

**ACHILLES: AUTOMATED TOOL FOR DETECTING AND
VISUALIZING NPM DEPENDENCY VULNERABILITIES**

อชิลล์ส เครื่องมือเพื่อการตรวจจับและการแสดงผลช่องโหว่ด้านความปลอดภัยของอีนพีเอ็มดีเพน
โดย

**MISS. VIPAWAN JARUKITPIPAT 6088044
MISS. WACHIRAYANA WANPRASERT 6088082
MR. KLINTON CHHUN 6088111**

**ADVISOR
DR. CHAIYONG RAGKHITWETSAGUL**

**CO-ADVISOR
DR. MORAKOT CHOETKIERTIKUL
ASST. PROF. DR. THANWADEE SUNETNANT**

**A Senior Project Submitted in Partial Fullfillment of
the Requirement for**

**THE DEGREE OF BACHELOR OF SCIENCE
(INFORMATION AND COMMUNICATION TECHNOLOGY)**

**Faculty of Information and Communication Technology
Mahidol University**

2020

ACKNOWLEDGEMENTS

To all the people who have been involved with the development until the completion of the project, we would like to dedicate this section to show our utmost gratitude. We are very thankful for their assistance throughout the long period of this project. First and foremost, we would like to express our appreciation towards Dr. Chaiyong Ragkhitwetsagul, our senior project advisor, and Asst. Prof. Dr. Thanwadee Sunetnanta as well as Dr. Morakot Choetkiertikul, our respectable co-advisors. Furthermore, we would like to thank Assist. Prof. Raula Gaikovina Kula, Bodin Chinthanet, and Assoc. Prof Takashi Ishio, our advisors from NAIST. Additionally, It would be outrageous of us to not thank our participants for partaking in our user study as well as those who participated in our online survey during the evaluation process. We would also like to thank the Faculty of Information and Communication Technology, instructors, staff, members for the support given to us. Lastly, we would like to thank, from the bottom of our heart, our dearest families for giving us the support we need during the making of this project. This project would not have been this successful if not for those mentioned and their offered support.

Miss. Vipawan Jarukitpipat

Miss. Wachirayana Wanprasert

Mr. Clinton Chhun

Achilles: Automated tool for detecting and visualizing npm dependency vulnerabilities

MISS. VIPAWAN JARUKITPIPAT 6088044 ITCS/B

MISS. WACHIRAYANA WANPRASERT 6088082 ITCS/B

MR. KLINTON CHHUN 6088111 ITCS/B

B.Sc.(INFORMATION AND COMMUNICATION TECHNOLOGY)

PROJECT ADVISOR: DR. CHAIYONG RAGKHITWETSAGUL

ABSTRACT

In contemporary software development, utilizing third-party libraries is common practice for developers to create high-quality software at a reduced cost. According to the State of Open Source Security Report 2020, the JavaScript ecosystem has driven the growth in open source packages. From the end of 2018 to the end of 2019, npm, which is a package manager for Node.js packages, grew by over 33% [1]. Nonetheless, third-party dependency vulnerabilities have become the Achilles's heel of most modern software systems. These vulnerabilities can come from direct dependencies or when the dependencies use other dependencies (indirect dependencies). A study by snyk.io shows that 86% of npm package vulnerabilities are discovered in indirect dependencies.

Several tools (e.g., GitHub Dependabot, npm audit) are developed to assist the developers in keeping their dependencies up-to-date, yet they have different ways of visualization. We propose the Achilles tool which detects and visualizes npm project's dependency vulnerabilities. Achilles assists programmers in comprehending and analyzing potential risks of vulnerabilities of npm packages via the dependency graph visualization and analysis report. We have performed an evaluation using a user study and found that the graph visualization of Achilles helps support developers' decisions on prioritizing vulnerability to fix by providing more information about complexity and direct/indirect dependencies compared to the state-of-the-art tools. The tool can also detect vulnerabilities currently existing in several most-starred GitHub open source projects.

KEYWORDS: VULNERABILITY, DEPENDENCY, NPM, VISUALIZATION

อคิลีส เครื่องมือเพื่อการตรวจจับและการแสดงผลช่องโหว่ด้านความปลอดภัยของอีนพีเอ็มดีเพนเดนซ์

นางสาว วิภาวรรณ จากรุกิจพัฒนา 6088044 ITCS/B

นางสาว วชิรญาณ์ วันประเสริฐ 6088082 ITCS/B

นาย คลินตัน ชน 6088111 ITCS/B

วท.บ. (เทคโนโลยีสารสนเทศและการสื่อสาร)

อาจารย์ที่ปรึกษาโครงการ: ดร. ชัยยงค์ รักขิตเวชสกุล

บทคัดย่อ

ในปัจจุบันการพัฒนาซอฟต์แวร์โดยใช้ไลบรารีของบุคคลที่สามเป็นแนวทางเพื่อสร้างซอฟต์แวร์อย่างรวดเร็วและลดต้นทุน รายงานจาก State of Open Source Security 2020 กล่าวว่า ระบบниковของ JavaScript ได้ขับเคลื่อนการเติบโตของแพ็กเกจ Open Source ซึ่งส่งผลให้ระบบขั้นการแพ็คเกจสำหรับ Node.js หรือ npm โดยที่ 33% ตั้งแต่ปลายปี 2018 ถึง 2019 ถึงอย่างไร ก็ตามช่องโหว่ด้านความปลอดภัยของ ไลบรารีกล้ายเป็นปัญหาสำหรับนักพัฒนา ช่องโหว่เหล่านี้สามารถมาจาก ไลบรารีที่เรียกใช้โดยตรง (Direct dependency) หรือ เมื่อ ไลบรารีเรียกใช้ ไลบรารีตัวอื่น (Indirect dependency) จากการศึกษาของ Snyk.io พบว่า 86% ของช่องโหว่ของแพ็กเกจ npm เกิดจากการใช้งาน ไลบรารีแบบทางอ้อม (Indirect dependency)

ผู้วิจัยพบว่ามีการพัฒนาเครื่องมือ (เช่น GitHub, Dependabot, npm-audit) จำนวนมาก เพื่อช่วยเหลือนักพัฒนาให้ติดตามข้อมูลของ ไลบรารี อย่างไรก็ตามเครื่องมือเหล่านี้มีการแสดงผลที่แตกต่างกัน ผู้วิจัยจึงเกิดแนวคิดในการสร้างเครื่องมือ Achilles ที่จะตรวจสอบและแสดงช่องโหว่จากการใช้ ไลบรารีทั้งทางตรงและทางอ้อม อีกทั้งยังสามารถช่วยนักพัฒนาในการทำความเข้าใจและวิเคราะห์ความเสี่ยงที่อาจเกิดขึ้นจากการนำแพ็กเกจ npm ที่มีช่องโหว่มาใช้ ผ่านทางการแสดงผลความสัมพันธ์ของแพ็กเกจในรูปแบบกราฟและรายงานการวิเคราะห์ ผู้วิจัยได้ทำการประเมินผลโดยการศึกษาจากผู้ใช้และพบว่าการแสดงผลของ Achilles สามารถช่วยให้การตัดสินใจของนักพัฒนาในเรื่องการจัดลำดับความสำคัญของช่องโหว่เพื่อแก้ไขและให้ข้อมูลเพิ่มเติมเกี่ยวกับความซับซ้อนและการพัฒนาโดยตรง (Direct dependency) / การพัฒนาโดยทางอ้อม (Indirect dependency) เมื่อเทียบกับเครื่องมืออื่น ๆ นอกจากนี้ Achilles ยังสามารถตรวจสอบช่องโหว่ที่มีอยู่ใน GitHub open source โปรดักท์ที่เป็นที่นิยม

CONTENTS

	Page
ACKNOWLEDGMENTS	ii
ABSTRACT	iii
LIST OF TABLES	viii
LIST OF FIGURES	ix
1 INTRODUCTION	1
1.1 MOTIVATION	1
1.2 PROBLEM STATEMENTS	2
1.3 OBJECTIVES OF THE PROJECT	3
1.4 SCOPE OF THE PROJECT	3
1.5 EXPECTED BENEFITS	3
1.6 ORGANIZATION OF THE DOCUMENT	3
2 BACKGROUND	5
2.1 FUNDAMENTALS	5
2.1.1 DEPENDENCIES VULNERABILITIES.....	5
2.1.2 OVERVIEW OF DEPENDENCIES VULNERABILITY ANALYSIS TOOLS	6
2.2 TOOLS AND TECHNIQUES	8
2.2.1 GITHUB	8
2.2.2 NODE.JS.....	8
2.2.3 MONGODB	9
2.2.4 D3.JS LIBRARY	9
2.2.5 GRAPHQL	9
2.3 LITERATURE REVIEW.....	12
2.3.1 LAGS IN LIBRARY DEPENDENCIES UPDATE	12
2.3.2 DEPENDENCIES VISUALIZATION.....	13
2.3.3 DATASET FOR VISUALIZING NODE.JS DEPENDENCY ECOSYSTEM IN GITHUB	16

2.4	CHAPTER SUMMARY	18
3	ANALYSIS AND DESIGN	19
3.1	ACHILLES: A TOOL FOR NPM ECOSYSTEM VISUALIZATION AND VULNERABILITY DETECTION	19
3.2	SYSTEM ARCHITECTURE OVERVIEW	20
3.3	USE CASE ANALYSIS.....	24
3.4	STRUCTURE CHART	26
3.5	SYSTEM ANALYSIS	27
3.5.1	DATA FLOW DIAGRAM LEVEL 0 (CONTEXT DIAGRAM)	27
3.5.2	DATA FLOW DIAGRAM LEVEL 1	29
3.6	COMPARISON TO RELATED WORK	30
3.7	PROJECT TIMELINE, CURRENT PROGRESS, AND FUTURE WORK.....	32
3.7.1	PROJECT TIMELINE.....	32
3.8	CHAPTER SUMMARY	33
4	IMPLEMENTATION	34
4.1	RETRIEVE USER'S REPOSITORIES.....	34
4.1.1	GET THE USER'S GITHUB REPOSITORIES	34
4.1.2	FILTER FOR NPM PROJECTS	36
4.1.3	STORING THE SELECTED REPOSITORY	37
4.2	VISUALIZATIONS	38
4.2.1	GENERATING NODES AND EDGES	38
4.2.2	GENERATING TOOLTIPS	45
4.3	CREATE REPORT	45
4.3.1	VULNERABILITY INFORMATION DATA TEMPLATE	48
4.4	SEMVER-EXISTING-MAX.....	50
5	EVALUATION RESULTS	54
5.1	EVALUATION METHODOLOGY	54
5.1.1	THE ONLINE SURVEY.....	54
5.1.2	THE USER STUDY	54
5.2	ONLINE SURVEY RESULT	61

5.2.1	LEVEL OF CONCERN REGARDING SECURITY VULNERABILITY SOURCES.....	61
5.2.2	PRIORITIZATION FACTORS FOR VULNERABILITIES UPDATES	61
5.2.3	DECISION TO UPDATE VULNERABLE DEPENDENCIES	65
5.2.4	FEEDBACK FROM ONLINE SURVEY	66
5.3	PARTICIPANTS' DEMOGRAPHIC DATA	69
5.4	USER STUDY RESULT	70
5.4.1	RESULTS AND ANALYSIS	70
5.4.2	ANSWER TO RESEARCH QUESTIONS	82
5.5	ANALYSIS OF GITHUB PROJECT	83
5.5.1	MOST STARRED GITHUB PROJECTS.....	84
5.5.2	MOST DEPENDENT NPM PROJECTS	85
6	CONCLUSIONS	88
6.1	PROBLEMS AND LIMITATIONS	88
6.2	THREATS TO VALIDITY	88
6.3	FUTURE WORK	89
6.3.1	POTENTIAL PERFORMANCE OPTIMIZATION	89
6.3.2	POTENTIALLY BETTER VISUALIZATION METHOD	89
6.4	CONCLUSION	90
APPENDIX A		92
APPENDIX B		99
APPENDIX C		124
APPENDIX D		140
APPENDIX E		148
APPENDIX F		205
REFERENCES		218
BIOGRAPHIES		221

