An Analysis of Risk in Open-Source Project Dependencies

Róisín Ní Bhriain - 23269640
Sneha Dechamma Mallengada Suresh - 23262168
Course: MCM Computing
Supervisor: Darragh O’Brien

Introduction

- Popularity of Open Source Software has shown a steady increase over time
  - 3.6 million repos depend on the top 50 open-source projects
- The use comes with the risk of software vulnerabilities
  - Attackers can exploit these
- Prediction of risk is important for this
  - Software metrics
  - Project Activity
  - Vulnerability Data

Research Questions

Question 1:

Are there feature combinations that can be made from risk prediction methods that could provide developers with a more effective risk measure that allows them to minimise risk when choosing between multiple candidate open-source components?

Question 2:

Can a visual dependency tree be created for a project consisting of colour-coded nodes based on the predicted risks?

Background Research

Vulnerability Propagation

- Only 1.2% directly use vulnerable code
- Small packages can affect many packages in the Maven ecosystem
    - CVEs can affect a large number of projects
- The more dependencies the more complex it is to find vulnerabilities in dependencies

Project Metadata Analysis

- Project activity level is an indicator of survival
  - Number of commits, time-to-fix
  - Long-time Contributors
- Less than 50% of abandoned projects find new contributors
- Standard measurement for commits is:
  - Number of commits per month

Vulnerability CVE Data Analysis

- ➔ Predicting number of vulnerabilities per month is also important
- ➔ ARIMA is a very popular method of prediction
    - ◆ Seasonality not a factor
- ➔ Datasets from the NVD were used in every study we analysed
    - ◆ Prediction of CVEs
- ➔ Most models fall flat at the three month mark

Case Studies

Log4j Vulnerability

- ➔ Discovered in November 2021 in popular logging library
- ➔ Gateway to gain control of the machine
- ➔ Many projects unaware they were affected

Heartbleed Vulnerability

- ➔ Discovered in April 2014 in popular cryptography library
- ➔ Receive private information after crafting similar messages
- ➔ Many servers unaware they were affected

Methodology

Data & Dependencies Gathering

- Different GitHub projects to test the algorithm
- GitHub API for project activity
- NVD API for vulnerability Data

- Each Maven Dependency as a node
- NetworkX graph
- Keywords and libraries extracted for prediction
- Colour-coded

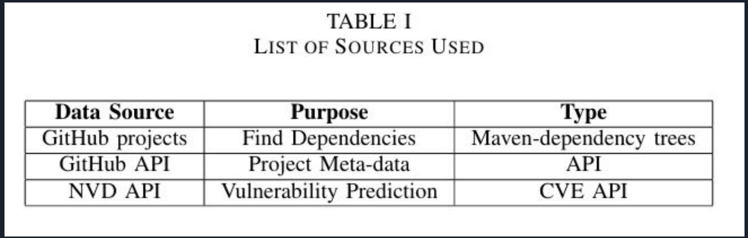


TABLE I
LIST OF SOURCES USED

Data Source	Purpose	Type
GitHub projects	Find Dependencies	Maven-dependency trees
GitHub API	Project Meta-data	API
NVD API	Vulnerability Prediction	CVE API

Predictions

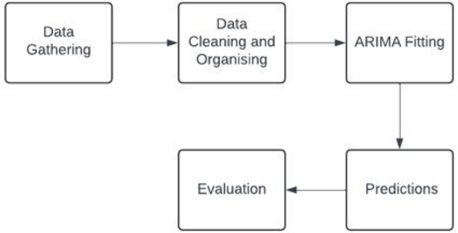


Fig. 1. The Time Series Pipeline.

