GLOBALRAIN

Artemis Financial Vulnerability Assessment Report

Table of Contents

Document Revision History

Version	Date	Author	Comments
1.0	13 Jul., 2023	Alexander Ahmann	

Client

Developer: Alexander Ahmann

1. Interpreting Client Needs

The organisation that we are working for is *Artemis Financial*, which is a for-profit financial analysis company. Specifically, they specialise in savings, retirements, investments, and insurance. They are also implementing a REST API for what I presume is to be used by financial analysts who are programming financial models and using *Artemis Financial* databases as a data source.

Because of the nature of the work done by *Artemis Financial*, information security is of the utmost importance. Banking is an attractive target to cybercriminals due to the many opportunities for financial gain. Some notable examples of cybercriminals attacking the finance sector include:

- The case of Max Butler (a/k/a Max Vision or Iceman) stealing nearly two billion credit card numbers through an elaborate cybercriminal operation.[1]
- The case of the *Earth Longzhi* using customised versions of the *Colbalt Strike*[2] red teaming toolkit to target banks and other financial institutions in Ukraine and Asian countries.[3]
- The case of Paige Thompson hacking into the *Capital One* bank's computer systems by exploiting a misconfigured AWS (*Amazon Web Services*) instance and leaking data onto GitHub.[4] What is interesting is that Thompson did not have a profit motive for hacking into Capital One.[5]

These are just a few of what are many cases of cybercriminals attacking organisations that work with finance, where the cybercriminals in question have varying motivations. Furthermore, there are baseline security standards that financial firms should abide by in order to be legitimate. They include:

- The *Payment Card Industry Data Security Standard*: https://www.pcisecuritystandards.org/
- For international transactions with some European nation-states, the European Union's *General Data Protection Regulations*: https://gdpr.eu/what-is-gdpr/
    - A related cybersecurity standard developed by the United Kingdom's *Information Commissioner's Office* includes their own version of the GDPR: https://ico.org.uk/for-organisations/uk-gdpr-guidance-and-resources/
    - It should be noted that the United Kingdom was previously associated with the European Union until 2016, and that their version of the GDPR may differ significantly from the one issued by the European Union.
- The *Sarbanes-Oxley act* puts forward a government-level standard for the financial industry regarding cybersecurity: https://www.law.cornell.edu/wex/sarbanes-oxley_act

---

[1] As documented by Kevin Poulsen (2011). *Kingpin: How One Hacker Took Over the Billion-Dollar Cybercrime Underground*. Crown Forum.

[2] Cobalt Strike: Adversary Simulation and Red Team Operations: https://www.cobaltstrike.com/

[3] As reported by Bill Toulas (2022). *New hacking group uses custom 'Symatic' Cobalt Strike loaders*. Bleeping Computer. Retrieved on Jul. 14, 2023 from: https://www.bleepingcomputer.com/news/security/new-hacking-group-uses-custom-symatic-cobalt-strike-loaders/

[4] As documented by the United States Department of Justice (c.a. 2022). *United States v. Paige Thompson*. https://www.justice.gov/usao-wdwa/united-states-v-paige-thompson

[5] As documented by Steve King (c.a. 2020). *Inside the Mind of the Capital One Hacker*. Cyber Theory. Archived at: https://web.archive.org/web/20200815172306/https://cybertheory.io/inside-the-mind-of-the-capital-one-hacker/

With this in mind, I can proceed with the areas of security that programmers and the rest of the engineering team should concern themselves with.

2. Areas of Security

I am presented with the following scenario:

> "*Artemis Financial wants to modernize their operations. As a crucial part of the success of their custom software, they also want to use the most current and effective software security. Artemis Financial has a RESTful web application programming interface (API). They are seeking Global Rain's expertise about how to protect the organization from external threats.*
>
> *As part of the team, you must examine Artemis Financials' web-based software application to identify any security vulnerabilities. You'll document what you learn in a vulnerability assessment report that will be used for mitigating the security vulnerabilities that you find.*" --- CS-305 Project 1 Assignment (n.d.).