LIST OF TABLES

	Page
Table 3.1: Comparison of Dependencies Vulnerability Detection Technique and Tools.....	31
Table 4.1: Parameter	35
Table 4.2: Parameter	35
Table 4.3: Parameter	35
Table 5.1: Characteristics of vulnerabilities in Test 1.....	58
Table 5.2: Characteristics of vulnerabilities in Test 2.....	60
Table 5.3: Participants' Demographic	69
Table 5.4: The Result of Achilles Test Case No. 1 (Complexity).....	71
Table 5.5: Factors for Prioritizing Package Updates	72
Table 5.6: The Result of npm audit Test Case No. 1 (Complexity)	74
Table 5.7: Factors for Prioritizing Package Updates	75
Table 5.8: The Result of Achilles Test Case No. 2 (Direct/ Indirect).....	77
Table 5.9: Factors for Prioritizing Package Updates	78
Table 5.10: The Result of npm audit Test Case No. 2 (Direct/ Indirect).....	80
Table 5.11: Factors for Prioritizing Package Updates	81
Table 5.12: Comparison of Developers' Decisions	82
Table 5.13: Comparison of Developers' Decisions	83
Table 5.14: showing the prioritization results after using npm audit and Achilles ...	83
Table 5.15: 10 Most Stars GitHub Project Used in the Study	84
Table 5.16: 10 Most Dependent GitHub Project Used in the Study.....	84
Table 6.1: Comparing the current and the proposed methods.....	90

LIST OF FIGURES

	Page
Figure 2.1: Dependabot alert's example	6
Figure 2.2: Dependabot pull request's example	6
Figure 2.3: Dependabot Change Recommendation example.....	7
Figure 2.4: npm audit report example	8
Figure 2.5: Directional Force Layout Diagram.....	10
Figure 2.6: Example of schema of securityVulnerabilities	11
Figure 2.7: VerXCombo - Parallel Sets Visualization	15
Figure 2.8: Coexistence Logic.....	16
Figure 2.9: React Overview of Sol Mantra [2]	16
Figure 3.1: Proposed npm Ecosystem Network Visualization.....	20
Figure 3.2: System Architecture	21
Figure 3.3: A mock-up of the Achilles vulnerabilities analysis report.....	23
Figure 3.4: Use case Diagram	25
Figure 3.5: Structure Chart.....	26
Figure 3.6: Data Flow Diagram Level 0 (Context Diagram).....	28
Figure 3.7: Data-flow diagram Level 1	29
Figure 3.8: Project Timeline	32
Figure 4.1: GHSA vs npm security advisory database	42
Figure 4.2: Chain of Dependencies of Package A	51
Figure 4.3: Chain of Dependencies of Package A with Different Version.....	52
Figure 5.1: Procedures of User Study	56
Figure 5.2: Achilles graph visualization of Test 1	59
Figure 5.3: Achilles graph visualization of Test 2	60
Figure 5.4: Participants' Survey	62
Figure 5.5: The Students' Level of Concern Regarding Different Sources.....	62
Figure 5.6: The Developers' Level of Concern Regarding Different Sources	63
Figure 5.7: The Students' Library Update Prioritization Factors	64

Figure 5.8: The Developers' Library Update Prioritization Factors.....	64
Figure 5.9: The Students' Decision on Updating Vulnerable Dependencies.....	65
Figure 5.10: The Developers Decision on Updating Vulnerable Dependencies	66
Figure 5.11: Bar graphs showing Direct and Indirect Vulnerable Dependency for Top 10 Most starred Repositories on GitHub.....	86
Figure 5.12: Bar graphs showing Direct and Indirect Vulnerable dependencies for Top 10 Most Dependent JavaScript Libraries in npm Registry	87

CHAPTER 1

INTRODUCTION

1.1 Motivation

In modern software development, developers usually depend on third-party libraries to provide specific functionality in their applications. The node package manager (npm) dependency network is evolving at a growing rate with over 1.3 million packages that have enabled over 12 million end-users to use them in their projects [3]. Libraries aim to save both time and resources and reduce redundancy by taking advantages of existing quality implementations [4].

Third-party dependency vulnerabilities become a concern for software developers. In a 2018 GitHub report, more than four million vulnerabilities were raised to the attention of the developers of over 500,000 GitHub repositories [5]. Not only the direct users of these software artifacts, but also the software ecosystem are exposed to the risk of vulnerabilities. Heartbleed [6] and ShellShock [7] are examples of the severe vulnerabilities which caused widespread damage to diverse software ecosystems which include both direct and indirect adopters.

However, developers are slow to update their vulnerable dependencies ,which is sometimes due to bundled release of the fix, management, and process factors. Kula et al.’s research shows that 85% of the studied systems still keep their outdated dependencies, and 69% of the interviewed developers are unaware of their vulnerable dependencies. In addition, developers are unlikely to prioritize a library update due to extra workload and responsibilities [4].

Chinhanet et al.’s research also revealed that npm developers are slow to respond to the threat of a vulnerability. It usually takes 4 to 11 months to update vulnerable dependencies [8]. This research disclosed that fixing release update is not consistent with the client-side fix release update. Possible causes of lags between vulnerable release and fixing release update is developer’s unawareness of the fixing release. Since the fixed

code tends to be small in size, developers of the library bundle the fixing with other updates, and they do not highlight the fixing updates in the update note [8].

Since developers become more aware of vulnerable dependencies, automated dependency updates tools are developed, e.g., Dependabot and Snyk.io. These tools help developers to check for outdated and insecure dependencies and send the vulnerability report to the user. They also create a pull request, i.e., a code review and merge request on GitHub platform, for ease of review and merge the update. Moreover, visualization tools are developed to assist system maintainers in making the decision whether to update or introduce a new third party library since incompatibility between internal library dependencies might occur [9]. Todorov et al.'s visualization tool, SOL Mantra, presents an opportunity to update libraries using a visualization of coexistence logic. It demonstrates whether libraries should be updated. the visualization adopt a solar system metaphor, which includes system, library, and coexistence between libraries [2].

From the literature review, the existing visualization only displays dependencies relationship for only one particular project. However, it does not show relationships among the whole npm ecosystem. Vulnerability detection tools on GitHub currently can report only at the current project's level itself. It does not display potential risk which one particular project might have due to its indirect adoption of a vulnerable library via other libraries. Although there exists a tool that can visualize the npm package, that tool still does not visualize in version level and state of ecosystem overtime [10]. Hence, we would like to propose a visualization tool which can display the npm ecosystem and produce a report which provides information about potential risks from vulnerability dependencies that a project might have. Also, the proposed visualization tool can demonstrate the state of the system at several points in time in order to exhibit the spreading of npm vulnerabilities.

1.2 Problem Statements

This project tackles the following problems in visualizing npm vulnerabilities:

1. It is challenging to understand the complexity of dependencies in an npm project.
2. Existing visualizations do not represent the potential security risks that a partic-

ular project might have due to multiple levels of dependencies, i.e., the chain of dependencies.

1.3 Objectives of the project

The objectives of the project are as follows:

1. Create a method for detecting and visualizing the complexity, direct and indirect vulnerabilities of dependencies in an npm project.
2. Create a prototype that can analyze a GitHub project according to the data in the npm registry and GitHub security Advisory.
3. Conduct a user study to investigate whether graph visualization would affect the user's decision on prioritizing vulnerability update.

1.4 Scope of the project

The project falls under the following scope:

1. The proposed system allows only login with GitHub account.
2. The proposed system runs as a web application
3. The proposed system supports only npm packages.

1.5 Expected Benefits

This project provides the following expected benefits:

1. Providing a new visualization of relationships in npm ecosystem network
2. Helping computer science students and software developers to be aware of potential vulnerable dependencies risks in their software system.

1.6 Organization of the document

The document consists of 6 parts including Introduction (Chapter 1), Background (Chapter 2), Analysis and Design (Chapter 3), Implementation (Chapter 4), Evaluation Results (Chapter 5), and Conclusion (Chapter 6).

The Introduction chapter includes motivation, problem statements, objectives, scope, expected benefits, and organization of the document. The Background chapter describes the overview of the project, which has fundamentals and related work. Analysis and design chapter contains work procedures, which are methodology, system architecture, structure chart, and system analysis. Implementation includes discussions of steps that the system is implemented. Evaluation Results consists of the evaluations of Visualization of npm ecosystem using Achilles, real software project, and by users. The last chapter is Conclusion that includes the conclusion, problems and limitations, and future work.

CHAPTER 2

BACKGROUND

This chapter provides the background and required knowledge for completing this project. It consists of 3 sections. The first section is Fundamentals section, which provides the basic knowledge of the project. The second section is Tools and Techniques section, which explains the tools and techniques that are applied into the project. The third section is Literature Review section that is the section of summary of the research studies which are related to the project.

2.1 Fundamentals

This section is to provide the basic knowledge of the project including third-party vulnerabilities, an overview of third-party vulnerability and detection techniques.

2.1.1 Dependencies Vulnerabilities

Third-party libraries have played an important role in contemporary software development. Developers highly rely on third-party libraries to provide a specific functionality in their application, especially in JavaScript ecosystem [4]. Node Package Manager (npm) which is a package manager for the JavaScript Programming language is initially released in 2014. In January 2017, snyk.io reported that 250,000 packages were available in npm registry [3]. The number of libraries reached 1 millionth package milestone in June 4th 2019 which is almost 3 times more than it was in 2017 [3]. npm becomes one of the largest registries of Open Source projects which support easy share packages modules of code for JavaScript developers. Libraries are intended to reduce resources, time, and redundancy by exploit existing quality implementations.

As software development have grown larger and more complex, the number of third-party dependencies have grown significantly. Several libraries are in constant evolution. One of growing concerns for the software developer is third-party dependencies vulnerabilities. New versions are released to fix defects, patch vulnerabilities and en-

hance features. When a security vulnerability due to the source code of the dependency is found, a fix is released, and developers are responsible to update their project's dependencies [11].

2.1.2 Overview of Dependencies Vulnerability Analysis Tools

Vulnerability detection tools on GitHub currently can report only at the current project's level itself.

Dependabot is a GitHub application that facilitates in updating dependency automatically. Currently, Dependabot has created more than 7 millions pull requests to help users in updating dependency up-to-date (7,312,824). Dependabot works in three ways. Firstly, Dependabot checks for dependency updates every day. Dependabot pull downs user's dependency files and looks for any updated or insecure requirements. Secondly, Dependabot is able to open pull requests for users' repository. If there is any user's dependencies that are out-of-date, Dependabot opens an individual pull request to update each one of the dependencies. Lastly, user can review and merge the process. After Dependabot made a pull request, users can check whether the tests pass, scan the included changelog and release notes, and merge the pull request or skip the new version. The Figure 2.1, 2.2, and 2.3 below show examples of Dependabot reports.



Figure 2.1: Dependabot alert's example

Snyk is a developer security solution that allows companies to use open source code and stay safe and secure. Snyk is the only solution that can transparently and proactively discover and fix vulnerabilities and license violations in open source dependencies and Docker images. Snyk is not totally free security solution. There are a different between user roles that are using snyk. Free accounts and starter plans have only administrators, while other paid plans allow adding employees as collaborators. Contributors



