BOOKHUB

SECURE SOFTWARE PLAN

Version *<2.0>*

<01/12/2022>

VERSION HISTORY

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Ivan Chuchulski	*<12/23/21>*	*<name>*	*<mm/dd/yy>*	Initial Design Definition draft
2.0	Ivan Chuchulski	*<01/12/22>*	*<name>*	*<mm/dd/yy>*	Finish Secure Design

TABLE OF CONTENTS

1 PROJECT SUMMARY

1.1 PROJECT GOALS

Целта на проекта е да предостави възможност на читателите за лесно намиране на книги, журнали и статии, удобно разглеждане на резюмета, както и за създаване и управление на собствени колекции от произведения.

Основните групи потребители са читателите и администраторите на системата. Резултата от проекта ще бъде да се създаде система, която позволява на читателите да намират разнообразни произведения и автори, да преглеждат информация за тях, както и да създават и управляват лична книжна колекция с категории за всяко произведение. За да използват системата, читателите трябва да се регистрират и успешно да се автентикират.

Администраторите могат да достъпват системата само след задължителна автентикация. При успешно влизане те ще имат възможност да виждат данни за натоварването на системата, да изготвят отчети за броя потребители, както и да преглеждат статистически данни за техните колекции.

1.2 PROJECT SCOPE

Обхватът на проекта е да се проектира, разработи и внедри сървърно и клиентско приложение. Сървърното приложение се грижи за обработването на заявки за търсене на библиотечно съдържание и сигурно съхранение на клиентските данни за вход и книжни колекции. Администраторите на системата ще може да получат информация за състоянието на системата, чрез приложение с графичен потребителски интерфейс, което работи на персонален компютър. Клиентската част може да бъде използвана от читателите, чрез приложение с графичен потребителски интерфейс, което работи на персонален компютър.

Проектът няма за цел да разработва система, която да управлява самото библиотечно съдържание, което читателите ще могат да преглеждат и да добавят в колекции. За осигуряване на множеството от книги и журнали ще бъде използвана външна система, която предоставя такава функционалност, като примери за такива системи за Goodreads, Google Books и Libib.

2 TESTING OBJECTIVES

Тестовите цели на системата са да провери и гарантира, че системата удовлетворява изискванията за безопасността и успешно устоява на заплахите, целящи да попречат на нормалното ѝ функциониране. Затова е нужно да се изготви примерен набор от данни, чрез който в процеса на тестването да се използват, за да се провери безопасността на системата.

В обхвата на тестването попадат софтуерните компоненти, които изграждат клиентската и сървърната част на системата, както и middleware частта, която осъществява комуникацията помежду им.

За клиентската и сървърната ще бъдат извършени автоматизирани тестове с инструменти за качеството на кода, както и ръчни тестове за използваемостта на графичния потребителски интерфейс.

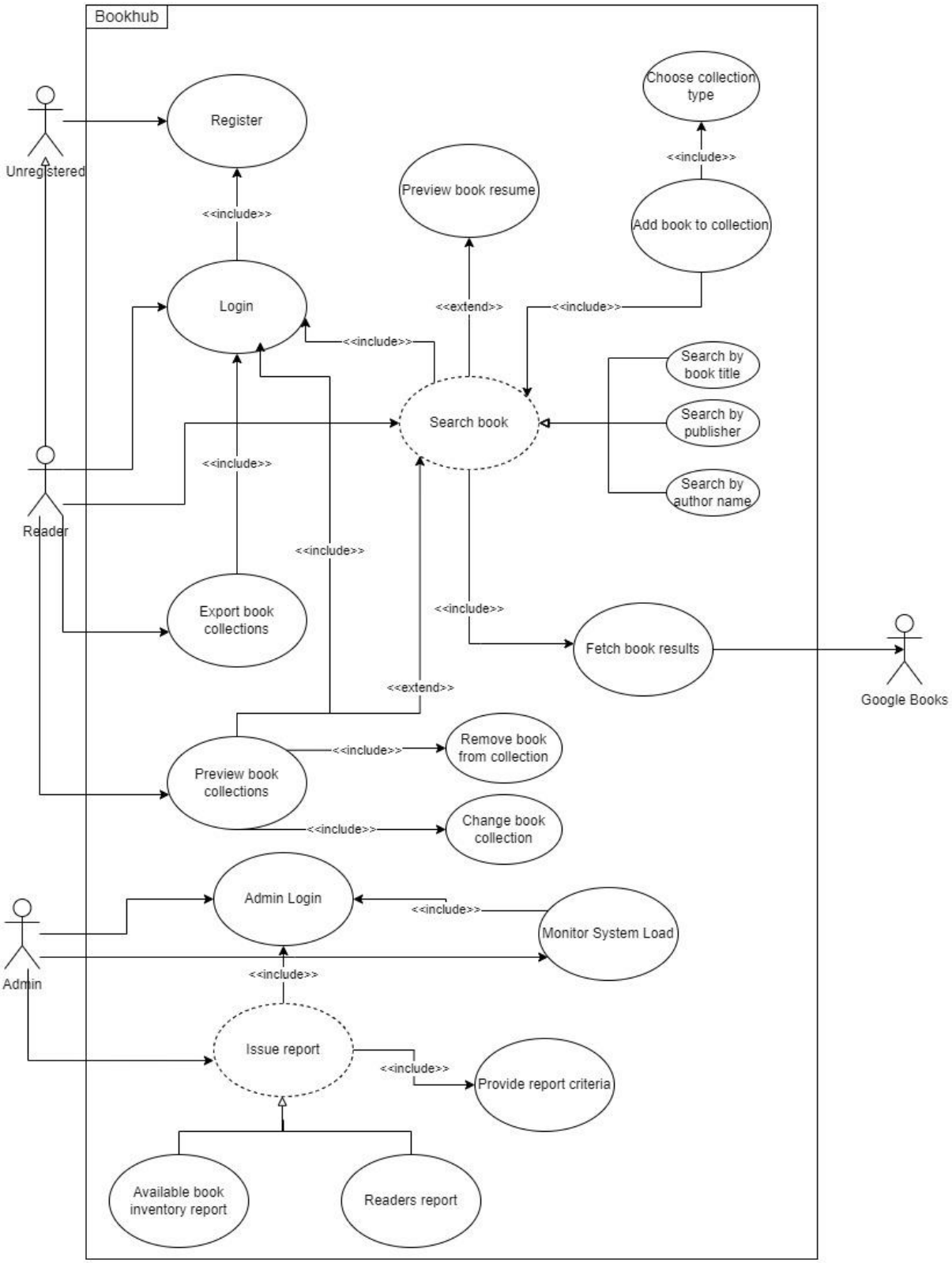
Тестването на сървърната част ще включва ръчни и автоматизиране тестове за проникване в системата, както и тестове за натовареност. Също така ще се извърши проверка на сигурността на софтуерната конфигурация за връзка с базата данни и комуникацията с клиентските части, за която се използва определен middleware.

