com/password_recovery?unique_token=14f748d89d` and clicks a link to `example-analytics.com`, that origin will receive the complete password recovery URL in the headers and might be able to set the users password. The same happens for requests made automatically by the application, such as XHR ones.

Applications should set a secure referrer policy that prevents sensitive data from being sent to third party sites.

Fix

This problem can be fixed by sending the header **Referrer-Policy** with a secure and valid value. There are different values available, but not all are considered secure. Please note that this header only supports one directive at a time. The following list explains each one and it is ordered from the safest to the least safe:

- `no-referrer`: never send the header.
- `same-origin`: send the full URL to requests to the same origin (exact scheme + domain)
- `strict-origin`: send only the domain part of the URL, but sends nothing when downgrading to HTTP.
- `origin`: similar to **strict-origin** without downgrade restriction.
- `strict-origin-when-cross-origin`: send full URL within the same origin, but only the domain part when sending to another origin. It sends nothing when downgrading to HTTP.
- `origin-when-cross-origin`: similar to **strict-origin-when-cross-origin** without the downgrade restriction.

Insecure options: * `no-referrer-when-downgrade`: sends the full URL when the scheme does not change. It will send if both origins are, for instance, HTTP. * `unsafe-url`: always sent the full URL

A possible, safe option is `strict-origin`, so the header would look like this:

```
Referrer-Policy: strict-origin
```

It is normally easy to enable the header in the web server configuration file, but it can also be done at the application level.

Please note that the referrer header is written `referer`, with a single `r` but the referrer policy header is properly written, with `rr`: `Referrer-Policy`.

Missing Content Security Policy header

Description

The Content Security Policy (CSP) is an HTTP header through which site owners define a set of security rules that the browser must follow when rendering their site. The most common usage is to define a list of approved sources of content that the browser can load. This can be used to effectively mitigate Cross-Site Scripting (XSS) and Clickjacking attacks.

CSP is flexible enough for you to define from where the browser can load JavaScript, Stylesheets, images, or fonts, among other options. It can also be used in report mode only, a recommended approach before deploying strict rules in a live environment. However, please note that report mode does not protect you, it just logs policy violations.

Fix

You can define a Content Security Policy by setting a header in your application. The header can look like this:

```
Content-Security-Policy: frame-ancestors 'none'; default-src 'self', script-src '*://*.example.com:*'
```

In this example, the **frame-ancestors** directive set to '**none**' indicates that the page cannot be placed inside a frame, not even by itself. The **default-src** defines the loading policy for all resources, in this case, they can be loaded from the current origin (protocol + domain + port). The example sets a more specific policy for scripts, through the **script-src**, restricting script loading to any subdomain of example.com.

The policy can be with different directives, and there are other less strict options for the directives above.