Figure 2.2: Dependabot pull request's example

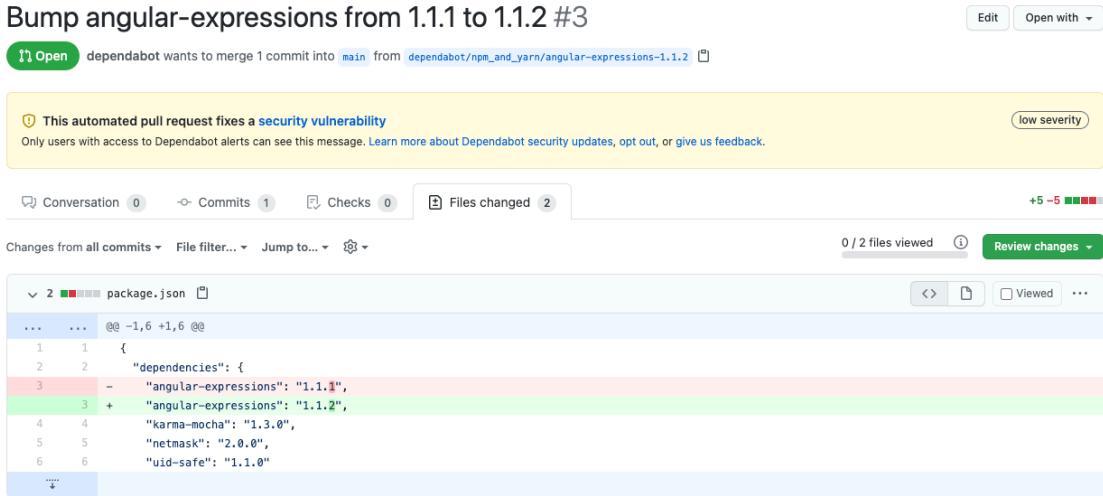


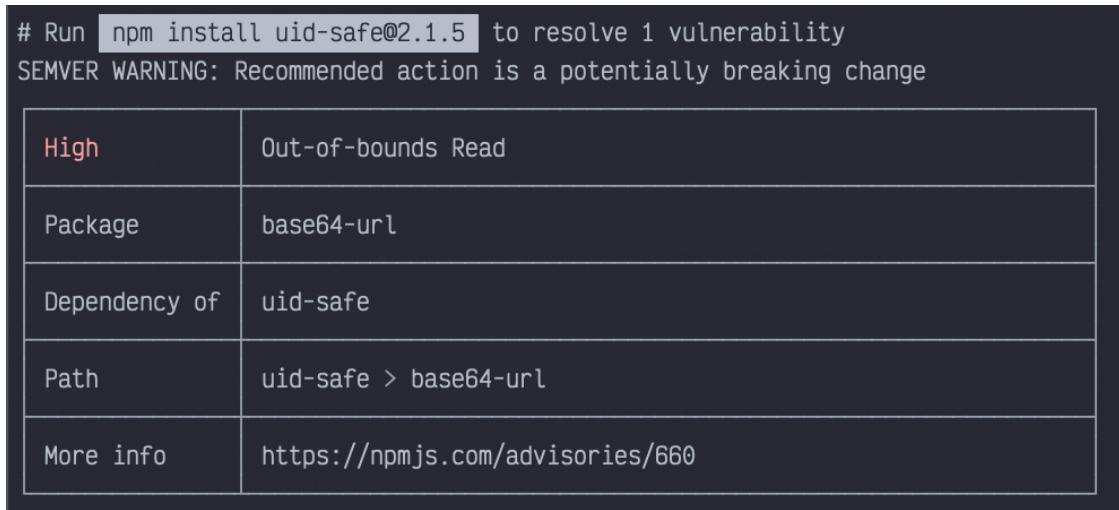
Figure 2.3: Dependabot Change Recommendation example

can view and contribute to the project, but cannot access billing information or invite team members. Snyk detects project vulnerabilities by scanning user projects, testing vulnerabilities, and importing project snapshots. Snyk regularly scans image snapshot dependencies based on user configurations (daily or weekly) and updates users when new security vulnerabilities (email or Slack) are discovered. Snyk reports are only available to subscribers. The report area provides data and analysis for all user projects and displays historical data and summary data about projects, issues, dependencies, and licenses.

npm-audit is the audit command that asking for report of vulnerabilities that are found in the client's project. If there are vulnerabilities are found, npm will provide the impact and remediation information. The security vulnerabilities that npm audit uses can be found in <https://www.npmjs.com/advisories>. In the user's npm project, users can view the report by typing npm audit in the command.

The information provided by npm audit is shown in the Figure 2.4. They can be described as follow:

1. **npm install uid-safe@2.1.5**: is the command line to fix this vulnerability
2. **Severity: High** is the level of severity. Level of severity is divided into Critical, High, Moderate, and Low
3. **Description: Out-of-bounds Read** is the description of the vulnerability



The screenshot shows the output of an npm audit command. It starts with a warning to run `npm install uid-safe@2.1.5` to resolve a vulnerability. Below this, it says 'SEMVER WARNING: Recommended action is a potentially breaking change'. A table follows, detailing the vulnerability:

High	Out-of-bounds Read
Package	base64-url
Dependency of	uid-safe
Path	uid-safe > base64-url
More info	https://npmjs.com/advisories/660

Figure 2.4: npm audit report example

4. **Package:** base64-url is the name of package that has vulnerability
5. **Dependency of:** uid-safe is the package that depends on vulnerable package
6. **Path:** uid-safe > base64-url is the path to code that have vulnerability
7. **More info:** it is a link that leads to security report.

2.2 Tools and Techniques

This section explains several tools and techniques which are applied into the project.

2.2.1 GitHub

Git is a version control system that keeps track of the changes of files in the repositories. GitHub is the largest code archive based on Git in the development community. GitHub hosts over 190 million repositories, including at least 28 million public repositories, and has over 40 million users, making GitHub, the largest host of source code in the world [5]. Repositories can be configured to be private or public, and they can be shared with other developers. Hence, it is one of the practical tools for group or organization work.

2.2.2 Node.js

Node.js is an open-source server-side runtime environment using JavaScript language. Node.js is primarily used to create a web-server. Conventionally, in web development, JavaScript is a client-side language. Hence, it has to use another language such as PHP, Ruby, C#, Python, to develop server-side. Node.js is developed by the community, and Ryan Dahl—the initiator of the project. Node.js is built on Google Chrome's JavaScript V8 engine. It is created for building fast and scalable network applications. Moreover, it accepts JSON, which is a standard format for exchanging data between a browser and a server [12].

2.2.3 MongoDB

MongoDB is an open-source document database. It is a non-relational and schema-less database (i.e., NoSQL), and it stores data as JSON format which mean that numbers or types of columns are not required before inserting the data. The data is stored as a pair of key and value, which is called a document, and many documents in MongoDB are stored as a collection. Since MongoDB is flexible and scalable, several well-known companies are using MongoDB, such as Adobe, Google, and ebay [13].

2.2.4 D3.js library

D3.js, which stands for Data-Driven Documents, is a JavaScript library for visualizing data in web browsers. D3.js utilizes Scalable Vector Graphics (SVG), HTML 5, Canvas, and Cascading Style Sheets (CSS) standards [14]. In D3, users can create power visualization as well as adopt interaction techniques with a data-driven approach to manipulate the Document Object Model (DOM). With D3, users can design an appropriate visual interface, which is suitable for the data. In this proposal, the Directional Force Layout Diagram [15] is adopted to exhibit the relationship among the chain of dependencies. In the graph as Fig 2.5, the direction of the connections is critical — the node that has arrow pointed to means that other nodes are dependent on this particular node. From the figure, if Mikey is removed, Elric and Henry will be affected.

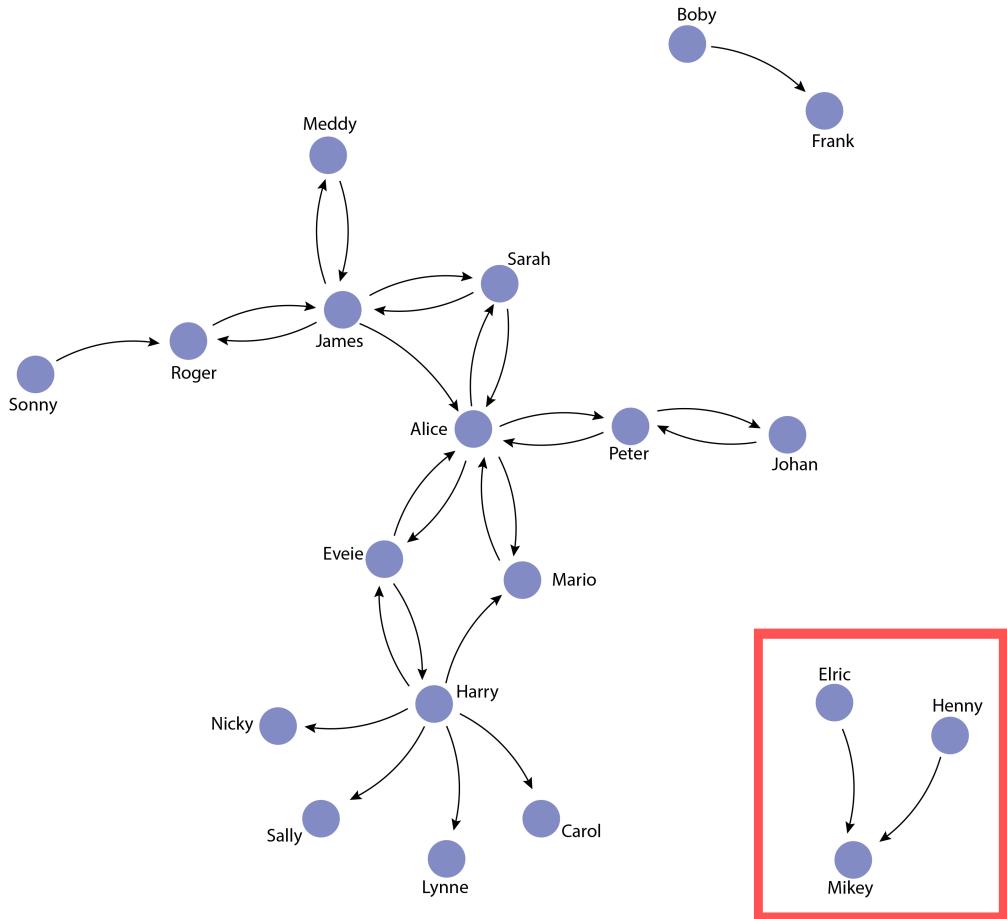


Figure 2.5: Directional Force Layout Diagram

2.2.5 GraphQL

GraphQL is a data query language developed by Facebook since 2012. GitHub provides GraphQL for its API v4 since it offers more flexibility and provides ability to define the data that user want to fetch more precisely. In term of flexibility, GraphQL allows user to replace multiple REST requests with a single call to fetch the data using nested fields. GraphQL provides schema for specifying available data and types of parameter that user need to send in order to fetch specific data. User can query the schema for details about the schema (Figure 2.6).

For example, we want to fetch some attributes of securityvulnerabilities: package name and advisory ghsId of package rpi

Software Vulnerabilities documented by GitHub
Security Advisories

TYPE

SecurityVulnerabilityConnection!

ARGUMENTS

after: String

Returns the elements in the list that come after the specified cursor.

before: String

Returns the elements in the list that come before the specified cursor.

ecosystem: SecurityAdvisoryEcosystem

An ecosystem to filter vulnerabilities by.

first: Int

Returns the first n elements from the list.

last: Int

Returns the last n elements from the list.

Figure 2.6: Example of schema of securityVulnerabilities

```
{
  securityVulnerabilities(first: 100, ecosystem: NPM, package: "rpi") {
    #totalCount
    nodes {
      package {
        name
      }
      advisory {
        ghsaId
      }
    }
  }
}
```

The response back will look like this.

```
{
  "data": {
    "securityVulnerabilities": {
      "nodes": [
        {
          "package": {
            "name": "rpi"
          },
          "advisory": {
            "ghsaId": "GHSA-vf26-7gjf-f92r"
          }
        }
      ]
    }
  }
}
```

2.3 Literature Review

The literature review section is a summary of the researches which are related to the project.

2.3.1 Lags in library dependencies update

Kula et al.'s empirical research investigates the extent to which developers update their library dependencies [4]. They conducted an empirical study on library migration, including 4,600 GitHub software projects and 2,700 library dependencies. The result

reveals that even though several systems depend utterly on library dependencies, 81.5% of the studied systems still keep their outdated dependencies. In the case of updating a vulnerable dependency, the study shows that although developers are affected, they are not likely to react to a security advisory. From the interview, 69% of the interviewees declared that they are ignorant of their vulnerable dependencies. Furthermore, library update is not the priority task for developers since it is considered to be extra workload and responsibility. This study concludes that updating library dependencies is not typical for developers despite the heavy dependence on these libraries.

Security vulnerabilities in third-party dependencies become an expanding concern for both affected developers and the entire software ecosystem. Previous studies show that developers respond to the threat of exposure slowly [4]. Chinthanet et al. conduct an empirical investigation to identify lags that may occur between the vulnerable release and its fixing release in order to promote quick adoption and propagation of a fixing release [8]. In a preliminary study of 131 fixing releases of npm projects on GitHub, they notice that the fixing release is often bundled with the other 92.86% of commits unrelated to a fix. Furthermore, they compare the fixing release update with changes on the client-side fixing release update. They conduct an empirical study of the adoption and propagation tendencies of 188 fixing releases that impact throughout a network of 882,222 npm packages. They find that the later library is updated, the more migration effort is required even if the patch landing was quick. In addition, they find that factors, including the fixing release landing branch, and the severity of the vulnerability, influences its propagation. This study concludes that there are factors that create lags in the release adoption propagation of npm vulnerability fixes. The research lays the groundwork for future research on how to mitigate propagation lags in an ecosystem.