Извън обхвата на тестването остава проверяване коректността на съдържанието на услугата за извличане на библиотечно съдържание, но е необходимо да се тества дали системата успешно осъществява сигурна връзка до съответната услуга.

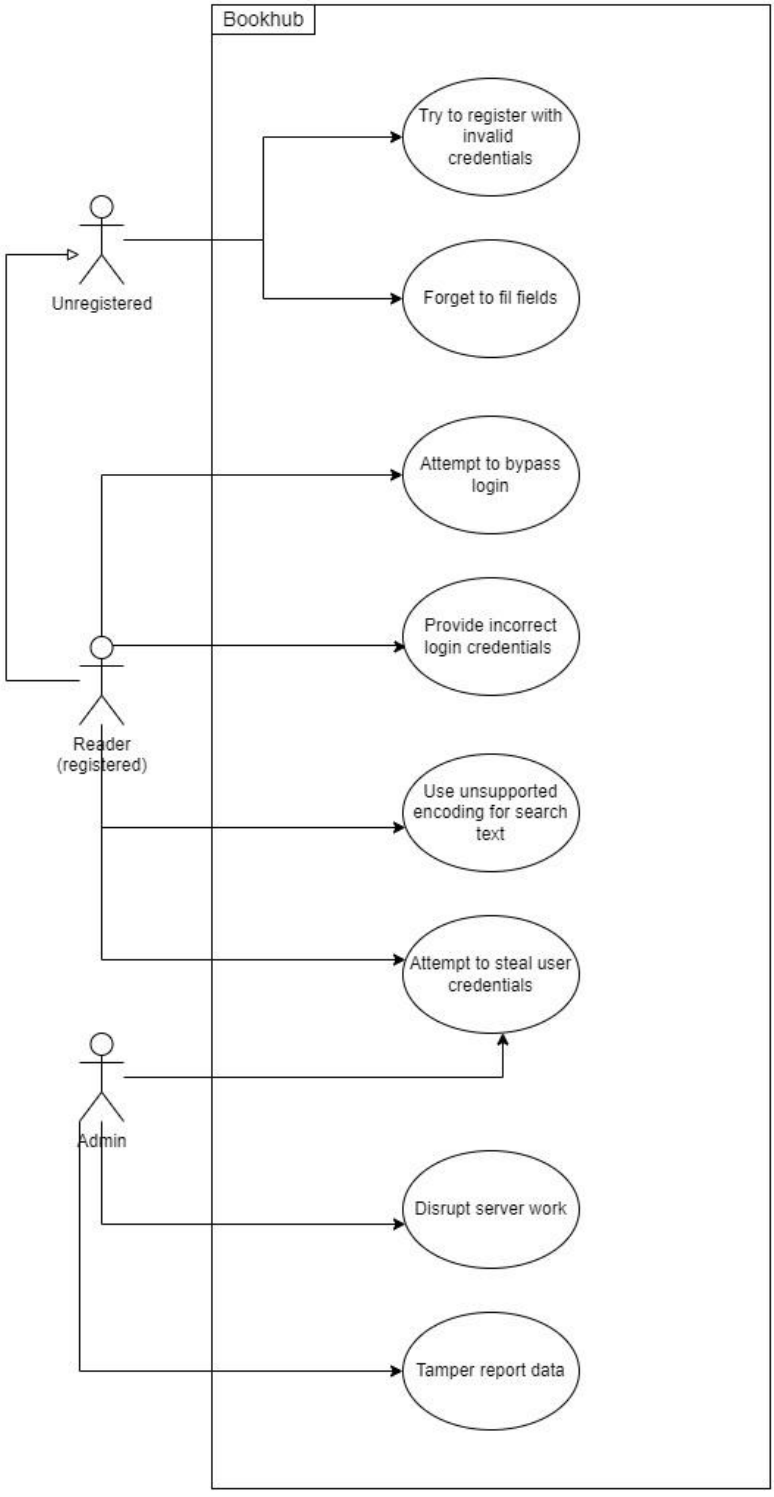
3 SECURE REQUIREMENTS

В тази секция ще изложим диаграмите, които описват потребителските случаи на употреба, както и abuse cases и misuse cases.

