

TP3: TeaStore Security

Group 02

Chun-An Bau
2165883
chun-an.bau@polymtl.ca

Youssef Amine BenDiha
2113648
youssef-amine.ben-diha@polymtl.ca

Jakub Profota
2165556
jakub.profota@polymtl.ca

CONTENTS

I	Static analysis	2
I-A	Manual	2
I-B	Vulnerabilities	2
I-B1	1st vulnerability	2
I-B2	2nd vulnerability	2
I-B3	3rd vulnerability	3
I-B4	4th vulnerability	3
I-C	Security hotspots	3
I-C1	1st security hotspot	3
I-C2	2nd security hotspot	4
I-C3	3rd security hotspot	4
I-C4	4th security hotspot	5
II	Penetration testing	5
II-A	Manual	5
II-B	Alerts	5
II-B1	1st alert	6
II-B2	2nd alert	6
II-B3	3rd alert	6
II-B4	4th alert	6
II-B5	5th alert	6
II-B6	6th alert	6
II-B7	7th alert	7
II-B8	8th alert	7
III	Conslusion	7
	References	7

Abstract

This document focuses on the security rating of TeaStore [1]. We found eight vulnerabilities and security hotspots through static analysis using SonarCloud [2], categorized and described them and recommended solutions. We conducted penetration testing on the deployed system using OWASP ZAP [3] and described eight alerts. We analyzed results from both tools and discussed differences.









I. STATIC ANALYSIS

A. Manual

Setting up SonarCloud is reasonably straightforward. One forks TeaStore, logs in on their website with GitHub account and selects Analyze new project. After authorizing SonarCloud, one selects a repository to analyze; the repository will show up in SonarCloud. Here, one has to finalize the setup by editing corresponding files; SonarCloud shows the instructions. Note that TeaStore's build system is maven. Choose it accordingly. After committing and pushing changes, one can see a status icon next to the commit name on the GitHub website. Clicking on this icon shows the console output of the build process. After waiting a couple of minutes, one should see static analysis results on the SonarCloud page.

B. Vulnerabilities

We used TeaStore version 1.4.1. By inspecting the issues tab of results of the main branch, we can see four vulnerabilities (duplicates not included) being reported; three of those Critical and one of those Minor. Some of the vulnerabilities are mostly duplicate, so we also included some security hotspots. However, these are not categorized according to OWASP, SANS, or CWE standards, so we used SonarCloud's categorization.

▼ Type		▼ Severity			
 Bug	104	 Blocker	0	 Minor	2
 Vulnerability	6	 Critical	4	 Info	0
 Code Smell	509	 Major	0		

1) 1st vulnerability: Change this code to use a stronger protocol.

File: utilities/.../teastore/registryclient/util/RESTClient.java, line 112

Severity: Critical

OWASP: A3 - Sensitive Data Exposure, A6 - Security Misconfiguration

SANS: Porous Defenses

CWE: CWE-295 - Improper Certificate Validation, CWE-326 - Inadequate Encryption Strength, CWE-327 - Use of a Broken or Risky Cryptographic Algorithm

Description: this vulnerability offers attackers the possibility to downgrade the protocol and use a less secure version.




Solution: it is recommended to enforce a specific version of SSL. For example:

```
context = SSLContext.getInstance("TLSv1.2");
```

```
SSLContext sslContext = SSLContext.getInstance("SSL");
```

Change this code to use a stronger protocol. Why is this an issue?

6 months ago ▼ L112 🔗

 Vulnerability ▼  Critical ▼  Open ▼ Not assigned ▼ 2min effort Comment

 No tags ▼

2) 2nd vulnerability: Enable server certificate validation on this SSL/TLS connection.

File: utilities/.../teastore/registryclient/util/RESTClient.java, line 100 and 104

Severity: Critical

OWASP: A3 - Sensitive Data Exposure, A6 - Security Misconfiguration

CWE: CWE-295 - Improper Certificate Validation

Description: X509TrustManager does not validate certificates. SSL connection may not be secure and could be prone to man-in-the-middle attacks.