Based on this description and an initial assessment of the project code and dependencies, I have worked out the following to be of concern to the Artemis Financial software engineering team:

- **Input Validation**: the application will be using an API for developers who are interested in working with Artemis Financial data. Input Validation is needed to prevent malicious users from gaining access to confidential information that they are not authorised to have.
- **APIs**: the web application in question is an API. Naturally, software testing and vulnerability assessments involved in API development are needed to justify its security.
- **Code Error**: The API and web application needs to handle unexpected input and other exceptions, and not leak confidential information to error messages.
- **Code Quality**: This vulnerability assessment item concerns itself with the quality of the code, how well the code can be maintained, how well the code prevents end users from accidently disrupting operations or violating the integrity and confidentiality of the system, and its robustness to a "harsh" environment that it is executed in.
- **Encapsulation**: This application will be dealing with financial information and records. It is of the upmost importance that the data structures are properly engineered to protect the confidentiality and integrity of said data, and secure data structures are one of many ways to do that.

3. Manual Review

I have specifically used the *PMD* source code analyzer as an automation code review tool to aid myself in the manual inspection of bad coding practices regarding this project. PMD will find examples of bad coding style and I will use that as a basis for finding more bad coding practices.

In this section, I intend to discuss the flaws found in the .java source code files. I may also discuss some possible ways to improve the code. The table below will discuss the problems that PMD and myself have worked out point-by-point and correlate the coding problems to the raw source code displayed in view panes (in the appendix):

Source Code (.java) File	Programming Flaws Detected
`CRUD.java` (Pane 3.1)	• The classes and methods in the source code file have not been sufficiently documented. Comments are needed to identify the parameters and return of methods, and other documentation of the class and constructor. • In the constructor, a `final` should be placed before the `String` type. • Some of the type names are inappropriate. For example, the class should not be named `CRUD`, as it is considered too short in length. • Suggestion: perhaps the developers should introduce an input validation mechanism to handle end user input.
`CRUDController.java` (Pane 3.2)	• Like with CRUD.java, the classes and methods in the source code file have not been sufficiently documented. For each of the methods, comments are needed to briefly describe its behaviour, what parameters it takes, and what it will return. Comments should also prepend the classes and constructors in the source code file. • A constructor should be declared for the `CRUDController` class. • The local variable in the CRUD instance should be declared as a `final` variable. • Suggestion: perhaps an input validation mechanism should be introduced to the `CRUDController`, especially since it is taking input from a `GET` request.

`DocData.java` (Pane 3.3)	• Like with source code files discussed previously, the source code's components, such as its class, consturctor and methods, have not been properly documented with comments. • While comments are used in this source code file, it is leaking information about the system. For example, around line 32, it is revealed that the application is using the `root` user account and that the database name is `test`. • Regarding the exception handling, while this is generally good programming practice, it can lead to security issues if not properly implemented. The exception handling near lines 25-31 should be examined more thoroughly to make sure that it does not leak any confidential information to `stdout` or log files. • An inconsistent naming scheme was used when writing this code; `read_document` should be in camel case: `readDocument`.
`Greeting.java` (Pane 3.4)	• As with source code files examined previously, this source file is missing documentation about the classes, their respective constructors, and methods. • A `final` should be added before the data types `long` and `String` in the constructor. • Some variables use improper naming conventions; such as id. It is recommended that they are changed. • Suggestion: since this class takes information from what may be a hostile execution environment, the engineers might want to add input validation for good measure, If possible.
`GreetingController.java` (Pane 3.5)	• As with source code files examined previously, this source file is missing documentation about the classes, their respective constructors, and methods. • A constructor is needed for the `GreetingController` class. • In the declaration of a `Greeting` object, the `name` variable should be set to `final`. • Suggestion: perhaps the programmers and engineers can introduce an input validation mechanism, as data is

	being received from an HTTP request, which may contain malicious or other bad input.

4. Static Testing

I have used both the *Maven*[6] tool and the *PMD*[7] source code analyzer to automatically find programming errors and dependency vulnerabilities in the project. Maven has identified seventy-four (74) vulnerabilities in the *Artemis* web application and *PMD* has identified seventy-two (72) instances of improper coding style. In this section, I will just be discussing the output of specifically Maven.