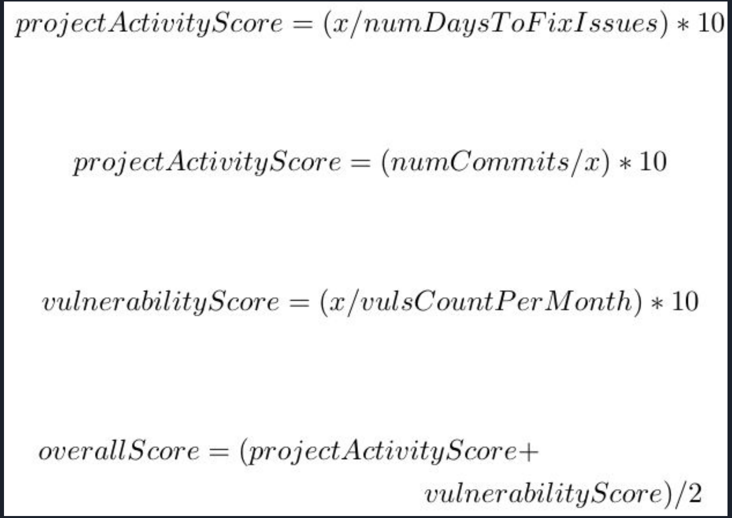
Project Activity

- ➔ GitHub API
  - ◆ All available data gathered
- ➔ Commits, average time-to-fix issues or both
- ➔ AutoARIMA prediction
- ➔ Final fitted value returned

Vulnerabilities

- ➔ NVD API
  - ◆ Call made for each keyword
- ➔ Number of vulnerabilities per month
- ➔ AutoARIMA Prediction
- ➔ Final fitted value returned

Risk Calculations



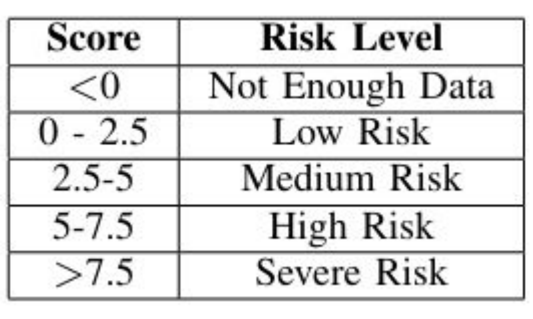
$$projectActivityScore = (x/numDaysToFixIssues) * 10$$

$$projectActivityScore = (numCommits/x) * 10$$

$$vulnerabilityScore = (x/vulsCountPerMonth) * 10$$

$$overallScore = (projectActivityScore + vulnerabilityScore)/2$$

- ➔ User-decided acceptable levels
- ➔ Options for project activity score
- ➔ Combination of project activity and vulnerability for overall score



Score	Risk Level
<0	Not Enough Data
0 - 2.5	Low Risk
2.5-5	Medium Risk
5-7.5	High Risk
>7.5	Severe Risk

Results

Example Final Graph

Key:

Severe Risk

🟥

High Risk

🟧

Medium Risk

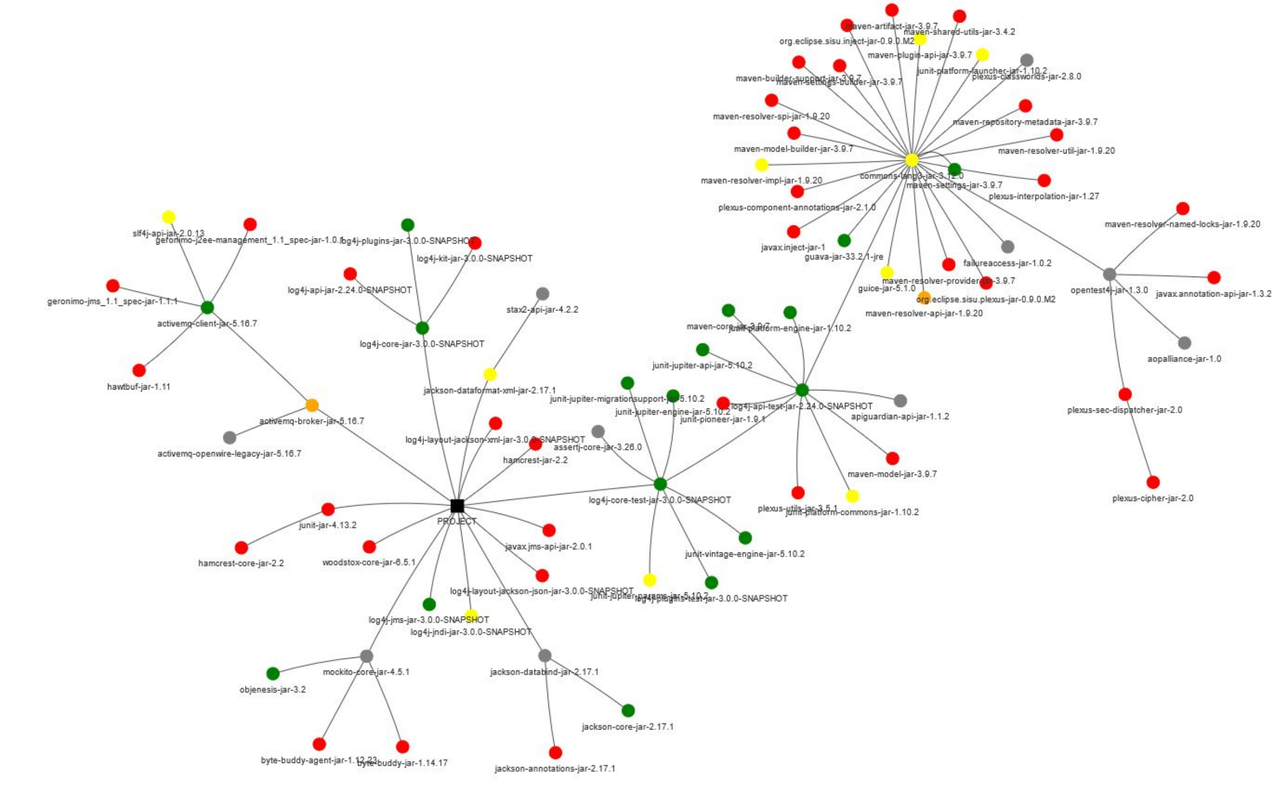
🟨

Low Risk

🟩

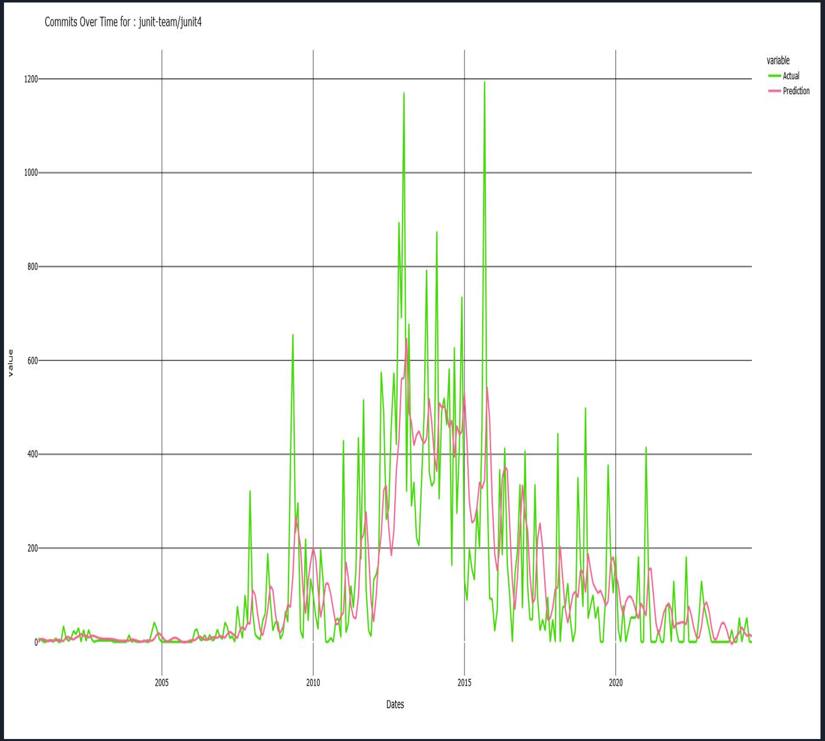
Not Enough Data

⬛


maven-artifact-jar-3.9.7
maven-shared-utils-jar-3.4.2
org.eclipse.sisu.inject-jar-0.9.0.M2
maven-plugin-api-jar-3.9.7
junit-platform-launcher-jar-1.10.2
plexus-classworlds-jar-2.8.0
maven-builder-support-jar-3.9.7
maven-settings-builder-jar-3.9.7
maven-resolver-spi-jar-1.9.20
maven-repository-metadata-jar-3.9.7
maven-model-builder-jar-3.9.7
maven-resolver-util-jar-1.9.20
maven-resolver-impl-jar-1.9.20
commons-lang3-jar-3.12.0
maven-settings-jar-3.9.7
plexus-interpolation-jar-1.27
plexus-component-annotations-jar-2.1.0
maven-resolver-named-locks-jar-1.9.20
javax.inject-jar-1
guava-jar-33.2.1-jre
failureaccess-jar-1.0.2
maven-resolver-provider-jar-3.9.7
guice-jar-5.1.0
org.eclipse.sisu.plexus-jar-0.9.0.M2
maven-resolver-api-jar-1.9.20
opentest4j-jar-1.3.0
javax.annotation-api-jar-1.3.2
aopalliance-jar-1.0
plexus-sec-dispatcher-jar-2.0
plexus-cipher-jar-2.0
slf4j-api-jar-2.0.13
geronimo-j2ee-management_1.1_spec-jar-1.0.1
log4j-plugins-jar-3.0.0-SNAPSHOT
log4j-kit-jar-3.0.0-SNAPSHOT