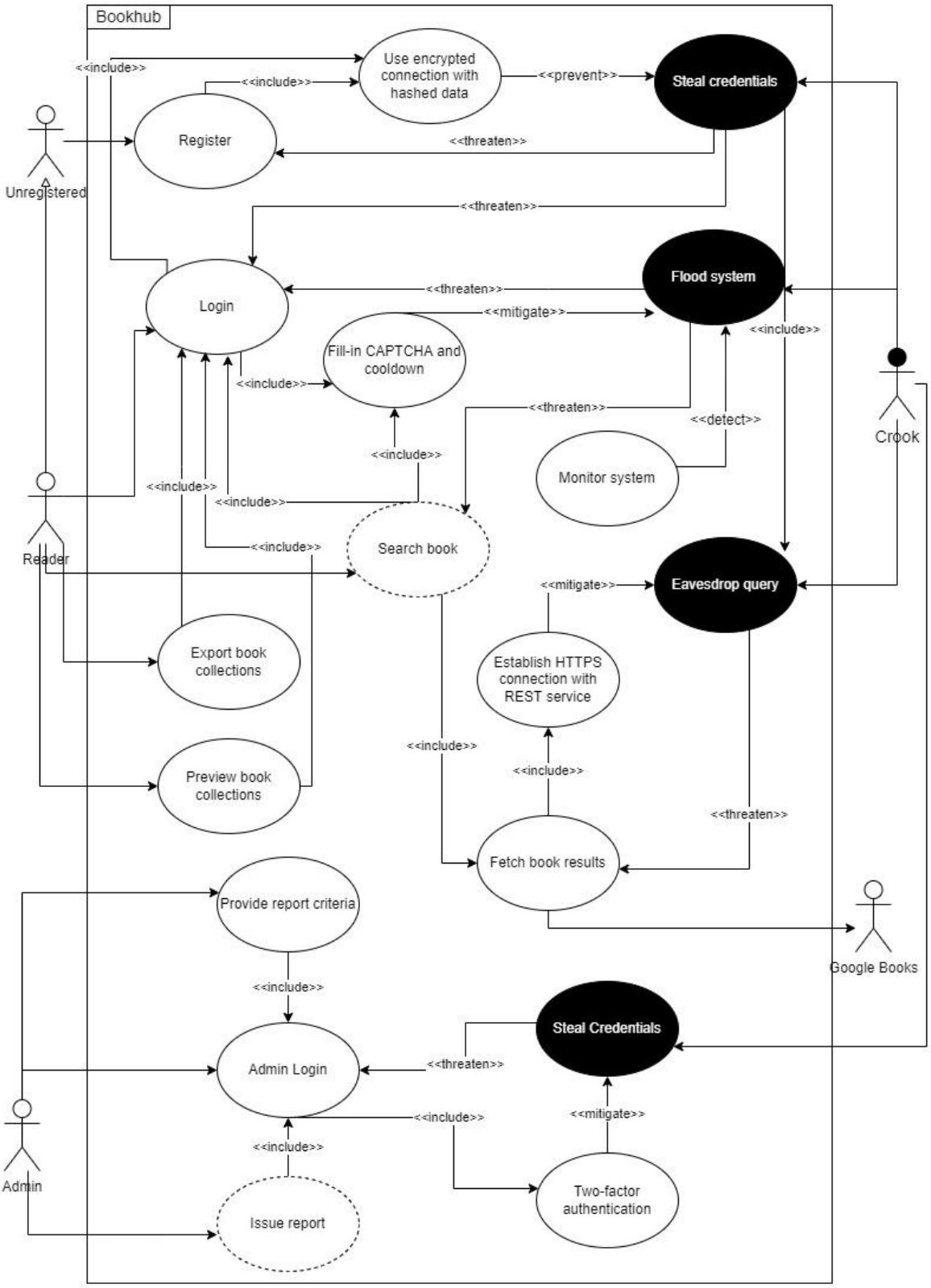
3.1 USE-CASES



3.2 ABUSE CASES



3.3 MISUSE CASES



4 THREAT ASSESSMENT

4.1 ASSETS

За да определим ценните активи в системата ще си отговорим на въпроса каква информация искаме да защитим от неоторизиран достъп. На първо място това са данните, които системата съхранява, като това са

- лични данни на потребителите на системата : това са потребителските имена и паролите, чрез които се осъществява автентикация на читателите и администраторите в системата
- съдържанието на читателските колекции
- докладите за състоянието на системата, които администраторите може да изискват
- наличността на базата данни и интегритета на съхраняваната в нея информация

Като активи можем да определим и всякаква информация, която се обменя между отделните подсистеми по време на работа. Това са :

- търсенията за произведения, които читателите отправят към сървъра
- заявките на сървъра към външни услуги, например към системата за библиотечно съдържание
- съдържанието на съобщенията, които се обменят между сървърната част и базата данни
- съдържанието на съобщенията, които осъществяват комуникацията между клиент и сървърната част

Към активите можем да отнесем и информацията в софтуерните конфигурации, които осигуряват работата на сървърната и клиентска част. Към тази част спадат :

- адрес и порт на сървърната част
- параметри за автентикация с база данни
- API ключове за достъп до външни услуги

4.2 THREATS MODELING

Целите за безопасността на системата са да се осигури опазване на личните данни на регистрираните читателите и администратори, гарантиране целостта на информацията в колекциите на читателите, осигуряване на достоверността на резултатите от търсенията и предотвратяване на възможност за подмяна съдържанието на системните доклади.

На база на така определените активи на системата, възможните заплахи към системата можем да разделим в две основни групи:

Първата група е заплахи, насочени към откradване, промяна или разрушаване на системните данни. Обект на тези атаки са личните данни на потребителите, информацията за книгите и авторите, както и съдържанието на читателските колекции. Примери за такива заплахи са подслушване на комуникациите,

Към втората група заплахи можем да включим, тези които целят да нарушат нормалното функциониране на системата и да предизвикат временен отказ, което нарушава качественото изискване наличност, или отговорите връщани от системата да са с невалидно или неправилно съдържание, т.е. нарушаване на качественото изискване надеждност. Такива заплахи може да се осъществят, чрез изпращане на огромен брой заявки за кратък период от време и/или отправяне на заявки с целенасочено невалидни или подменени данни.