The following tables show a more succinct version of Maven's output for each of the dependencies that it found to have documented n-day vulnerabilities. I focus on vulnerabilities that are of high impact to critical impact, but I do discuss some lower impact vulnerabilities:

bcprov-jdk15on-1.46.jar		
CVE Assignment	**Description**	**Notes**
CVE-2013-1624, CVE-2015-6644[8]	Bad implementation in cryptographic procedures where the ciphertext can be decrypted by a skilled attacker who has managed to intercept data as it was being transmitted from the client and server. A possible remedy is to update the dependency to the latest version.	-
CVE-2015-7940[9]	Bad implementation in cryptographic procedures. Specifically, it is susceptible to leaking private keys if an attacker sends specifically crafted elliptic curves to it. A possible remedy is to update the libraries to the latest version.	-

[6] See the Apache Maven Project (n.d.). Retrieved on Jul. 14, 2023 from: https://maven.apache.org/what-is-maven.html

[7] PMD (n.d.). An extensible cross-language static code analyzer. Retrieve on Jul. 14, 2023 from: https://pmd.github.io/

[8] National Vulnerability Database (n.d.). *Search Results*. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Abouncycastle&cpe_product=cpe%3A%2F%3Abouncycastle%3Alegion-of-the-bouncy-castle-java-crytography-api&cpe_version=cpe%3A%2F%3Abouncycastle%3Alegion-of-the-bouncy-castle-java-crytography-api%3A1.46

[9] National Vulnerability Database (n.d.). CVE-2015-7940 Vulnerability. Retrieved on Jul. 17, 2023 from: https://ossindex.sonatype.org/vulnerability/CVE-2015-7940?component-type=maven&component-name=org.bouncycastle%2Fbcprov-jdk15on&utm_source=dependency-check&utm_medium=integration&utm_content=5.3.0

CVE-2016-1000338, CVE-2016-1000339, CVE-2016-1000341, CVE-2016-1000342, CVE-2016-1000343, CVE-2016-1000344, CVE-2016-1000345, CVE-2016-1000346, CVE-2016-1000352[10]	Bad implementation of cryptographic libraries. These documented n-day vulnerabilities can lead to a violation of the confidentiality and integrity of the system. A possible remedy is to update the dependency to the latest version.	Individual CVEs may have additional fixes that go beyond simple software updates.

log4j-api-2.12.1.jar		
CVE Assignment	**Description**	**Notes**
CVE-2021-44228[11]	This is a critical impact vulnerability that allows for an attacker who can write to log files used by this dependency to execute their own code in the system. Possible remedies are to update the dependency to the latest version and disable JDNI features.	-
CVE-2021-45046[12]	This is a critical impact vulnerability that allows for an attacker who has access to the *Thread Context Map* to execute code on the system, locally or remotely. Possible remedies are to disable the JDNI component and to update the dependency to the latest version.	-
CVE-2021-44832[13]	This is a medium impact vulnerability that allows for an attacker who has access to the LDAP server to remotely execute code through the JNDI component.	-

[10] National Vulnerability Database (n.d.). *Search Results*. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/search/results?form_type=Advanced&results_type=overview&search_type=all&cpe_vendor=cpe%3A%2F%3Abouncycastle&cpe_product=cpe%3A%2F%3Abouncycastle%3Alegion-of-the-bouncy-castle-java-crytography-api&cpe_version=cpe%3A%2F%3Abouncycastle%3Alegion-of-the-bouncy-castle-java-crytography-api%3A1.46

[11] National Vulnerability Database (n.d.). *CVE-2021-44228 Details*. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2021-44228

[12] National Vulnerability Database (n.d.). *CVE-2021-45046 Details*. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2021-45046

[13] National Vulnerability Database (n.d.). *CVE-2021-44832 Details*. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2021-44832