2.3.2 Dependencies Visualization

Utilizing third-party libraries becomes common in software development since it helps lower development time and cost by reusing the implemented software. Dependencies occur when code are reused from other libraries and/or when a function is called by other libraries, which, as they increase over time, become strenuous to manage and avoid compatibility issues or bugs. When newer versions release, new features, fix releases,

and quality improvements are introduced. However, it is challenging for large software with many dependencies to decide whether to update a new version of the library or not due to the library's potential incompatibility with the existing source code. Hence, researchers have invented visualizations, i.e., VerXCombo and SoL Mantra, to assist the system maintainers in the decision-making process by providing more information about each opportunity update's complexity.

Yano et al.'s research [9] found that it would be tough for maintainers to decide whether to update or introduce new third-party libraries into the system since there is a vast range of Open Source System (OSS) libraries. For instance, system maintainers need to consider how this new library will best fit the existing dependency environment. Incompatibility between internal library dependencies may cause complicate adoption. Therefore, system maintainers especially need adequate assurance of any candidate library release. To assist system maintainers in determining the best-fit combination of libraries, they proposed VerXCombo (Version X Combination). VerXCombo platform can assist system maintainers by mining popular library dependency patterns of similar systems. Through data interactions, VerXCombo leverages parallel sets to break-down large and complex datasets into distinguishable patterns of 1) popular and 2) latest library dependency release combinations as shown in Figure 2.7 . VerXCombo is a web-application that was built by HTML5 and JavaScript for the front-end and Apache Tomcat and a Neo4j5 graph database for the server backend. In this research, the researchers populated the VerXCombo database with systems that depend on java libraries that are managed and hosted on the Maven 2 Super Repository. They analyzed library dependency information from 4,367 projects hosted on GitHub. They used an extension of the Pomwalker tool to extract system and library dependencies from respective pom.xml files.

Most developers may care about untested and early bugs or new releases; still, many other factors such as the compiler, development environment, and programming language influence a system maintainer's decision to update a library. In this work, the researchers specifically target existing system dependency libraries to reduce library incompatibility issues. The future work of this research is gathering feedbacks and implements to use in real-world system maintainers.

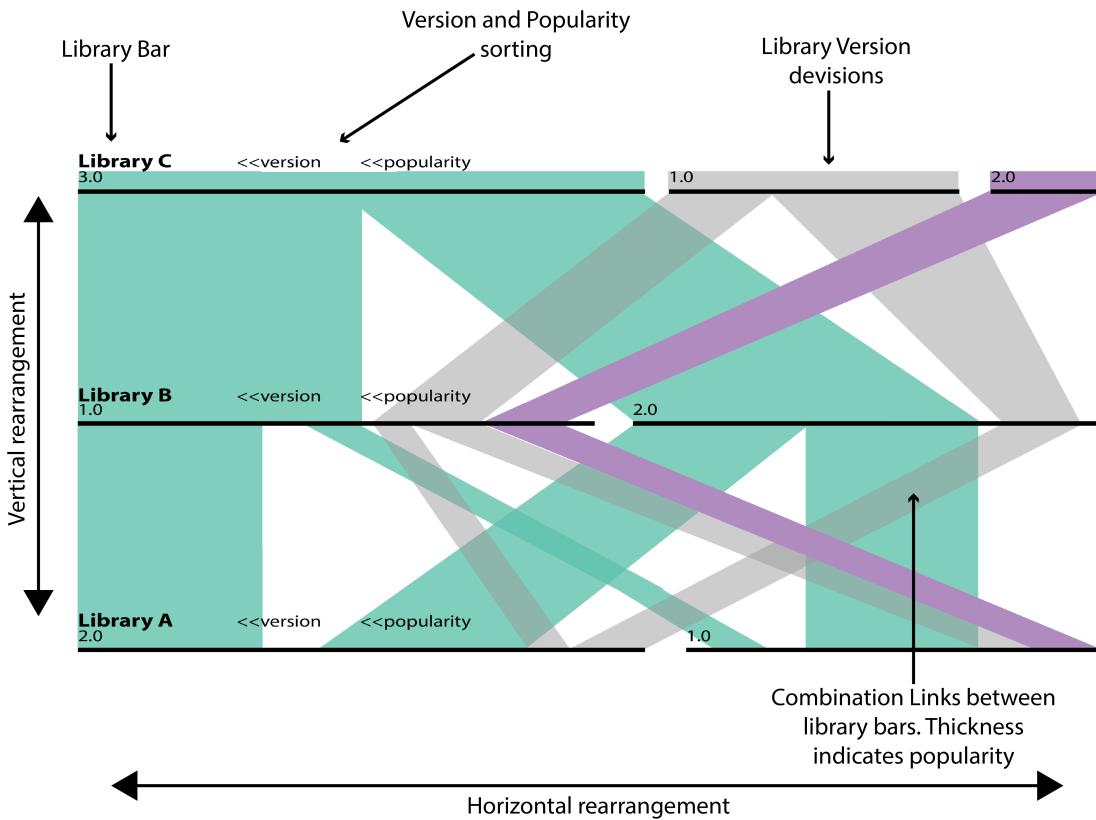


Figure 2.7: VerXCombo - Parallel Sets Visualization

Todorov et al. [2] presents an opportunity update of libraries that visualization by using coexistence logic as shown in Figure 2.8 . They address the issue with updating libraries and propose the peculiar software library mantra tool to demonstrate which libraries are up-to-date or should be updated. In their concepts, they select the layout of the solar system metaphor for visualization. This visualization system includes system, library, and coexistence, which means the relationship between two libraries where they use a similar library or system.

To illustrate their visualization of the software library mantra tool, the core is represented as a system that comes with planets used by the software system. Every planet has an outdated flag by using color to show the library is obsolete or up-to-date. For example, the jQuery library is red, which means it an outdated library, but if the jQuery library is green, which means it an up-to-date library. Moreover, some planet have their own moons to represent coexisting libraries.

Figure 2.9 shows the visualization of React. React system overview with a total

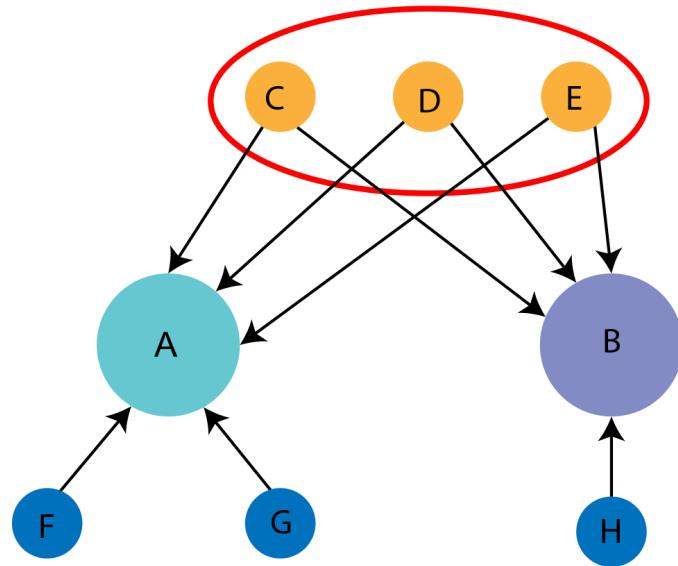


Figure 2.8: Coexistence Logic

of 5 dependencies, represented by the 5 planets orbiting the core. Two of them are green colored (up-to-date) which is prop-types and create-react-class, and 3 are red (outdated) which is loose-envify, object-assign, and fbjs. To look inside of Loose-envify can see that it has 100% coexistence with prop-types and object-assign. Furthermore, fbjs has 3 coexisting packages - 95.7% cc with prop-types and object-assign, and 23.16% with loose-envify.

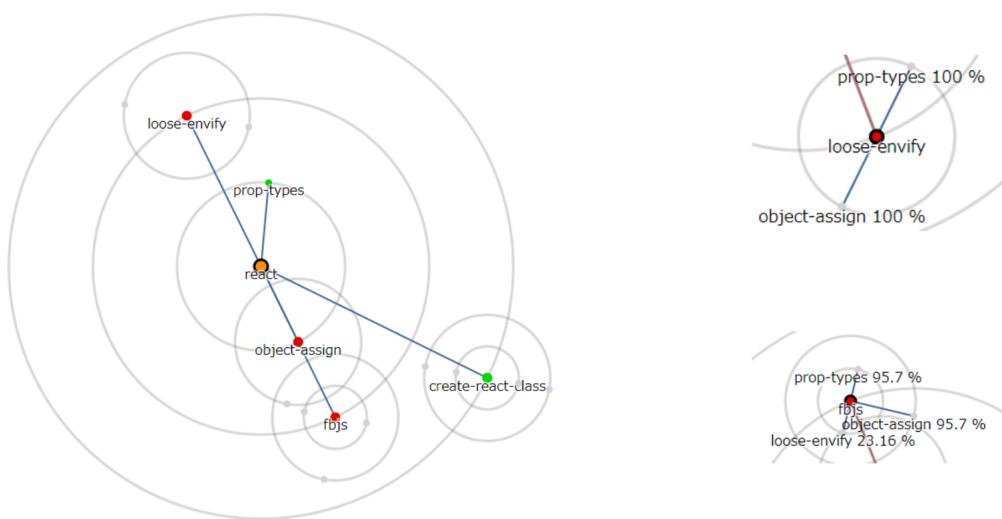


Figure 2.9: React Overview of Sol Mantra [2]

2.3.3 Dataset for Visualizing Node.js Dependency Ecosystem in GitHub

Chinthanet et al. [8] presented an open dataset, GH-node.js, which includes data from the official npm registry and JavaScript applications hosted on GitHub. The snapshots of git repositories exhibits the dependencies between npm packages and their applications on GitHub. To complement the git repositories, the researchers also added meta-data, including vulnerability information and other developers activities such as issues and pull requests.

The structure of a typical Node.js Package includes nine files.

1. package.json: This file is the configuration of the document, and it also contains meta-data relevant to the project, including project dependencies, scripts, and version information.
2. node_modules: All third-party dependencies are stored in node_modules.
3. Authors.md: It contains individuals contributors information.
4. Changelog.md: All changes after each release of the package are kept in this file.
5. Code_of_Conduct.md: This file contains the guidelines when contributors report issues.
6. Contributing.md: This is a guideline of how others may contribute to the package.
7. Source code
8. License.md
9. Readme.md: This file contains the purpose of the package usage, installation instructions, and all related information.

The meta-data structure includes data from GitHub API and vulnerability information from GitHub. The structure consists of seven files.

1. Repositories: storage location for software packages
2. Repositories_Info: Basic information of a git repository

3. Dependencies_History: This file stores version and related information of dependencies which change over time for the package based on Software Universe Graph
4. Issues: Information which keep track of tasks, enhancement, and bugs of project retrieved from GitHub API
5. Pull_requests: Changes pushed by contributor to a branch in a repository on GitHub
6. Contributors: Information about contributor collected from GitHub API
7. Security Advisories: Information is retrieved from the GitHub Security Advisory database

2.4 Chapter Summary

This chapter explains the fundamental knowledge of this project, including third-party vulnerabilities and detection tools. In addition, existing vulnerability analysis tools, which are Dependabot, Snyk, and npm-audit, are described. Lastly, related research studies about lags in dependencies update, existing dependencies visualizations are presented in this chapter. The studies show that most developers are unaware of vulnerabilities in the project due to the indirect adoption of dependency in the chain of dependencies. The existing visualization tools only detect vulnerabilities from a particular project's direct dependencies. A gap of studying the whole npm ecosystem's dependency adoption and vulnerabilities infection still remains. Besides, potential vulnerability risks due to a chain of dependencies that a project might encounter are unidentified. Lastly, the spreading of npm vulnerabilities have not been presented in time series.

CHAPTER 3

ANALYSIS AND DESIGN

Analysis and Design chapter discusses the key concepts used in the project, and illustrates the system design of the project from the overview to the detailed steps in each process.

3.1 Achilles: A tool for npm ecosystem visualization and vulnerability detection

In order to tackle the define problem statements and to fulfill the objectives of the project. We propose an automated tool, called “Achilles”, that can visualize npm ecosystem to vividly show the relationships, i.e., dependencies, among the npm packages and the existing vulnerabilities. The Achilles system aims to raise the developer’s awareness about the potential vulnerability that the project might have when 1) the adopted package is vulnerable or 2) the adopted package adopts a vulnerable package. In this project, we call this situation of indirect adoption of dependencies as “a chain of dependencies”. A chain of dependencies can be of any length. For example, the study by Chinthanet et al. [8] shows that there can be more than 4 level of this chain of dependencies in npm packages.