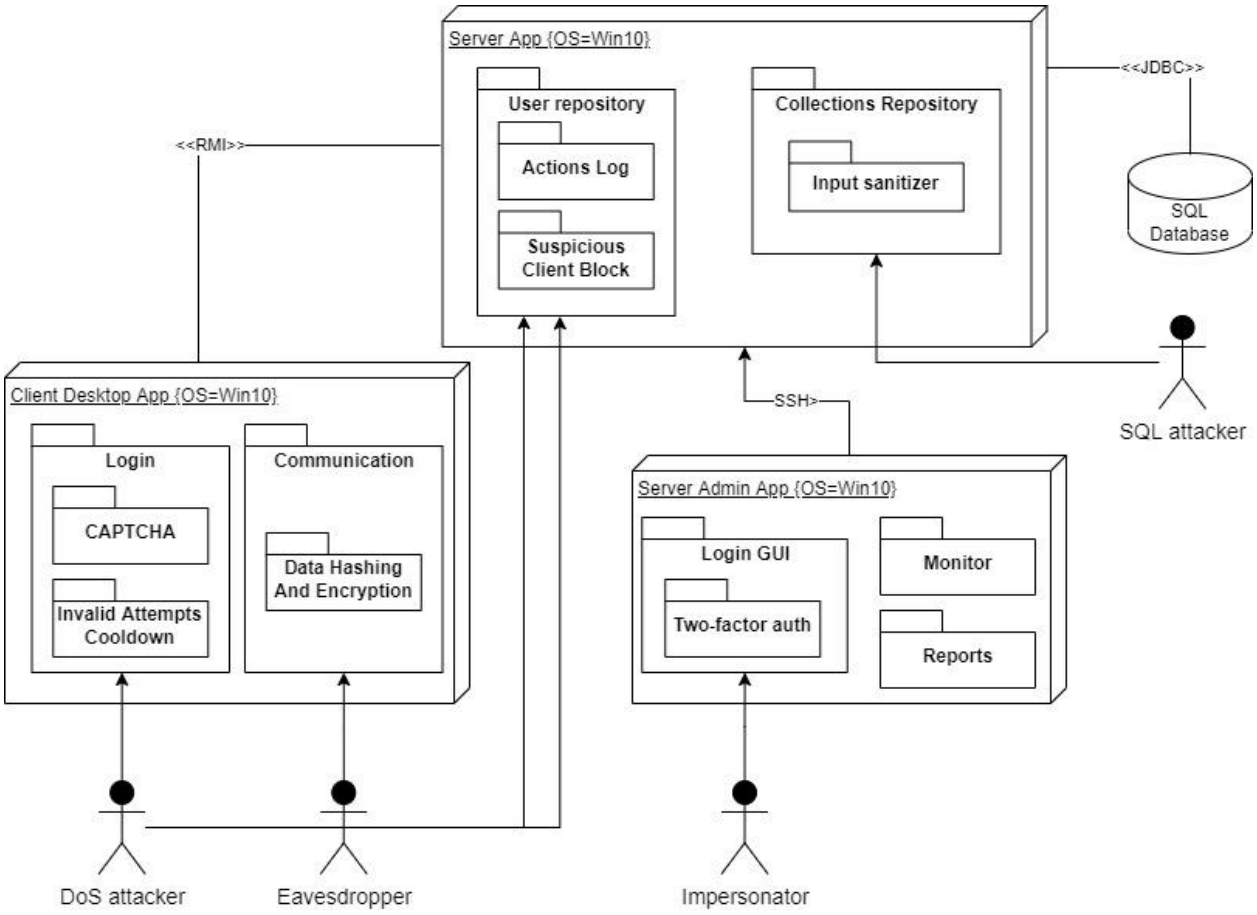
За оценяване на заплахите и начини за справяне с тях ще разгледаме следната таблица, като построяваме таблицата на база да ценните активи.

Asset(актив)	Threats(заплаха)	Vulnerabilities(уязвима част)	Likehood/ Impact Severity	Countermeasure(противодействие)
лични данни на потребителите на системата съдържанието на читателските колекции доклади за състоянието на системата	• неавторизиран достъп • откradване, подмяна или разрушаване на информацията • атаки с груба сила(brute force) • SQL Injection атаки	• полета за вход/търсене в клиенска част • полета за вход/търсене в сървърна част	Extreme / Serious	• Криптиране и хеширане на данните, които се обменят по мрежата • Осъществяване на комуникацията чрез криптирана връзка • Осъществяване на достъп до системата чрез двуфакторна автентикация • Временно суспендиране (cooldown) на потребител при множество неуспешни заявки • Валидиране на потребителските данни
търсенията за произведения заявки на сървъра към външни услуги комуникация между БД и сървъра комуникация между клиент и сървърната част	• подслушване • подмяна на данните • отказ от услуга(DoS)	• отворени портове на клиентската машина • отворени портове на сървърната машина	Very High / Critical	• Осъществяване на комуникацията чрез криптирана връзка • Ограничаване на въздействието на автоматизирани скриптове за атака чрез CAPTCHA • Ограничаване на действията на новорегистриран потребител за определено време • Използване на система за логове на потребителската дейност • Наличие на защитна стена при сървъра
адрес и порт на сървърната част	•подслушване •получаване на на отдалечен достъп	• конфигурационни файлове на сървърна и клиентска част	High / Serious	• Използване на динамично генерирани конфигурации за съхранение на порта • Избягване използване на стандартните портове за услугите
параметри за автентикация с база данни	•получаване на на отдалечен достъп до базата данни	• конфигурационни файлове на сървърна част	High / Critical	• Съхранение на данните за автентикация с БД в динамично генерирани конфигурации • Осъществяване на криптирана връзка с БД • Създаване на отделен потребител с ограничени права в базата данни
API ключове за достъп до външни услуги	• кражба и злоупотреба с ключа	• конфигурационни файлове на сървърна част	High / Serious	• Съхранение стойностите на API ключове в динамично генерирани конфигурации • Осъществяване на криптирана връзка с външна услуга