Solution: provide an appropriate trust store i.e. the method should throw exception when validation fails.

```
@Override
public void checkServerTrusted(java.security.cert.X509Certificate[] x509Certificates, String
```

Enable server certificate validation on this SSL/TLS connection. [Why is this an issue?](#)

6 months ago ▾ L104 🔗

🔒 Vulnerability ▾ 🚨 Critical ▾ 🔵 Open ▾ Not assigned ▾ 5min effort [Comment](#)

🏷 No tags ▾

3) *3rd vulnerability*: Enable server hostname verification on this SSL/TLS connection.

File: utilities/.../teastore/registryclient/util/RESTClient.java, line 115

Severity: Critical

OWASP: A3 - Sensitive Data Exposure, A6 - Security Misconfiguration

CWE: CWE-297 - Improper Validation of Certificate with Host Mismatch

Description: TLS/SSL libraries provide built-in hostname verification functions that are not used. It is not recommended to re-invent the wheel. The system may be prone to man-in-the-middle attacks.

Solution: use TLS/SSL libraries methods.

```
sslContext.init(null, trustAllCerts, new java.security.SecureRandom());
HttpsURLConnection.setDefaultSSLSocketFactory(sslContext.getSocketFactory());
HostnameVerifier allHostsValid = (hostname, session) -> true;
```

Enable server hostname verification on this SSL/TLS connection. [Why is this an issue?](#)

6 months ago ▾ L115 🔗

🔒 Vulnerability ▾ 🚨 Critical ▾ 🔵 Open ▾ Not assigned ▾ 5min effort [Comment](#)

🏷 No tags ▾

4) *4th vulnerability*: Handle the following exception that could be thrown by `getWriter`: `IOException`.

File: utilities/.../descartes/teastore/kieker/rabbitmq/DisplayLogs.java, line 30
and

utilities/.../descartes/teastore/kieker/rabbitmq/IndexServlet.java, line 28

Severity: Minor

OWASP: A3 - Sensitive Data Exposure

CWE: CWE-600 - Uncaught Exception in Servlet

Description: not catching exceptions is a bad idea. The system typically sends debugging information back to the user. Such behavior could be prone to denial-of-service attacks by exposing potentially sensitive information.

Solution: catch and process exceptions.

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException {
    PrintWriter writer = response.getWriter();
```

Handle the following exception that could be thrown by "getWriter": `IOException`. [Why is this an issue?](#)

3 years ago ▾ L28 🔗

🔒 Vulnerability ▾ 🟡 Minor ▾ 🔵 Open ▾ Not assigned ▾ 20min effort [Comment](#)

🏷 No tags ▾

C. Security hotspots

As stated before, security hotspot results do not include standardized security categories. We use SonarSource categories for the following security hotspots:

1) *1st security hotspot*: Make sure the regex used here, which is vulnerable to super-linear runtime due to backtracking, cannot lead to denial of service.

File: services/.../src/main/webapp/bootstrap/js/bootstrap.js, line 109

Category: Denial of Service

Priority: Medium

Description: regular expression engines may cause a catastrophic backtracking situation. In the worst case, the complexity of the regular expression is exponential to the input size. This could lead to denial-of-service attacks.

Solution: do not expose input to a user unless necessary. Restrict the input size to a small number of characters. Add timeout to limit the regex evaluation time.

```
104     var $this    = $(this)
105     var selector = $this.attr('data-target')
106
107     if (!selector) {
108         selector = $this.attr('href')
109         selector = selector && selector.replace(/.*(?=#[^\s]*$)/, '') // strip for ie7
110     }
111
112     var $parent = $(selector === '#' ? [] : selector)
113
114     if (e) e.preventDefault()
```

2) *2nd security hotspot*: Make sure that using this pseudo-random number generator is safe here.

File: services/.../src/main/webapp/bootstrap/js/bootstrap.js, line 1681

Category: Weak Cryptography

Priority: Medium

Description: when software generates predictable values in a context requiring unpredictability, it may be possible for an attacker to guess the next value that will be generated.

Solution: `Math.random()` relies on a weak pseudo-random number generator. It is recommended to use a cryptographically strong pseudo-random number generator like `crypto.getRandomValues()`.