Our work is inspired by Kula et al.[4]’s study. They conducted a developer survey to investigate to what extent are developers updated their library dependencies, and the result showed that 11 out of 16, which is 69% are unaware of the vulnerability in their software potentially because of the indirect adoption of a vulnerable library via other libraries, i.e., chains of dependencies.

Figure 3.1 shows the concept of the proposed npm ecosystem graph visualization. Each node can represent both package and client (explained next). The starting nodes of the paths are the npm package in the npm registry (P1, P2, P3). Other npm packages or projects that use that package in the first level are called clients (C). The study from Chinthanet et al.[8] shows that the chain of dependency occurs when one package or project adopts another package, as shown in the figure 3.1 where C2 adopts package

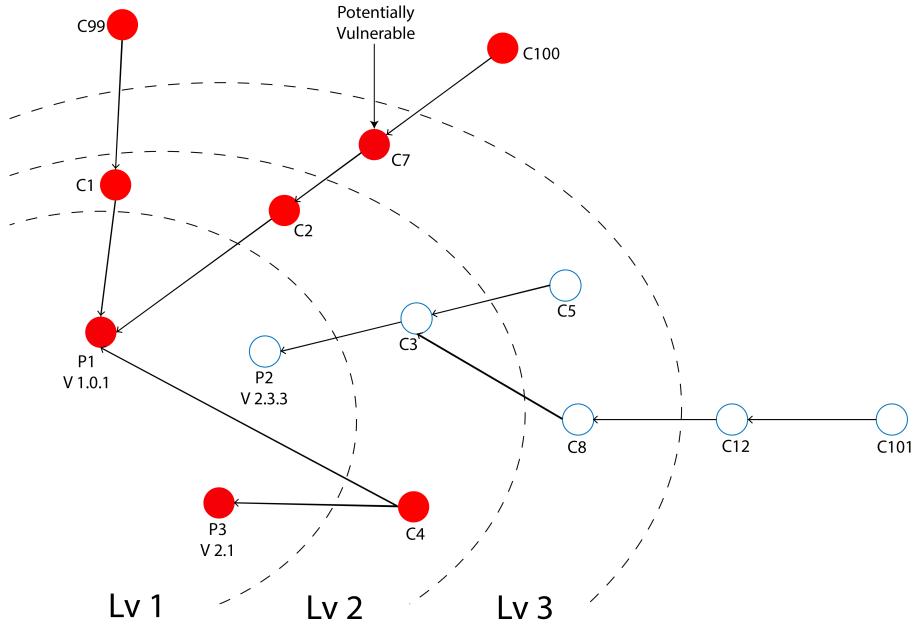


Figure 3.1: Proposed npm Ecosystem Network Visualization

P1 and C7 utilize package in C2. From this example, a chain of dependency includes P1–C2–C7–C100. If P1 is found to be vulnerable (highlighted in red in the Figure), all the packages in the chain are “potentially” vulnerable. The other vulnerable chain of dependencies include P1–C1–C99, P1–C4 and P3–C4. On the other hand, the chains of P2–C3–C5 and P2–C3–C8–C12–C101 are clean. Our tool aims to detect this chain of dependencies and reports to the developers if their project lies in any vulnerable chains. For example, if the developers projects include C99, C100, and C101. Achilles will detect that C99 and C100 are potentially vulnerable and need to be carefully checked or fixed by replacing the vulnerable package with other similar packages. On the other hand, C101 does not need any fix.

3.2 System Architecture Overview

For the front-end, we use the React library for the interface. After user login with GitHub, we use Node.js to query the user’s list of repositories. After users select package.json to analyze, we query vulnerability information using GraphQL and retrieve chain of dependencies from npm registry. Then, we use the D3 library to create a visualization graph, and we keep the history of the report in MongoDB Atlas.

The Achilles system composes of three main parts.

Firstly, Achilles analyzes the relationships between the project and their packages using the data from npm's package.json file, which the tool gets the first level of dependencies. Then, Achilles uses the list of first-level dependencies to send requests to the npm registry to retrieve other levels of dependencies using GraphQL. The chain of dependencies information is used to create a graph using D3.js to exhibit the relationship between packages and projects.

Secondly, the analysis showing the **potential of vulnerability** is performed by identifying the chain that the package is part of and determine whether any node in the chain of dependency is vulnerable. If one of the nodes in the chain has a vulnerability, the packages in that chain are also potentially vulnerable.

Lastly, the Achilles system creates a **report** of the project risks of vulnerabilities based on the package's location in the graph to raise the developer's awareness of the project's security issues from third-party dependencies, as shown in Figure 3.3.

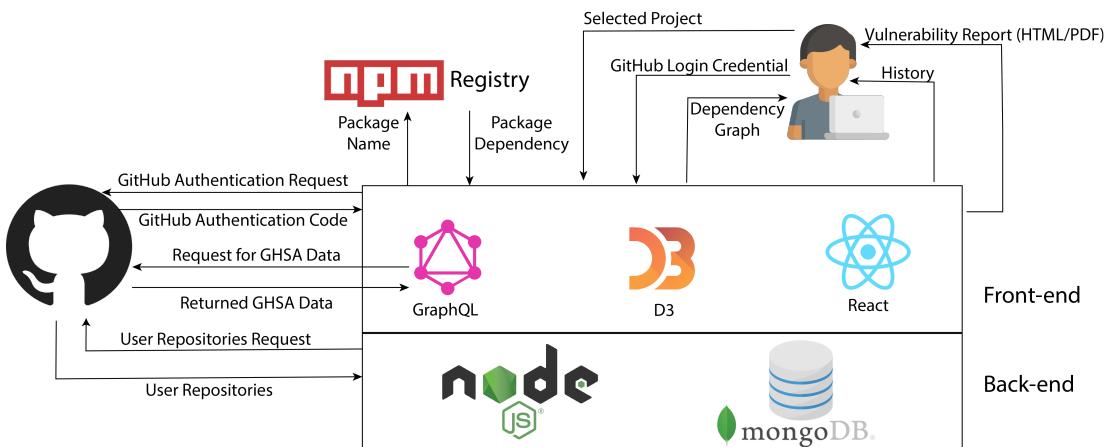


Figure 3.2: System Architecture

In Achilles software system architecture (Figure 3.2), we chose MongoDB Atlas database to store user and report information. We decided to choose this database because the data structure of the report that we need to store in database has various form of data such as vulnerable chaining node and vulnerability information that we get from GitHub Security Advisory which is returning JSON format. Moreover, MongoDB Atlas can store the data that is similar to the objects in the applications which benefits in reducing

time in the need of translating the form of data that is stored in the database and the the form of data that is used in the code.

Users can interact with the Achilles software to see the visualization of the dependency graph via the website, which we developed using React, a front-end framework for developing a website, and D3.js library, which is used to create the visualization. Public GitHub repositories can be retrieved without any authorization. However, users are required to sign in with GitHub in order to allow our system to access their private repositories.

The Achilles software is developed using node.js to query data from MongoDB, get the users' authentication from GitHub, retrieve users' repositories information, and retrieve the package.json file.

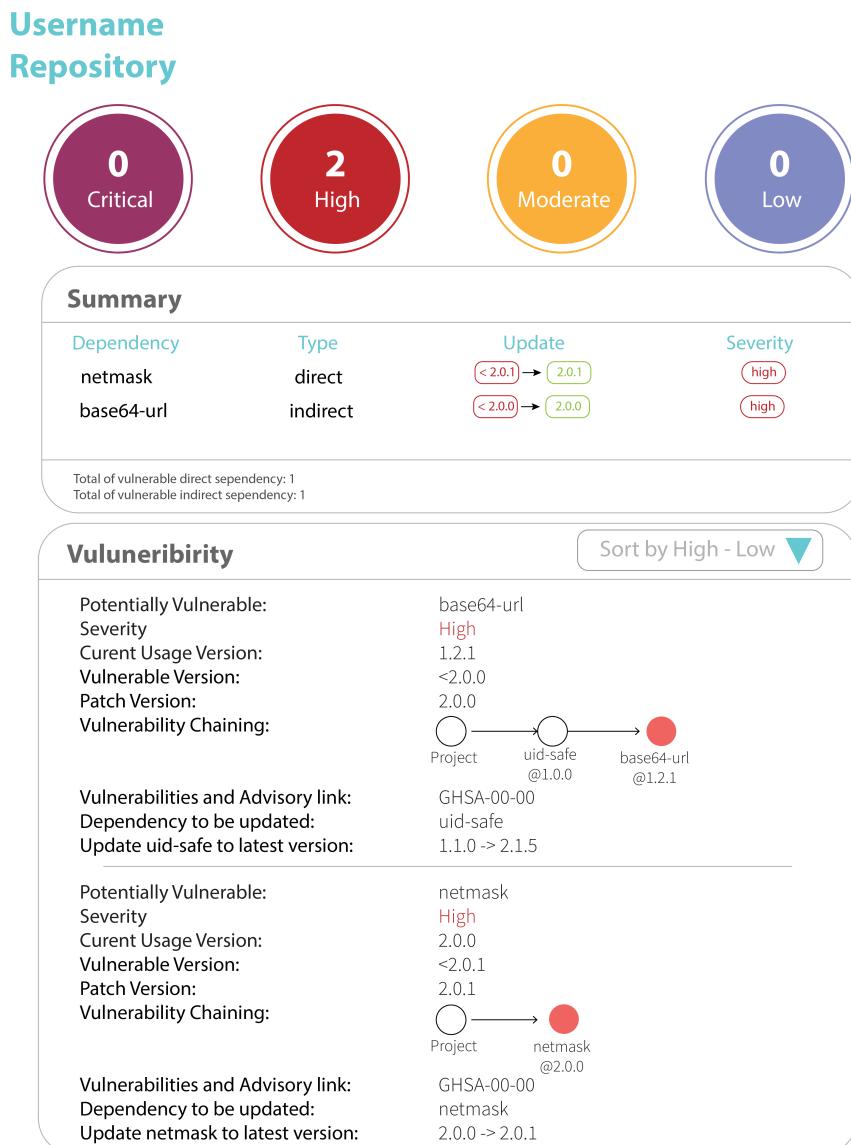


Figure 3.3: A mock-up of the Achilles vulnerabilities analysis report

3.3 Use Case Analysis

In the use case diagram (Figure 3.4), Achilles interacts with an external actor (User) and two entities (GitHub and npm registry). There are seven use cases.

First, the user has to **log in with GitHub** to allow the Achilles System to access the user's private and public repositories. GitHub is the secondary actor which authenticates user credential and permission. Then, GitHub returns the lists of all **user's repositories** (both public and private repositories) for the user to see on a web page. After that, the user can **select only one repository**. In case there are multiple package.json files in that repository, the user must **choose one package.json**, and Achilles System will use that package.json as the system's input. The user can **see the dependency graph** and **see the vulnerable node in the graph**. Besides, the user can **see the tooltip** for more information about the package and the vulnerabilities. The user can also **create a vulnerability report** in which the user can **sort the vulnerabilities by severity** according to their preferences. Furthermore, the user can **download the vulnerability report** in PDF format. Finally, the user can **see the report history**.