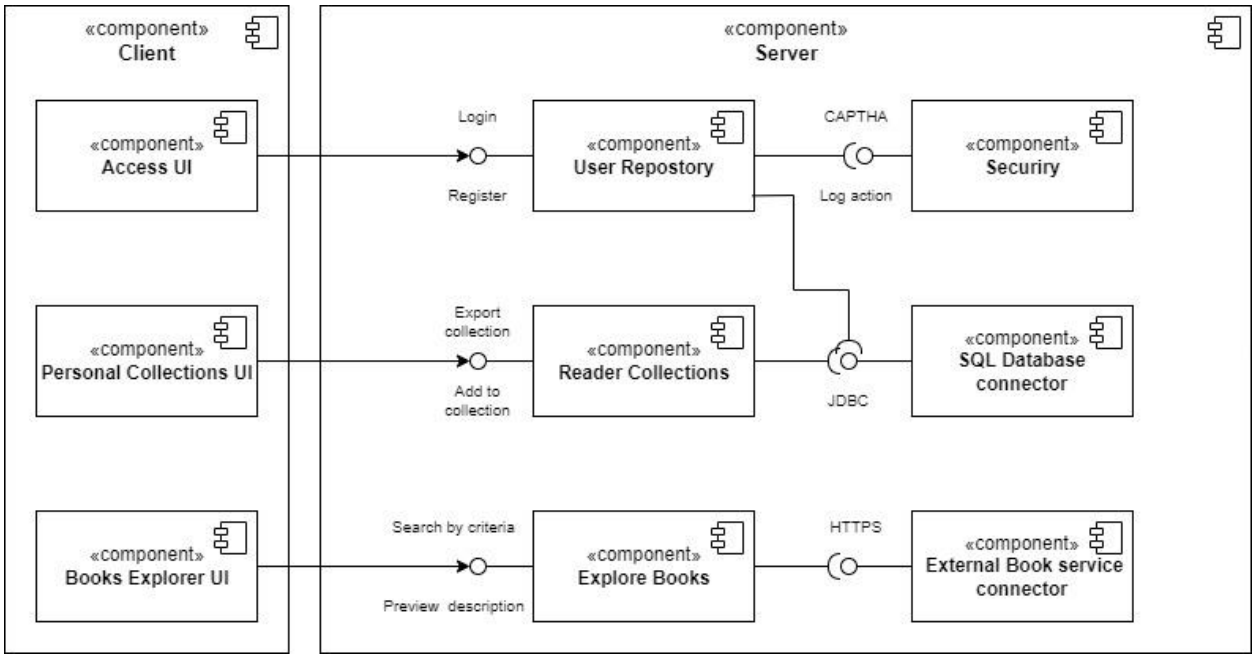
5 SECURE DESIGN

This section outlines the system and hardware architecture design of the system that is being built.

5.1 PHYSICAL SECURITY ARCHITECTURE



5.2 COMPONENT / SERVICE SECURITY ARCHITECTURE



6 SECURITY DEVELOPMENT AND TESTING

6.1 STATIC ANALYSIS

В графата за статично тестване ще проведем два основни вида тестване. Първо ще проведем инспекции със следните софтуерни инструменти : “Sonarlint” и “Checkstyle”, както и инспекциите по подразбиране от средата Intellij. След това ще се направи ревю на кода.

Със статичния анализ се стремим да открием често срещани грешки в кода, съмнителни части(т.нар. “code smells”), недостижими части на кода и да идентифицираме логически последователности, които може да предизвикат неопределено поведение. Такъв тип инструменти може да бъдат конфигурирани да следят и уведомяват за нови версии на библиотеки. Също така може да се използват за спазване на специфично оформление на кода, спрямо дефинирани правила.

Инструментите “Sonarlint” и “Checkstyle” предоставят възможността за сканиране на целия проект и изготвяне на доклад за откритите нередности, а инспекциите в “Intellij” се показват при разглеждане на кода и на място посочват проблема и предлагат варианти за поправка.

6.2 DYNAMIC TESTING

Динамичния тип тестване включва тестване на ниво програмна единица(unit test) и цялостно системно тестване, което ще се състои в ръчно пускане на системата и преглед на за коректната работа на нейните функционалности.

Целта е да се провери дали системата коректно реализира дефинираните в спецификацията изисквания за функционалност.