```
1676
1677     return title
1678 }
1679
1680 Tooltip.prototype.getUID = function (prefix) {
1681     do prefix += ~~(Math.random() * 1000000)
1682     while (document.getElementById(prefix))
1683         return prefix
1684 }
1685
1686 Tooltip.prototype.tip = function () {
```

3) *3rd security hotspot*: Make sure this debug feature is deactivated before delivering the code in production.

File: services/.../tools/descartes/teastore/auth/security/ShaSecurityProvider.java, line 74

Category: Insecure Configuration

Priority: Low

Description: sending debugging information back to the user could be prone to denial-of-service attacks by exposing potentially sensitive information.

Solution: do not enable debug features on production builds.

```

69     for (int i = 0; i < bytes.length; i++) {
70         sb.append(Integer.toString((bytes[i] & 0xff) + 0x100, 16).substring(1));
71     }
72     generatedPassword = sb.toString();
73 } catch (NoSuchAlgorithmException | UnsupportedEncodingException e) {
74     e.printStackTrace();
75 }
76 return generatedPassword;
77 }
78 }
79

```

4) 4th security hotspot: Make sure not using `rel="noopener"` is safe here.

File: `services/.../src/main/webapp/WEB-INF/pages/about.jsp`, line 12

Category: Others

Priority: Low

Description: a newly opened window having access back to the originating window could allow basic phishing attacks. The application opens untrusted external URL.

Solution: use `noopener` to prevent untrusted pages from abusing `window.opener`.