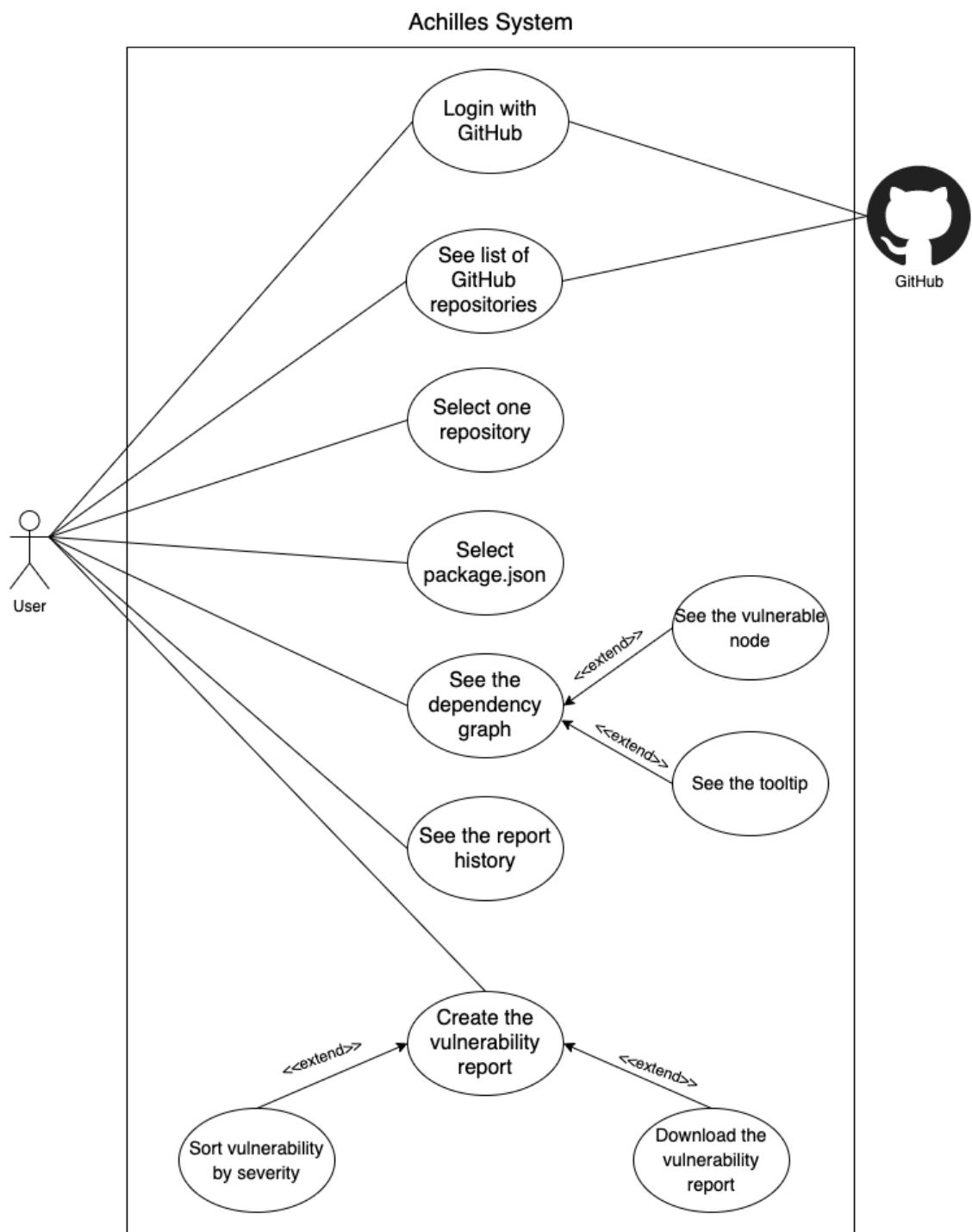


Figure 3.4: Use case Diagram

3.4 Structure Chart

Seven modules in the Achilles system are presented in the structure chart (Figure 3.5). These seven modules include Login with GitHub, See the GitHub repositories list, Select one repository, Select package.json, See the visualization, Create the report and See the report history. The first, fifth, and sixth modules also have submodules.

The first module, **Log in with GitHub**, includes two submodules:

1. **Request GitHub identity**: This submodule is for requesting a GitHub credential to authenticate the user.
2. **Authenticate via GitHub**: This submodule is for authenticating the credentials via GitHub and grant access permission to the public and private repositories.
3. **Grant access permission to repositories**: This submodule is for granting user's permission to access both public and private repositories.

The second module is **See list of GitHub repositories**. This module is for showing the list of private and public of the user' repositories and repositories that user is a contributor.

The third module is **Select one repository**. This module is for the user to select one repository which has package.json file with dependencies to analyze. In a case that the selected repository has no package.json file, Achilles will warn the user that the project cannot be analyzed.

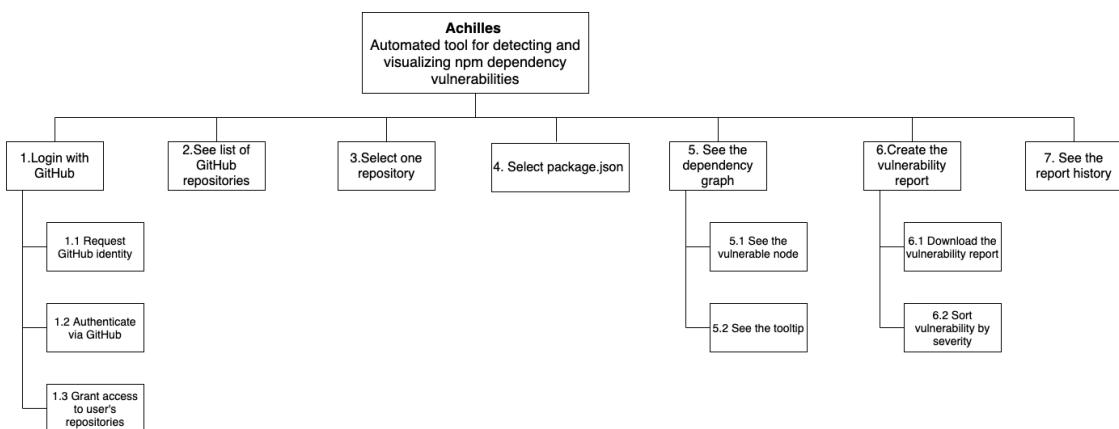


Figure 3.5: Structure Chart

The forth module is **Select package.json file**. In a case that the selected repository has multiple package.json files, Achilles will list all the files for the user to choose only one.

The fifth module, **See the visualization**, are composed of two submodules:

1. **See the vulnerable node**: This submodule is for differentiating the vulnerable node from others by changing the vulnerable node to red.
2. **See the tooltip**: This submodule is for presenting package and vulnerable information for users.

The sixth module, **Create the report**, are composed of two submodules:

1. **Download vulnerable report**: This submodule is for providing the downloadable PDF version of the vulnerability report.
2. **Sort vulnerability by severity**: This submodule is for providing level of severity sorting according to users' preferences.

The seventh module is **See the report history**. The module is for retrieving user information and vulnerability information from internal storage which is MongoDB Atlas to show the history of the vulnerability report which user can revisit.

3.5 System Analysis

System Analysis of the project is represented by data flow diagram level 0 and level 1

3.5.1 Data Flow Diagram Level 0 (Context Diagram)

This figure 3.6 the data flow between the user, Achilles system, GitHub, and npm registry. First, users have to authenticate using GitHub oath in our system. The system requires a GitHub user id and password from users and passes it to GitHub. After authenticating with GitHub, a list of the user's repositories is sent to Achilles to be displayed to the user. Users can then choose the repository that they would like to analyze, and the selected repository name is sent to Achilles, and Achilles will return the list of package.json files in the same project (in case that the project has multiple package.json files)

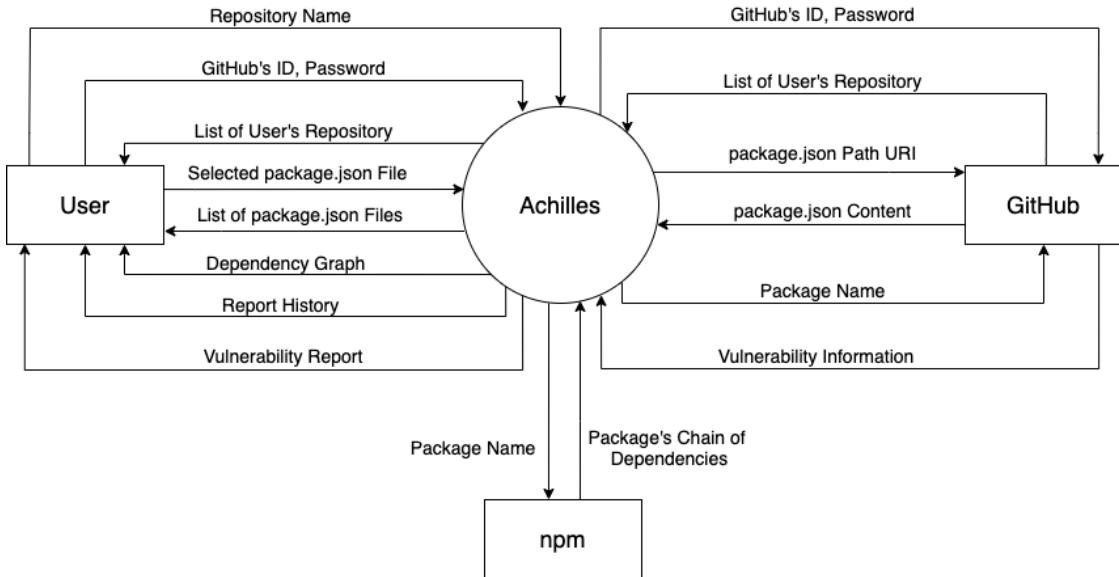


Figure 3.6: Data Flow Diagram Level 0 (Context Diagram)

to the user to choose. Once the user selects a package.json file, the file content will be retrieved from GitHub. The content of that package.json will be used by Achilles to create the graph visualization. Achilles then send the package name to GitHub to request for vulnerability information, and that information will be sent back to Achilles to identify the vulnerable node. Achilles will then find the chain of other packages by sending the package name to the npm registry, which will return the package's chain of dependencies to Achilles to add indirect dependencies to the visualization. A dependency graph will be shown to the user, and the user can create the vulnerability report. The reports' history will be kept by Achilles for a user to revisit.

3.5.2 Data Flow Diagram Level 1

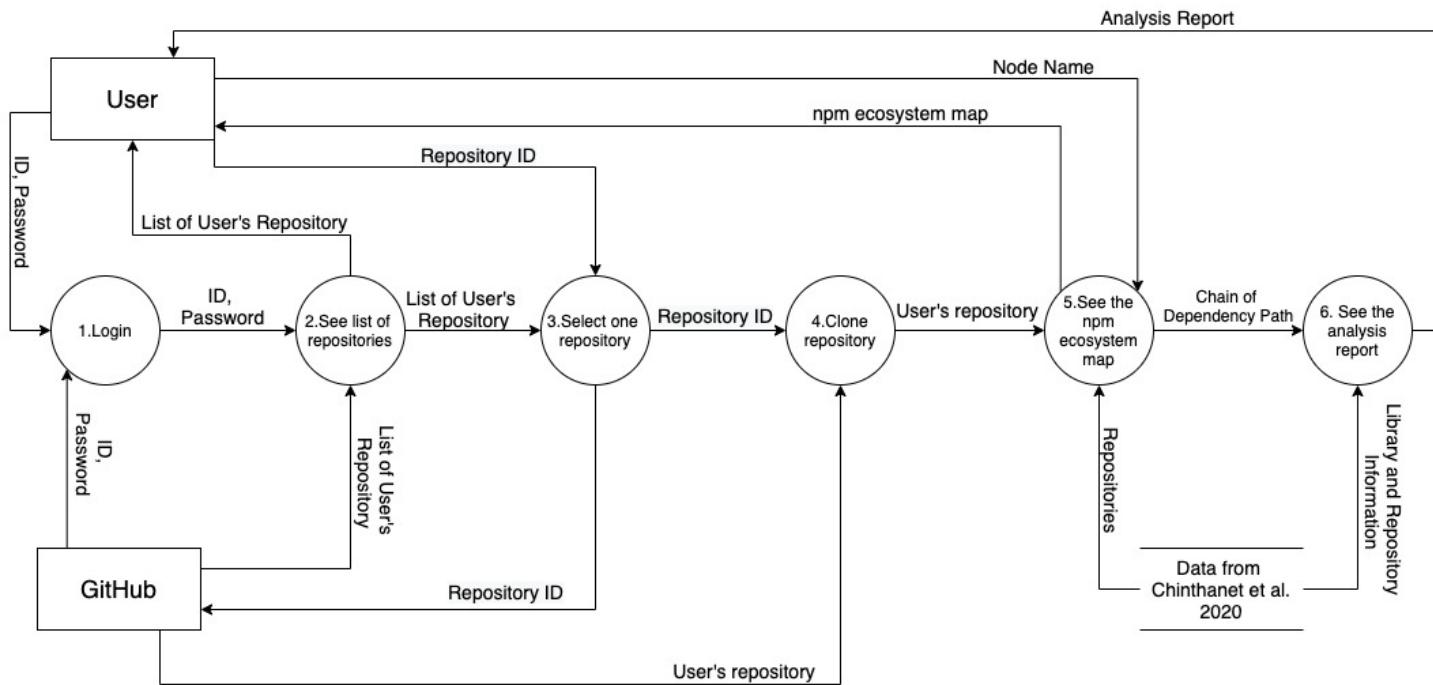


Figure 3.7: Data-flow diagram Level 1

Data Flow Diagram Level 1 (Figure 3.7) shows all the data flow within our system. It consists of seven subprocesses: login, See a list of repositories, Select one repository, Select package.json file, See the dependency graph, See the vulnerability report, and See the report history already been shown in the context diagram. However, there is an internal data store for keeping user data from login (subprocess 1). The user can see the list of repositories and select the package.json to analyze. Achilles will check for vulnerabilities and retrieve the chain of dependencies to create the graph visualization. Users can create the vulnerability report (subprocess 6), which will also be stored in the internal data store and retrieve for the report history in subprocess 7.