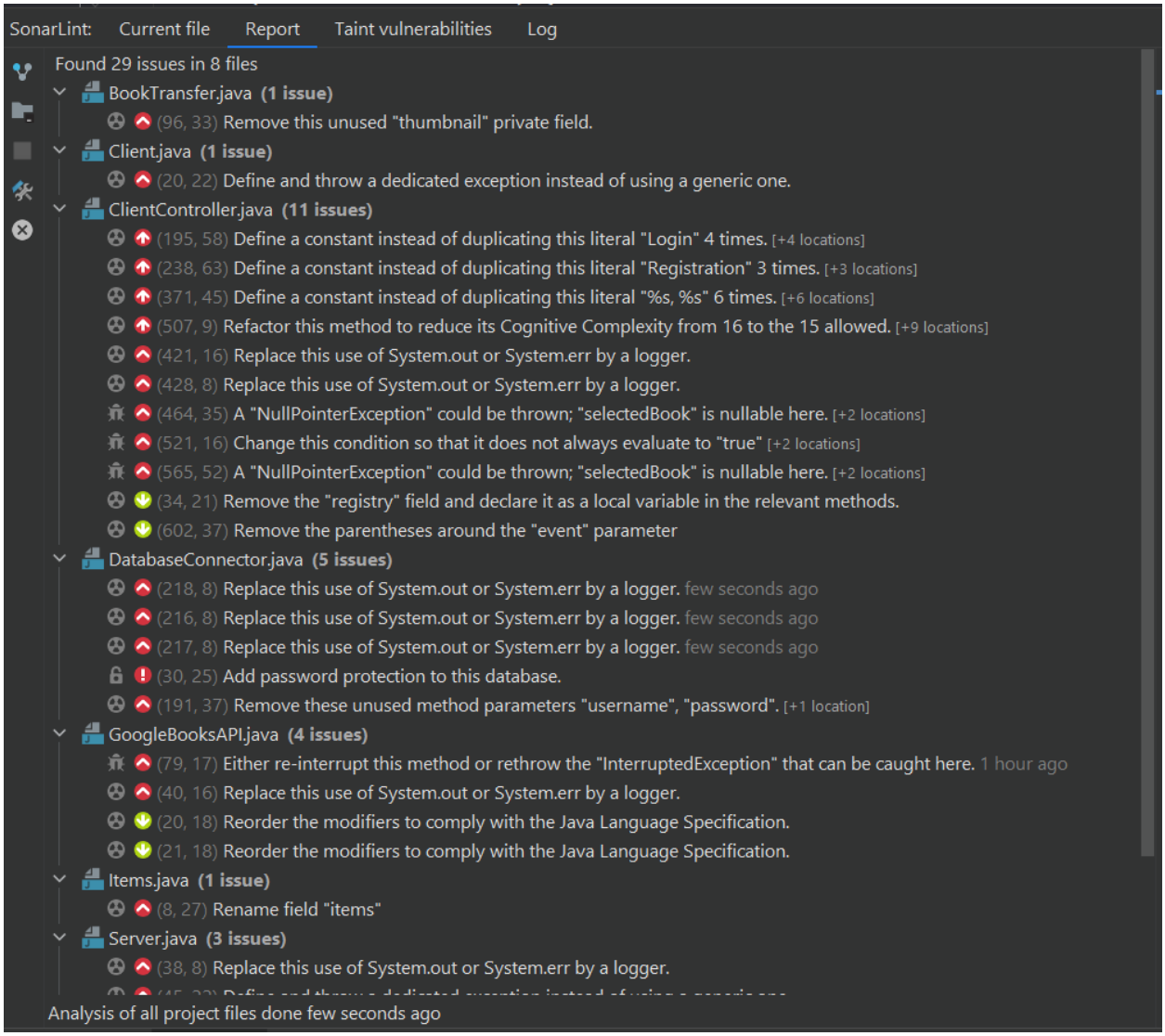
6.3 PENETRATION TESTING

Тестове за проникване на системата, които ще се проведат са опити за SQL инжекция, сканиране на сървърната машина за отворени портове и за уязвимости на RMI регистъра с инструмента “nmap”, както и ще опитаме да прихванем трафика между клиент и сървър с “Wireshark”.

7 TEST RESULTS

7.1 РЕЗУЛТАТИ ОТ СТАТИЧНИТЕ ТЕСТОВЕ

Резултатите от проведено сканиране със “Sonarlint” можем да видим в следния доклад, който се генерира от инструмента :



Фигура 1 Доклад на Sonarlint

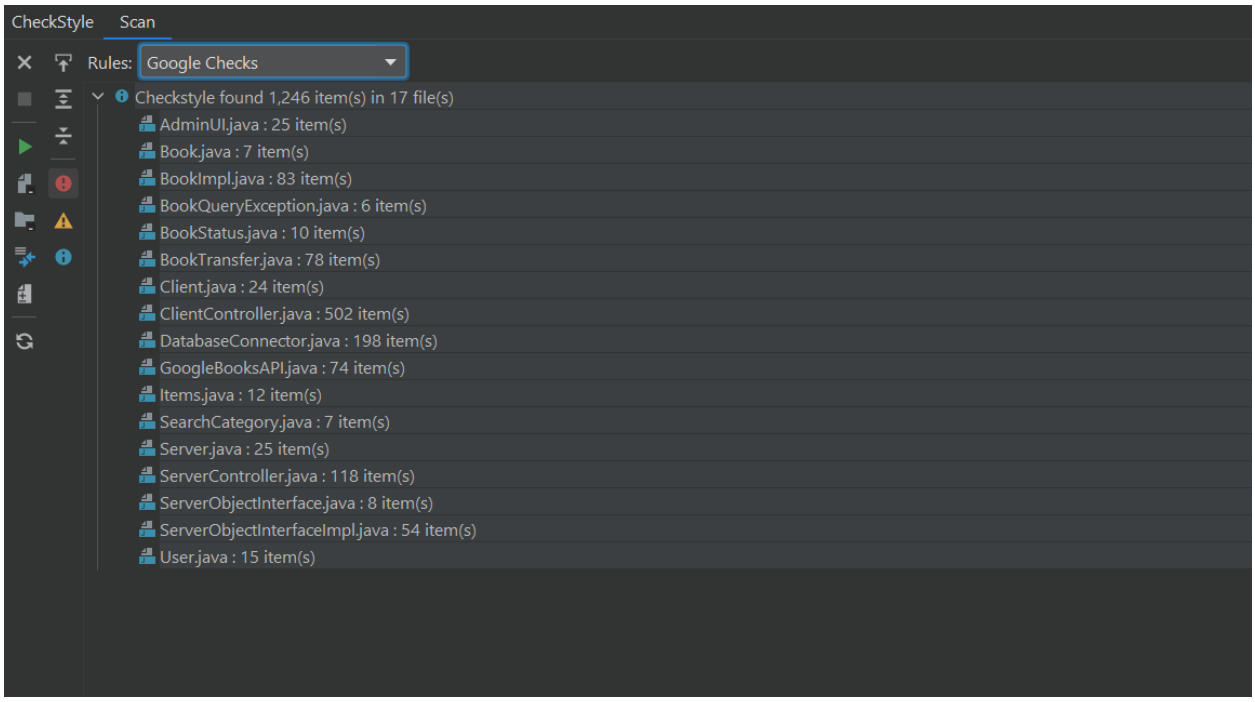
Виждаме някои безобидни предупреждения като това, че не е спазена спецификацията на Java за подреждане на модификаторите при декларации на полета, като “public”, “static” и други.

Друго предупреждение, които виждаме е, че съобщенията които се изписват на стандартния изход от “System.out” е по-добре да бъдат заменени с отделен logger компонент, за да се предотврати изтичане на чувствителна информация като съобщенията и stacktrace-овете на изключенията, които се случват.

Виждаме и предупреждения за възможно възникване на null-указател в частта, които се грижи за потребителския интерфейс, както и възможни проблеми със изключение, което сигнализира за прекъсване на нишка.