	Possible remedies are to disable the JDNI component and to update the dependency to the latest version.	
CVE-2021-45105[14]	This is a medium impact vulnerability that allows for an attacker who has access to the Thread Context Map to instigate a denial-of-service attack by injecting a string crafted to do so. Possible remedies are to disable the JDNI component and to update the dependency to the latest version.	-

tomcat-embed-core-9.0.30.jar		
CVE Assignment	**Description**	**Notes**
CVE-2020-1938[15]	This is a critical impact vulnerability that allows an attacker who makes connections through the *Apache JServ* Service to remotely execute code on the system. Possible remedies for this is to block any connections to the *JServ* Service and to update the dependency to the latest version.	-
CVE-2022-29885[16]	This is a high impact vulnerability that allows for an attacker to instigate a denial-of-service against the system by attacking the *EncryptInterceptor* component of the package. A possible remedy is to update the dependency to the latest version.	-
CVE-2021-41079[17]	This is a high impact vulnerability that allows for an attacker to instigate a denial-of-service against the system by attacking TLS/OpenSSL components that do	-

[14] National Vulnerability Database (n.d.). CVE-2021-45105 Details. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2021-45105
[15] National Vulnerability Database (n.d.). *CVE-2020-1938 Details*. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2020-1938
[16] National Vulnerability Database (n.d.). *CVE-2022-29885 Details*. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2022-29885
[17] National Vulnerability Database (n.d.). *CVE-2021-41079 Details*. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2021-41079

	not properly validate encrypted packets through said components. A possible remedy is to update the dependency to the latest version.	
CVE-2021-25122[18]	This is a high impact vulnerability that allows for an attacker to view confidential information from another user or the server. A possible remedy is to update the dependency to the latest version.	-
CVE-2020-13935[19]	This is a high impact vulnerability that allows for an attacker to instigate a denial-of-service by sending specifically crafted payloads regarding the WebSocket components. A possible remedy is to update the dependency to the latest version.	-
CVE-2020-8022[20]	This is a high impact vulnerability that, I think, allows an attacker to escalate vulnerabilities. A possible remedy is to update the dependency to the latest version.	-
CVE-2020-11996[21]	This is a high impact vulnerability that allows an attacker to instigate a denial-of-service attack by exploiting the Tomcat service into consuming too many CPU resources, thus causing the server it is hosted on to become unresponsive. A possible remedy is to update the dependency to the latest version.	-

snakeyaml-1.25.jar

[18] National Vulnerability Database (n.d.). CVE-2021-25122 Details. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2021-25122
[19] National Vulnerability Database (n.d.). *CVE-2020-13935 Details*. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2020-13935
[20] National Vulnerability Database (n.d.). *CVE-2020-8022 Details*. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2020-8022
[21] National Vulnerability Database (n.d.). CVE-2020-11996 Details. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2020-11996

CVE Assignment	Description	Notes
CVE-2022-1471[22]	This is a critical impact vulnerability resulting from a bad implementation of SnakeYAML's Constructor class. The class does not restrict the object type when building it from raw data. This can allow an attacker to craft a special input that can lead to remote code execution. A possible remedy is to update the dependency to at least version 2.0, preferably to the latest version.	-
CVE-2022-25857[23]	This is a high vulnerability impact that allows for an attacker to instigate a denial-of-service. A possible remedy is to update the dependency to the latest version.	-
CVE-2017-18640[24]	This is a high impact vulnerability that allows for an attacker to exploit an entity expansion. A possible remedy is to update the dependency to the latest version.	-

spring-boot-2.2.4.RELEASE.jar		
CVE Assignment	**Description**	**Notes**
CVE-2023-20883[25]	This is a high impact vulnerability that allows for an attacker to instigate a denial-of-service if the Spring Framework's Model-View-Controller (MVC) is used in conjunction with a reverse proxy cache. A potential remedy is to update the dependency to the latest version.	-

[22] National Vulnerability Database (n.d.). CVE-2022-1471 Details. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2022-1471
[23] National Vulnerability Database (n.d.). CVE-2022-22857 Details. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2022-25857
[24] National Vulnerability Database (n.d.). CVE-2017-18640 Details. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2017-18640
[25] National Vulnerability Database (n.d.). *CVE-2023-20883 Details*. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2023-20883

CVE-2022-27772[26]	This is a high impact vulnerability that derives from unsupported components in the Spring Framework. A possible remedy is to update the dependency to the latest version.	I am not quite sure what the implications of this vulnerability are.