3.6 Comparison to Related Work

The Table 3.1 below compares the functionalities of the three tools and techniques for dependency vulnerability detection.

As discussed in the Literature Review, several existing vulnerability detection techniques are available, including Dependabot, Snyk.io, and npm audit. Dependabot is available to be used automatically on GitHub. It only needs to the developer to enable its execution. It can check for vulnerable dependencies in the project repository automatically, and it will generate the pull request to keep the dependencies up to date. Snyk.io is an open-source security management. It can automatically find, prioritize, and fix vulnerabilities in the developers open source repositories. npm audit is the command line that is provided by npm. npm audit provides the summary of vulnerabilities with different severity, suggests an updating package version to a patch version, and gives a short vulnerability information as table. Even though they are capable of report each individual dependency vulnerability in the project, they cannot determine potential risks that the project might be exposed to due to the complexity of indirect adoption of dependency nor show the chain of the dependency.

After developers are aware of vulnerabilities, they have to make a decision whether to update the newer version, which might expose the risk of breakage. Visualization is one of the tools which help the developers to make the decision on the opportunity update. The existing visualization includes VerXCombo and SoL Mantra These visualizations display the relationship for only one particular project but do not consider the whole

Table 3.1: Comparison of Dependencies Vulnerability Detection Technique and Tools

	Dependabot	Snyk	npm-audit
Web service	●	●	
Command line service			●
Find direct dependency	●	●	●
Find indirect dependency			●
Recommend patch version or Receive update Pull Request	●	●	●
Recommend breaking change			●
Update dependencies	●	●	●

npm ecosystem. The risks of vulnerability can be exposed due to the indirect adoption of libraries in the chain of dependencies.

The Achilles system can provide the full complement to these tools and visualizations. The system will visualize the dependency graph from the package.json file and analyze the potential risk of a given npm project according to direct and indirect dependencies that users used in the project.

3.7 Project Timeline, Current Progress, and Future Work

3.7.1 Project Timeline

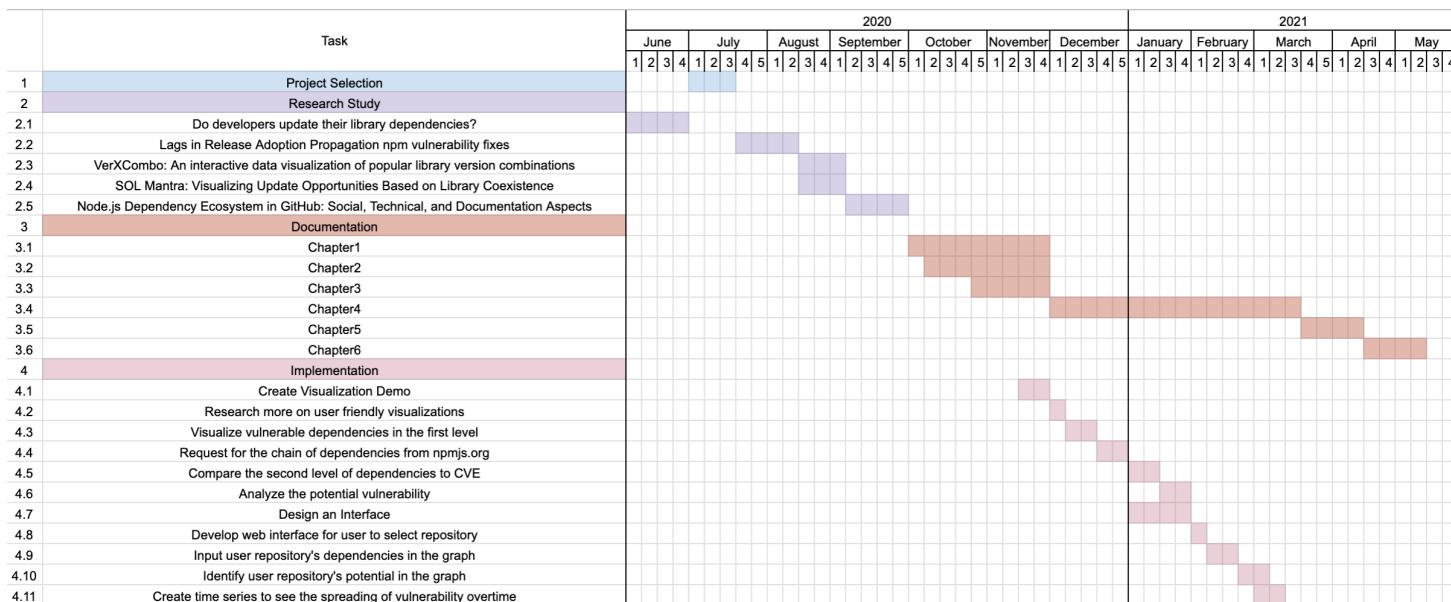


Figure 3.8: Project Timeline

3.8 Chapter Summary

This chapter explains the Achilles system's analysis and design, including an introduction of Achilles, system architecture, a mock-up of vulnerabilities analysis report, use case diagram, structure chart, level 0 and level 1 of data flow diagram, a comparison to related work. Achilles's six main components, including a database for the npm ecosystem, a visualization of vulnerable npm ecosystem graph, a web interface for users to select a repository to analyze potential vulnerable risks from a chain of dependencies, lastly, an analysis report about the project's vulnerable risks. Lastly, a Gantt chart exhibiting the project's time line, current work, and future work are presented in this chapter.

CHAPTER 4

IMPLEMENTATION

In this section, the implementation are divided into 4 parts which are retrieving user's repositories and storing selected repository, visualization, create report, and semver-existing-max

4.1 Retrieve User's Repositories

In order to create the dependency graph visualization, we need to have package.json file of the user's repository. Thus, first and foremost, we require the user to give us a permission to get their GitHub repositories. After the permission is granted via GitHub OAuth, the list of the user's repositories will be shown to the user. When the user selects a repository to analyze, there will be a warning if the repository is not an npm project. However, if the repository has multiple package.json files, Achilles will ask the user to select a package.json file that they want to check for vulnerability.

4.1.1 Get the User's GitHub Repositories

In order to get the user's GitHub repositories, we need to have the GitHub's user access token from GitHub OAuth. The user access token will be provided when the user logs in to Achilles with their GitHub account.

There are two steps to get the user access token.

- 1. Request a user's GitHub identity which is performed in the front-end stack (React)**

```
GET https://github.com/login/oauth/authorize
```

We use this URL with the parameters to get the exchange code for getting access token. The parameters that we use are as follows the table 4.1.

- 2. Users are redirected back to the pre-configured site by GitHub which is performed in the back-end stack (NodeJS)**

Table 4.1: Parameter

Name	Type	Description
client_id	String	The client ID is received from GitHub OAuth application.
redirect_uri	String	The URL that will be redirected to after user's authorization.
scope	String	The scope is a limitation of requesting users's repo scope.

Table 4.2: Parameter

Name	Type	Description
client_id	String	The client ID is received from GitHub OAuth application.
client_secret	String	The client secret is received from GitHub OAuth application.
code	String	The code you received as a response to Step 1.

```
POST https://github.com/login/oauth/access_token
```

After getting the exchange code, we need to use the following API to get the user access token. We use this URL with the parameters to get the access token. The parameters that we use are as follows the table 4.2.

The default response that we get from Github is

```
access_token=<token>&token_type=bearer
```

After getting access token, we will use the below URL to get the user repositories. The GitHub access token is required to use in the header request with the authorization parameter on the table 4.3. Otherwise, users' repositories are not fetchable with the given URL.

```
GET https://api.github.com/user/repos
```

Table 4.3: Parameter

Name	Type	In	Description
accept	string	header	Setting to application/vnd.github.v3+json is recommended.
authorization	String	header	Setting to access token that we received
per_page	integer	query	Result per page (max 100)
page	integer	query	Page number of the results to fetch.

Code example by using axios:

```
const GITHUB_ACCESS_TOKEN = 'e72e16c7e42f292c6912e7710c838347ae178b4';

axios.get(
  'https://api.github.com/user/repos?per_page=100&page=1',
  {
    headers: {
      Authorization: `token ${GITHUB_ACCESS_TOKEN}`,
      Accept: "application/vnd.github.v3+json",
    },
  }
);
```

4.1.2 Filter for npm Projects

When the user select one of these repositories, Achilles uses the API below to check whether this repository is an npm project. This API is searching for the given repository name by checking if the repository has package.json file name and in the package.json file has "dependencies" word.

```
const user = "username";
const repoName = "repository name";

GET `https://api.github.com/search/code?q=user:${user}+
dependencies+repo:${user}/${repoName}+filename:package.json`
```

Code example by using axios:

```
const user = "username";
const repoName = "repository name";

axios.get(
  `https://api.github.com/search/code?q=user:${user}+
  ↪ dependencies+repo:${user}/${repoName}+filename:package.json`,
  {
    headers: {
      Authorization: 'token
        ↪ e72e16c7e42f292c6912e7710c838347ae178b4',
    },
  }
);
```

After verifying that the selected repository is npm project, Achilles will ask users to choose a package.json file if the repository has multiple package.json files. Otherwise, Achilles will use the default path to query the package.json content from GitHub.

API to get package.json content

```
const user = "username";
const repoName = "repository name";
const path = "package.json path";

GET `https://api.github.com/repos/${user}/${repoName}/contents/${path}`
```

Code example by using axios:

```
const user = "username";
const repoName = "repository name";
const path = "package.json path";

axios.get(
  `https://api.github.com/repos/${user}/${repoName}/contents/${path}`,
  {
    headers: {
      Authorization: 'token e72e16c7e42f292c6912e7710c838347ae178b4',
      Accept: "application/vnd.github.VERSION.raw",
    },
  }
);
```

4.1.3 Storing the Selected Repository

Lastly, before going to visualize the dependency graph, Achilles temporary stores package.json content in localStorage, which is provided by the web application for storing client-side data. The localStorage allows a web application to store persistent data and saves the data as key-value pairs in a web browser with no expiration date. However, localStorage does not allow storing data in json format so that we use JSON.stringify() function to convert package.json content to string format.

Code example:

```
const packageJsonContent = {
    ...,
    "dependencies": {
        "dep1": "v1"
    }
}

localStorage.setItem('packageJsonContent', JSON.stringify(packageJsonContent));
```

4.2 Visualizations

The graph visualization composes of three main components which are node, edge and tooltip. To create the force-directed graph which it used to show the relationship between the packages, we utilize the D3.js library.

4.2.1 Generating Nodes and Edges

For generating nodes and edges in the graph, there are 7 main steps for preparing data and creating the visualization which includes setup D3 and arrow head, setup force simulation, retrieving dependencies from package.json, animating the visualization, checking for the vulnerability and fetching chain of dependencies.