Това, което е изключително полезно при инструмента Sonarlint е, че за всеки проблем се предоставя кратко обяснение защо е възникнал и се предлага решение или насока как да се поправи.

При провеждане на сканиране с “Checkstyle” не виждаме грешки, а само предупреждения за некоректно форматиране на кода.



Фигура 2 Доклад на Checkstyle

Това е така, защото в проекта няма наложена конвенция за форматиране спрямо някои code style guide, а единствено е форматиран с помощта на средата за пренасяне на нов ред при преминаване на вертикален разделител.

Чрез автоматично сканиране от средата “Intellij” виждаме информация за по-нова версия на външните зависимости на проекта.

Фигура 3 Сканиране от Intellij

Средата дава възможност при обновяване на дадена библиотека да изберем конкретна версия.

Проведено беше ревю на кода, при което установихме факта, че паролите не се прехвърлят нито се съхраняват хеширани в базата данни. Това е проблем, тъй като при евентуално компрометиране достъпа до базата, недоброжелателите може да се сдобият с личната информация на потребителските акаунти.

7.2 РЕЗУЛТАТИ ОТ ДИНАМИЧНИТЕ ТЕСТОВЕ

В тази част обикновено трябва да проверим резултатите от unit тестовете и други автоматизирани системни тестове. За съжаление в проекта не присъстват никакви автоматизирани тестове, които да проверяват коректността на отделните програмни единици, а също така на системата като цяло.

Това което можем да направим е т.нар. ad-hoc системно тестване, което се състои в ръчно проверяване функционалностите на системата. За целта създадохме следните тестови акаунти :

Тестов акаунт	Потребителско име	Парола
Читателски 1	test	test
Читателски 2	test1	test1
Читателски 3	test2	test2
Администраторски 1	admin	admin

За всеки от читателските профили изпробвахме функционалностите на система за търсене и филтриране на библиотечно съдържание, добавянето на избрани произведение в различни колекции и разглеждане на личните колекции.

С администраторския акаунт беше бяха наблюдавани регистрираните акаунти и общия инвентар от произведения, който е наличен в системата.

Чрез тези тестове беше проверено, макар и без никакви точни замервания, че времето за отговор при заявка, при едновременна работа на повече от един потребител е в рамките на една до три секунди.

7.3 РЕЗУЛТАТИ ОТ ТЕСТОВЕТЕ ЗА ПРОНИКВАНЕ

Проведени бяха опити за SQL инжекция посредством полетата за вход със следните примерни данни :

Тестов случай	Стойност на username	Стойност на password	Резултат
SQL инжекция 1	' OR 1=1 --	testingpass	неуспешен
SQL инжекция 2	testingusername	' OR 2=2 --	неуспешен
SQL инжекция 3	testingusername	' OR '1'='1	неуспешен

Целта на теста е да прескочим автентикацията на системата, но при провеждането на теста не наблюдавахме такова нещо. Това се дължи на факта, че в кода са използвани “prepared statements”, които компилират текста на заявката във вид на процедурен код с параметри, които имат тип.

В текстовите полета за търсене на съдържание бяха опитани следите инжекции :

Поле	Стойност	Резултат
Търсене на произведения	potter; DROP preferences --	неуспешен
Търсене в лични колекции	potter; DROP preferences --	неуспешен

Атаките не причиняват вреда на системата, тъй като текста от тези полета не се използва за отправяне на заявки към базата данни.

Чрез инструмента “nmap“ проведохме сканиране за отворени портове на сървърната машина. Тъй като при провеждане на теста изпълняване сканирането на същата хардуерна машина, на която е разположен и сървъра, то можем лесно да видим отворените портове и информация за услугите на тях.

При тестовете с “nmap” също така бяха използвани следните скриптове от частта за откриване на уязвимости “vuln” :

- https://nmap.org/nsedoc/scripts/rmi-dumpregistry.html
- https://nmap.org/nsedoc/scripts/rmi-vuln-classloader.html

Тестовете обобщаваме в следната таблица :

Тестов случай	Команда	Резултат
сканиране на отворени портове	nmap –sS –p 80,443,3306,1098,1099,7777 <host>	успешен
сканиране на отворени портове с проверка за реалната услуга	nmap –sSV –p 80,443,3306,1098,1099,7777 <host>	успешен
преглед на съдържанието на RMI регистъра	nmap –script rmi-dumpregistry -p 7777 <host>	неуспешен

преглед на заредените класове в RMI регистъра	nmap --script=rmi-vuln-classloader -p 7777 <target>	неуспешен

Това което може да видим е, че сканирането за портове е успешно и дава информация за активни портовете на базата данни и порт 7777, на който всъщност сме стартирали RMI регистъра. Интересното в случая е, че на порт 7777 всъщност се докладва информация за услуга, която не е всъщност регистъра.

```
Host is up (0.00041s latency).

PORT     STATE  SERVICE
80/tcp   open   http
443/tcp  open   https
1098/tcp closed rmiactivation
1099/tcp closed rmiregistry
3306/tcp open   mysql
7777/tcp open   cbt

Nmap done: 1 IP address (1 host up) scanned in 0.21 seconds
```