log4j-api-jar-2.24.0-SNAPSHOT
geronimo-jms_1.1_spec-jar-1.1.1
activemq-client-jar-5.16.7
stax2-api-jar-4.2.2
log4j-core-jar-3.0.0-SNAPSHOT
hawtbuf-jar-1.11
jackson-dataformat-xml-jar-2.17.1
activemq-broker-jar-5.16.7
activemq-openwire-legacy-jar-5.16.7
log4j-layout-jackson-xml-jar-3.0.0-SNAPSHOT
assertj-core-jar-3.26.0
hamcrest-jar-2.2
maven-core-jar-3.9.7
junit-platform-engine-jar-1.10.2
junit-jupiter-api-jar-5.10.2
junit-jupiter-migrationsupport-jar-5.10.2
junit-jupiter-engine-jar-5.10.2
junit-pioneer-jar-1.9.1
log4j-api-test-jar-2.24.0-SNAPSHOT
apiguardian-api-jar-1.1.2
maven-model-jar-3.9.7
log4j-core-test-jar-3.0.0-SNAPSHOT
plexus-utils-jar-3.5.1
junit-platform-commons-jar-1.10.2
PROJECT
junit-jar-4.13.2
hamcrest-core-jar-2.2
woodstox-core-jar-6.5.1
javax.jms-api-jar-2.0.1
junit-vintage-engine-jar-5.10.2
log4j-layout-jackson-json-jar-3.0.0-SNAPSHOT
junit-jupiter-params-jar-5.10.2
log4j-plugins-test-jar-3.0.0-SNAPSHOT
log4j-jms-jar-3.0.0-SNAPSHOT
log4j-jndi-jar-3.0.0-SNAPSHOT
mockito-core-jar-4.5.1
jackson-databind-jar-2.17.1
objenesis-jar-3.2
jackson-core-jar-2.17.1
byte-buddy-agent-jar-1.12.23
byte-buddy-jar-1.14.17
jackson-annotations-jar-2.17.1