jackson-databind-2.10.2.jar		
CVE Assignment	**Description**	**Notes**
CVE-2021-46877[27]	This is a high impact vulnerability that allows attackers to instigate a denial-of-service on the system through exploiting the `JsonNode` serialization. A possible remedy is to update the dependency to the latest version.	-
CVE-2022-42004,[28] CVE-2022-42003[29]	These are high impact vulnerabilities that allow attackers to instigate a denial-of-service attack through the *FasterXML* component. A possible remedy is to update the dependency to the latest version.	-
CVE-2020-25649	This is a high impact vulnerability that allows for an attacker to violate the integrity of a system via an XML external entity attack. A possible remedy to this problem is to update the dependency to the latest version.	-

5. Mitigation Plan

I propose the following solutions to mitigate the problems listed above:

- Update dependencies to their most recent version.
- Disable any unnecessary features in each dependency.

[26] National Vulnerability Database (n.d.). *CVE-2022-27772 Details*. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2022-27772
[27] National Vulnerability Database (n.d.). *CVE-2021-46877 Details*. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2021-46877
[28] National Vulnerability Database (n.d.). *CVE-2022-42004 Details*. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2022-42004
[29] National Vulnerability Database (n.d.). *CVE-2022-42003 Details*. Retrieved on Jul. 17, 2023 from: https://nvd.nist.gov/vuln/detail/CVE-2022-42003

    - For programmers and software engineers: they should avoid adding too much unneeded functionality to their product. Less code generally implies less attack surface for a malicious hacker to work with.
- Properly document the codebase and use an object-oriented design when accessing or modifying data in the system.
    - For programmers and software engineers: they should use `get*` and `set*` functions when accessing and making changes to variables in another class.
- Add input validation to parsers and other methods of handling of external data to prevent injection-based attacks against the API and overall web application.
- Remove any unnecessary comments, especially comments that leak out information regarding how the computer system is set up.
- Regarding any exception handling of code, more testing should be done to make sure that confidential information is not leaked into `stdout` or log files when an exception is thrown.

6. Appendix: Source code (*.java) view panes

```java
package com.twk.restservice;

public class CRUD {
    private final String content;
    private final String content2;

    public CRUD(String content) {
        this.content = content;
        this.content2 = content;
    }

    public CRUD(String content1, String content2) {
        this.content = content1;
        this.content2 = content2;
    }

    public String getContent() {
        return content;
    }

    public String getContent2() {
        return content2;
    }
}
```

Pane 3.1: CRUD.java

```java
package com.twk.restservice;
```

```
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class CRUDController {

    @RequestMapping("/read")
    public CRUD CRUD(@RequestParam(value="business_name") String name) {
        DocData doc = new DocData();
        return new CRUD(doc.toString());
    }
}
```

CRUDController.java (Pane 3.2)

```
package com.twk.restservice;

import java.sql.*;

public class DocData {
    private String id;

    public DocData()
    {
    }



    public String getId()
    {
return id;
    }



    public void read_document(String key, String value)
    {
/* implement read method */
//Class.forName("com.mysql.jdbc.Driver");
try {
    Connection con=DriverManager.getConnection(
           "jdbc:mysql://localhost:3306/test","root","root");
} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
       }
//here test is database name, root is username and password
    }
```

```
}
```

DocData.java (Pane 3.3)

```
package com.twk.restservice;

public class Greeting {
private final long id;
private final String content;

public Greeting(long id, String content) {
this.id = id;
this.content = content;
}

public long getId() {
return id;
}

public String getContent() {
return content;
}
}
```

Greeting.java (Pane 3.4)

```
package com.twk.restservice;

import java.util.concurrent.atomic.AtomicLong;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class GreetingController {

private static final String template = "Hello, %s!";
private final AtomicLong counter = new AtomicLong();

@GetMapping("/greeting")
public Greeting greeting(@RequestParam(value = "name", defaultValue = "World") String
name) {
return new Greeting(counter.incrementAndGet(), String.format(template, name));
}
}
```

GreetingController.java (Pane 3.5)