Фигура 4 Резултат от сканиране на портовете

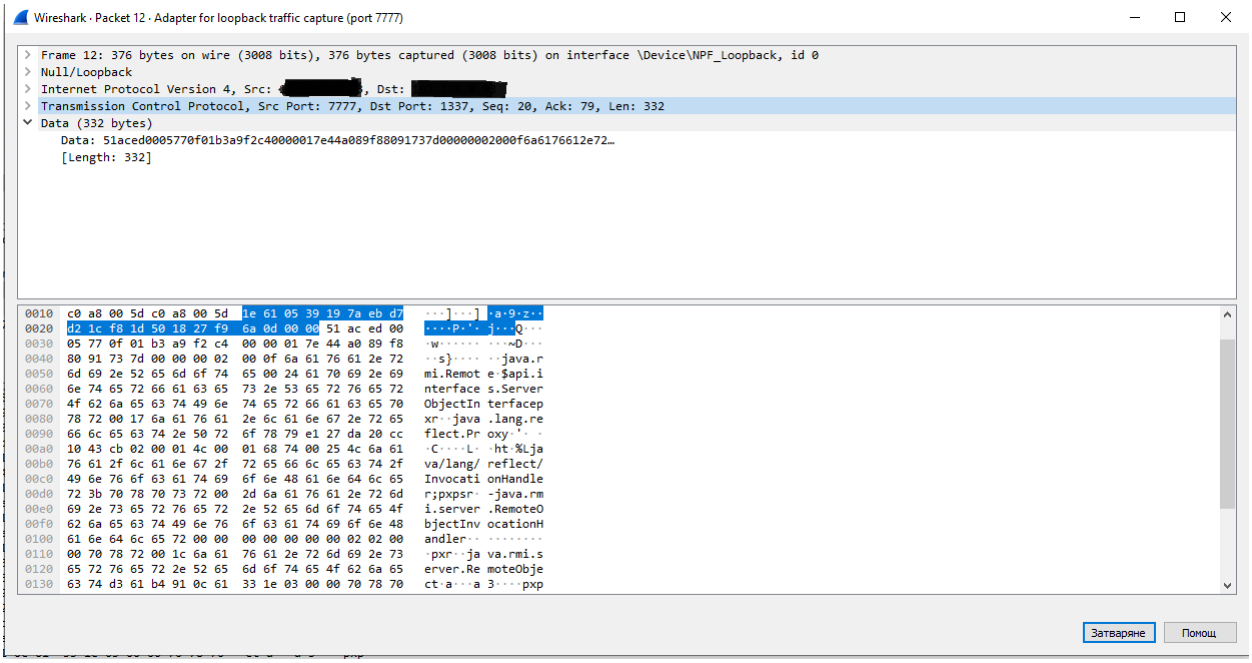
При извършване на повторно сканиране с опцията "-V", която опитва да определи реалната услуга, която е активна на порта успяваме да разберем, че на порт 7777 е стартиран регистъра.

```
Host is up (0.00067s latency).

PORT     STATE  SERVICE       VERSION
80/tcp   open   http          Apache httpd 2.4.41 ((Win64) OpenSSL/1.1.1c PHP/7.4.2)
443/tcp  open   ssl/http      Apache httpd 2.4.41 ((Win64) OpenSSL/1.1.1c PHP/7.4.2)
1098/tcp closed rmiactivation
1099/tcp closed rmiregistry
3306/tcp open   mysql?
7777/tcp open   java-rmi      Java RMI
1 service unrecognized despite returning data. If you know the service/version, please submit the following fingerprint
at https://nmap.org/cgi-bin/submit.cgi?new-service :
SF-Port3306-TCP:V=7.92%I=7%D=1/10%Time=61DC3C0E%P=i686-pc-windows-windows%
SF:r(NULL,4E,"J\0\0\x01\xffj\x04Host\x20'DESKTOP-4D5KDLT'\x20is\x20not\x20
SF:allowed\x20to\x20connect\x20to\x20this\x20MariaDB\x20server");

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.21 seconds
```

Фигура 5 Резултат от подобреното сканиране на портовете

При изпълнение на скриптовете не беше наблюдавана никаква допълнителна информация за съдържанието или classpath-а на RMI регистъра.

С помощта на инструмента Wireshark знаейки порта на който се намира RMI регистъра можем да опитаме да прихванем съобщенията към него.



Фигура 6 Прихванат отговор от регистъра

В едно от съобщенията виждаме прихванатa информация от регистъра обратно към клиента за името на отдалечения интерфейс и някои други служебни данни за протокола.

Ако опитаме да подслушаме порта на базата данни можем да се сдобием с информация за заявките и параметрите им.

Трябва да спомена факта, че заявките от сървъра към външната услуга за набавяне на библиотечно съдържание се осъществява по HTTPS и дори и да бъде прихванатa от атакуващ, то информацията в заявките е криптирана и следователно защитена.

За да се справим с тези проблеми е необходимо при регистриране на обекти в RMI регистъра да указваме начина на установяване на връзка с клиенти да става през SSL протокола. Същия принцип може да бъде приложен и при свързване на сървъра с базата данни, т.е. използвани на конектори, които предоставят възможност за установяване на връзка чрез SSL.

Друго решение, касаещо внедряването на системата е да бъде използвана защитна стена(firewall) пред сървъра, която би прихванала опитите за сканиране на портовете и в последствие може да предотврати опитите за подслушване на трафика.

8 TOOLSET

*[See **Appendix C**.]*

9 APPENDIX A: REFERENCES

[0] „хранилище на проекта”, линк: https://github.com/ivanchuchulski

[1] “OWASP testing guide”, линк:

https://wiki.owasp.org/index.php/OWASP_Testing_Guide_v4_Table_of_Contents

[2] Alex Kalinovsky, “Covert Java. Techniques for Decompiling, Patching and Reverse Engineering”, “Sams Publishing”, May 2004, ISBN: 0-672-32638-8

[3] инструмент “Sonarlint”, линк: https://www.sonarlint.org/intellij

[4] скриптове за nmap,

линк: https://nmap.org/nsedoc/scripts/rmi-dumpregistry.html,

линк: https://nmap.org/nsedoc/scripts/rmi-vuln-classloader.html

[5] “MaribDB SSL Java connector”,

линк: https://mariadb.com/kb/en/using-tls-ssl-with-mariadb-java-connector/

[6] “JVM SSL sockets”, линк:
https://docs.oracle.com/javase/8/docs/technotes/guides/rmi/socketfactory/SSLInfo.html

10 APPENDIX C. TOOLSET

Tool	Open source
Source-code analyzers	Sonarlint, Checkstyle, Intellij
Port scanner	nmap
Packet sniffer	Wireshark
Diagram generator	draw.io