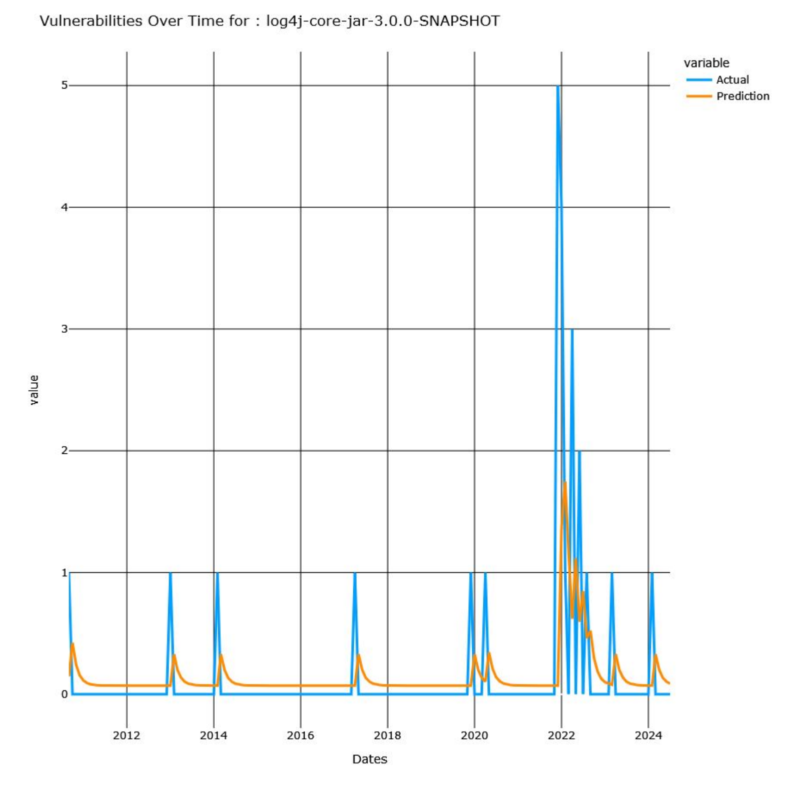
Example Graph of Project Activity Risk Prediction

- JUnit4 package commits over time
- Clear to see that activity has declined in recent years
- 2012 to 2017 were high commit years
  - At its height over 200 commits per month
- Prediction is fairly accurate as can be seen in the graph

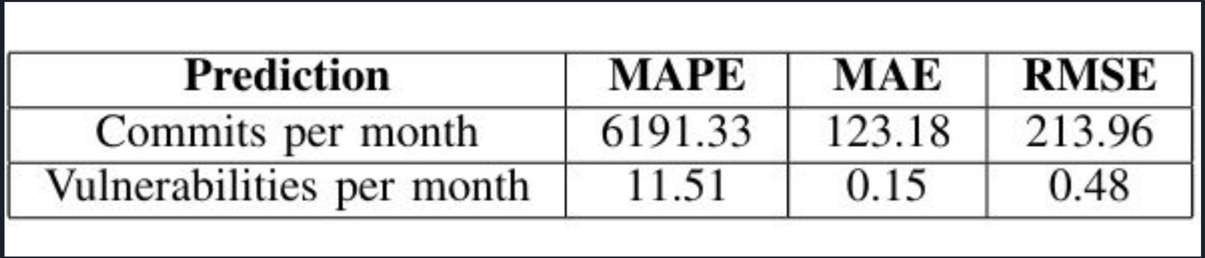


Example Graph of Vulnerability Risk Prediction

- ➔ Log4j vulnerabilities over time
- ➔ Clear where many were released in 2021
  - ◆ 5 released in one month
- ➔ Reasonably accurate predictions in subsequent months



Evaluations



Prediction	MAPE	MAE	RMSE
Commits per month	6191.33	123.18	213.96
Vulnerabilities per month	11.51	0.15	0.48

- Log4j sample project
- Project Activity Predictions > CVE Data Predictions
- Commonly used metrics
  - Mean Absolute Percentage Error (MAPE)
  - Mean Absolute Error (MAE)
  - Root Mean Squared Error (RMSE)
- Large fluctuations in commits
- Small fluctuations in vulnerability prediction

Technical Challenges

➔

Student Contribution

Róisín Ní Bhriain

- Gitlab setup + Issues
- Literature Review
- Software
- Practicum Paper First + Second Draft

Sneha Dechamma Mallengada Suresh

-

Conclusion

- ➔

References

Thank You For Listening!