```

7
8     <div class="col-sm-4 col-md-4">
9         
11         <blockquote>
12             <p><a class="name" target="_blank" href="https://se.informatik.uni-wuerzburg.de/staff/joakim.kistowski/">Jóakim v. Kistowski</a></p>
13             <small><cite title="Source Title">Team leader, Interfaces, Persistence Provider Service</cite></small>
14         </blockquote>
15     </div>
16     <div class="col-sm-4 col-md-4">
17         

```

II. PENETRATION TESTING

A. Manual

TeaStore microservices can be deployed as Docker containers automatically via `docker-compose`. Follow instructions in the repository. ZAP is a GUI application with an installer. The installation is straightforward. Upon launching, the application asks for a port to operate on. As 8080 is reserved for TeaStore, we set the port to 8081. Then, one selects Automated Scan from the Quick Start tab and pastes the URL. AJAX spider does not work correctly; an additional setup is needed. We used a traditional spider to crawl the website. After the automated scan finished, we ran Manual Explore, from which we logged in to the TeaStore and explored the rest of the website.

B. Alerts

Throughout the penetration testing, we discovered 12 potential vulnerabilities. We chose and described eight of them.

- ▼ Alerts (12)
 - > Cross Site Scripting (Reflected) (4)
 - > Buffer Overflow (5)
 - > Format String Error
 - > Vulnerable JS Library (2)
 - > X-Frame-Options Header Not Set (216)
 - > Absence of Anti-CSRF Tokens (1266)
 - > Cookie No HttpOnly Flag (11)
 - > Cookie without SameSite Attribute (11)
 - > Timestamp Disclosure - Unix (6)
 - > X-Content-Type-Options Header Missing (224)
 - > Information Disclosure - Suspicious Comments (2)
 - > Loosely Scoped Cookie (234)

1) 1st alert: Cross Site Scripting (Reflected)

Priority: High

OWASP: A03 - Injection, A7 - Cross-Site Scripting (XSS)

CWE: CWE-79 - Improper Neutralization of Input During Web Page Generation (Cross-site Scripting)

URL: <http://cwe.mitre.org/data/definitions/79.html>

Description: an attack technique that involves echoing attacker-supplied code into a user's browser instance.

Solution: use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

2) 2nd alert: Buffer Overflow

Priority: Medium

OWASP: A03 - Injection, A1 - Injection

CWE: CWE-120 - Buffer Copy without Checking Size of Input (Classic Buffer Overflow)

URL: <https://cwe.mitre.org/data/definitions/120.html>

Description: buffer overflow errors are characterized by the overwriting of memory spaces of the background web process, which should have never been modified intentionally or unintentionally.

Solution: rewrite the background program using proper return length checking.

3) 3rd alert: Format String Error

Priority: Medium

OWASP: A03 - Injection, A1 - Injection

CWE: CWE-134 - Use of Externally-Controlled Format String

URL: <https://cwe.mitre.org/data/definitions/134.html>

Description: occurs when the submitted data of an input string is evaluated as a command by the application.

Solution: rewrite the background program using proper deletion of bad character strings.

4) 4th alert: Vulnerable JS Library

Priority: Medium

OWASP: A9 - Using Components with Known Vulnerabilities, A06 - Vulnerable and Outdated Components

CWE: CWE-829 - Inclusion of Functionality from Untrusted Control Sphere

URL: <https://cwe.mitre.org/data/definitions/829.html>

Description: library is vulnerable.

Solution: upgrade to the latest version of the library.

5) 5th alert: X-Frame-Options Header Not Set

Priority: Medium

OWASP: A05 - Security Misconfiguration, A6 - Security Misconfiguration

CWE: CWE-1021 - Improper Restriction of Rendered UI Layers or Frames

URL: <https://cwe.mitre.org/data/definitions/1021.html>

Description: X-Frame-Options header is not included in the HTTP response to protect against ClickJacking attacks.

Solution: most modern Web browsers support the X-Frame-Options HTTP header. Ensure it is set on all web pages returned by your site.

6) 6th alert: Absence of Anti-CSRF Tokens

Priority: Low

OWASP: A01 - Broken Access Control, A5 - Broken Access Control

CWE: CWE-352 - Cross-Site Request Forgery (CSRF)

URL: <https://cwe.mitre.org/data/definitions/352.html>

Description: a cross-site request forgery is an attack that involves forcing a victim to send an HTTP request to a target destination without their knowledge or intent in order to perform an action as the victim.

Solution: use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

7) *7th alert*: Cookie No HttpOnly Flag

Priority: Low

OWASP: A05 - Security Misconfiguration, A6 - Security Misconfiguration

CWE: CWE-1004 - Sensitive Cookie Without HttpOnly Flag

URL: <https://cwe.mitre.org/data/definitions/1004.html>

Description: a cookie has been set without the HttpOnly flag, which means that the cookie can be accessed by JavaScript.

Solution: ensure that the HttpOnly flag is set for all cookies.

8) *8th alert*: Timestamp Disclosure - Unix

Priority: Low

OWASP: A01 - Broken Access Control, A3 - Sensitive Data Exposure

CWE: CWE-200 - Exposure of Sensitive Information to an Unauthorized Actor

URL: <https://cwe.mitre.org/data/definitions/200.html>

Description: a timestamp was disclosed by the application/web server.

Solution: manually confirm that the timestamp data is not sensitive, and that the data cannot be aggregated to disclose exploitable patterns.

III. CONSLUSION

This document successfully obtained results from both static analysis and penetration testing. We can see multiple security concerns regarding the latest version of TeaStore, some of them of the highest priority. We believe more security issues would arise if we correctly set up ZAP with AJAX spider. We are satisfied with the CWE list of common security issues; the website was comfortable to work with.

REFERENCES

- [1] <https://github.com/DcartesResearch/TeaStore>
- [2] <https://sonarcloud.io/>
- [3] <https://www.zaproxy.org/>
- [4] <https://maven.apache.org/index.html>