We define the mock data which will be used for the project node and create an empty array for the link data. For the node, D3 only requires id property. Other properties are added according to our utilization.

```
const mockNodesData: INode[] = [
{
    id: 'PROJECT',
    name: 'PROJECT',
    type: NODE_TYPE.ROOT,
    status: NODE_STATUS.CLEAN,
    version: '',
    dependenceiesAmount: 0,
    level: 0,
},
];
const mockLinksData: ILink[] = [];
```

1. Setup D3 and arrow head

The first step is to setup D3 for the chart component and set the node and link data.

```
function setupD3() {
  ref.current = d3
    .select(svgRef.current)
    .attr('viewBox', `-${width / 2}, -${height / 2}, ${width}, ${height}`)}
    .style('font', '12px sans-serif');

  ref.current.call(
    zoom.current.on('zoom', (event: any) => {
      ref.current.select('g').attr('transform', event.transform);
    })
  );
}

setNodesData(mockNodesData);
setLinksData(mockLinksData);
```

We also need to setup attributes of the arrow head including the reference position, the size of the arrow, color, etc.

```
function setupMarker() {
  ref.current
    .append('defs')
    .append('marker')
    .attr('id', 'arrow-head')
    .attr('viewBox', '-4 -4 10 10')
    .attr('refX', '-4')
    .attr('refY', '0')
    .attr('orient', 'auto')
    .attr('markerWidth', '10')
    .attr('markerHeight', '10')
    .attr('xoverflow', 'visible')
    .append('svg:path')
    .attr('d', 'M 4,-2 L 0 ,0 L 4,2')
    .attr('fill', '#fff');
}
```

2. Setup force simulation

When there is a change in nodesData and linksData, we have to setup the force simulation of the graph. D3 provides .forcesimulation which has its own algorithm that

will generate position x and y for the node at the certain time.

```
function setupForceSimulation(_nodes: any, _links: any) {
    simulation.current = d3
        .forceSimulation(_nodes)
        .force('charge', d3.forceManyBody().strength(-300))
        .force(
            'link',
            d3.forceLink(_links).id((d: any) => d.id)
        )
        .force('x', d3.forceX())
        .force('y', d3.forceY());
}
```

3. Retrieving dependencies from package.json

Then, we retrieve the direct dependencies data from package.json file and keep them in the array of depData. In the function animate, each dependency is retrieved and add to node as the target node and create the link which connected to the PROJECT node which is the source node. setTimeout() is used to controlled the speed of the adding process.

```
async function animate() {
    const ADD_NODE_SPEED = 50;
    // Gently add node/link
    function gentlyAddNodeLink(dependency: any) {
        return new Promise((resolve) => {
            const node = createNodeData(
                dependency.node.id,
                dependency.node.version,
                dependency.node.level
            );
            setTimeout(() => {
                addNodeLink(dependency.link.id, node); <-- resolve(null);
            }, ADD_NODE_SPEED);
        });
    }
    const _depData = depData;
    const newDepData = _depData.splice(1);
    await gentlyAddNodeLink(_depData[0]);
    setDepData(newDepData);
}
```

4. Animating the visualization

5. Checking for the vulnerability

Next step is to check the vulnerability of the dependencies. We investigate npm audit, and we found that npm audit use npm security advisory database as the source of vulnerabilities. However, we decided to choose GitHub security advisory or GHSA over npm security advisory database because GHSA has a larger number of vulnerabilities in the npm ecosystem than npm security advisory, according to figure 4.1. This is because the GHSA database already includes npm security advisory database and a few additional vulnerability resources such as National Vulnerability Database, Security advisories reported on GitHub, and vulnerabilities reports that come from combination of machine learning and human review on GitHub. Moreover, the GitHub document stated that “If you created a security advisory in your repository, the security advisory will stay in your repository. We publish security advisories for any of the ecosystems supported by the dependency graph to the GitHub Advisory Database on github.com/advisories. If a security advisory is specifically for npm, we also publish the advisory to the npm security advisories. For more information, see npmjs.com/advisories.[16]” This means that some vulnerabilities that are found in the user’s repository will be updated in GHSA first so that we can get the latest vulnerabilities information from GHSA.

A project can publicize security fixes in several places - CVE feed, mailing lists, open-source groups, or within its release notes or changelog. Some vulnerabilities might not be disclosed in the National Vulnerability Database or published in the CVE feed. GitHub creates machine learning models that scan text associated with public commits (the commit message and linked issues or pull requests) to detect these vulnerabilities from activity within the GitHub developer community and generate security alerts. While npm security advisory only depends on the security advisories already published in the National Vulnerability Database. This results in different numbers of security advisories in GHSA and npm security advisories. [17]

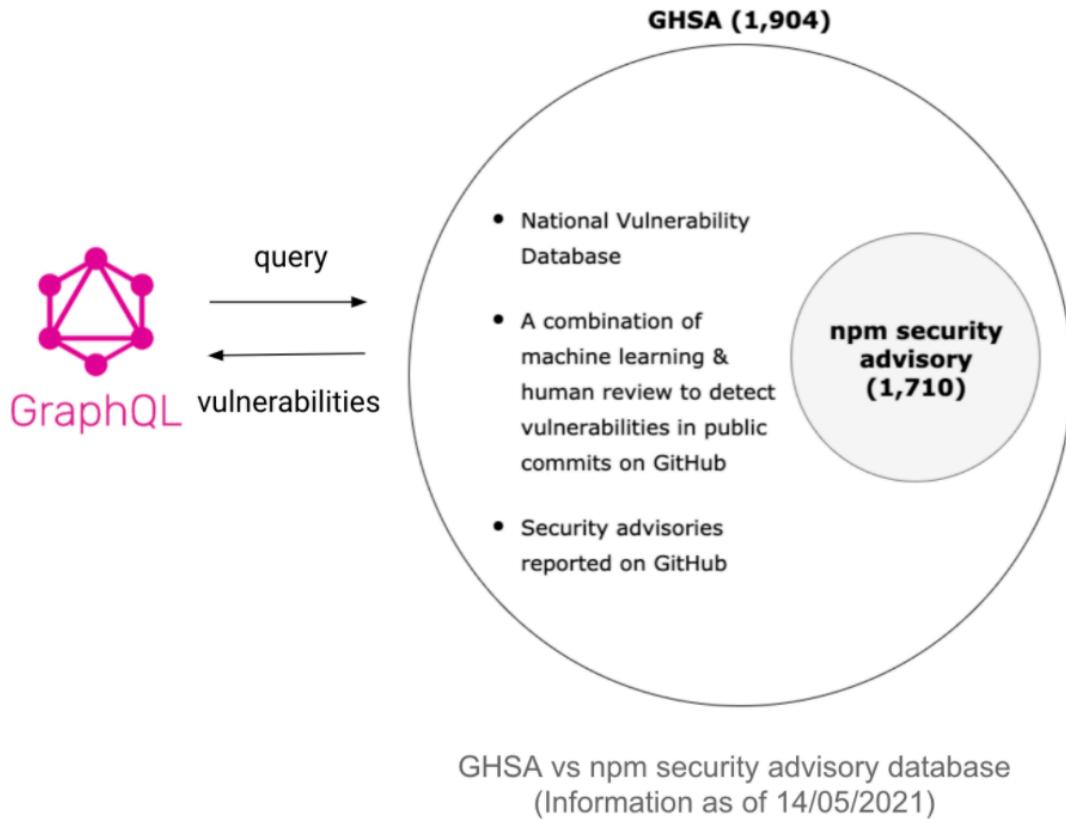


Figure 4.1: GHSA vs npm security advisory database

We loop through each node in the current level and query for the vulnerability using graphQL api and save them to result.

```
export async function queryVulnerability(packageName: string) {
  const vulnerability = await client.query({
    query: gql` 
      query {
        securityVulnerabilities(first: 100, ecosystem: NPM, package:
          ↗ "${packageName}") {
          totalCount
          nodes {
            package {
              name
            }
            firstPatchedVersion {
              identifier
            }
            severity
            vulnerableVersionRange
            advisory {

```

```
        identifiers {
          type
          value
        }
        permalink
      }
    }
  }
`,
});
```

After getting vulnerabilities information, we filter only the related vulnerabilities by checking whether the node version and vulnerable version is intersected. Then, we update the node status whether it is vulnerable or not, and change the node color to red if that node is vulnerable.

```
export const filterRelatedVulnerabilities = (node: INode, vulnerabilities: ISecurityVulnerability[]) => {
  let relatedAdvisories: ISecurityVulnerability[] = []

  relatedAdvisories = vulnerabilities.filter((vulnerability) => {
    if (node.version !== 'latest') {
      const versionRange = vulnerability.vulnerableVersionRange &&
        vulnerability.vulnerableVersionRange.toString().replace(',', '')

      if (semver.intersects(node.version, versionRange)) {
        if (vulnerability.firstPatchedVersion) {
          if (semver.intersects(node.version,
            vulnerability.firstPatchedVersion.identifier)) return false;
        }
        return true;
      }
    }
    return false
  })

  return relatedAdvisories
}
```

6. Fetching chain of dependencies

Finally, we retrieve the chain of dependencies by sending the name of dependencies via REST API to npm registry. The processes are repeated until the dependencies reached the forth level.

```
import axios from 'axios'

const GET_DEPENDENCIES = 'https://registry.npmjs.cf/'

export const getDependencies = async (packageName: string) => {
  const result = await axios({
    url: GET_DEPENDENCIES + packageName,
  })

  return result
}
```

4.2.2 Generating Tooltips

The third component of the visualization is the tooltip. We retrieve information of the node and advisory and show them when the mouse is hover over the node.

```
function onMouseOverNode(event: any, d: any) {
    const svgStyle = svgRef.current!.style;
    const topAwayOffset = 10;
    const leftAwayOffset = 10;
    if (document !== null) {
        const tooltipEl = document.querySelector('#tooltip')! as HTMLDivElement;
        if (tooltipEl) {
            tooltipEl.style.left = `${event.offsetX - svgStyle.width / 2 + topAwayOffset}px`;
            tooltipEl.style.top = `${event.offsetY - svgStyle.height / 2 + leftAwayOffset}px`;
            setTooltipVisibility(true);
        }
        if (advisoriesData[d.id]) {
            setTooltipData({
                node: d,
                advisory: filterRelatedVulnerabilities(d, advisoriesData[d.id]),
            });
        } else {
            setTooltipData({
                node: d,
                advisory: [],
            });
        }
    }
}
```

4.3 Create Report

The vulnerability report is created by the user after seeing the dependency graph. Storing report aims to help users keep track of the vulnerability information in their project. For example, if the users' project had vulnerabilities and they had fixed the vulnerabilities, they can come back to see the report history later in Achilles. Moreover, storing the vulnerability report intents to help users see the vulnerability information without visualizing the project again since visualizing the dependency graph might take

time to complete. The data in the report are stored in MongoDB atlas in JSON format.

When the user click "Create Report" button, Achilles will collect all the vulnerability information that are displayed in the dependency graph including the chain of dependencies that has vulnerable nodes, package.json path, repository name, username, patch version of vulnerable nodes, GHSA, CWE, and CVE. The CVE and CWE are the vulnerability records that are considered standards of information security community. Both sources are different in term of security practitioners.

The CVE (Common Vulnerabilities and Exposures) is a public resource that reports the information security and exposures. The CVE information can assist developers to search the attack signatures and identify particular vulnerability exploits. On the other hand, the CWE stands for Common Weakness Enumeration, which is a formal list of common software weaknesses. The CWE is the common software weaknesses that caused by software architecture, design, code, and implementation. In short, CVE is a problem that developers have to deal with the specific instance in a system while CWE is a problem that developers have to deal with vulnerability, not the instance within a system. Nevertheless, CWE information is the information that we did not get from visualization part. This is because security vulnerability API in visualization does not provide CWE information. Thus, Achilles uses another API to request the CWE information when creating report. We did not query the CWE information in the visualization simultaneously since it might affect time usage in creating the dependency graph.

Querying CWE Information

An example code snippet that uses GraphQL to query CWEs information

```
import { ApolloClient, InMemoryCache, createHttpLink, ApolloLink } from
  'apollo/client'
import { setContext } from 'apollo/client/link/context'

const GITHUB_ACCESS_TOKEN = <github access token>

const ENDPOINT: ApolloLink = createHttpLink({
  uri: 'https://api.github.com/graphql',
})

const authLink = setContext(() => {
  return {
    headers: {
      authorization: `Bearer ${GITHUB_ACCESS_TOKEN}`,
    }
  }
})

const client = new ApolloClient({
  link: authLink.concat(ENDPOINT),
  cache: new InMemoryCache()
})

const ghsaId = "gshaId from Security Github Advisory";
const queryCwes = async (ghsaId: string) => {
  const securityAdvisory = await client.query({
    query: gql`  

      query {
        securityAdvisory(ghsaId: "${ghsaId}") {
          cwes(first: 10) {
            nodes {
              cweId
              name
            }
          }
        }
      }
    `,
  });
}
```

In CWEs information, we found that there is no link to view more information for specific CWE so that we decide to attach "cweId" to the URL in order to link to the CWE information website.

Example:

```
const CWE_ID = "cwe id that getting from query"
const CWE_URL_LINK = `https://cwe.mitre.org/data/definitions/${CWE_ID}.html`
```

4.3.1 Vulnerability Information Data Template

The below is the data template that Achilles uses to store the vulnerability information.

```
interface IReport {
    username: string; // user Github name
    jsonPath: string; // package.json path
    repository_name: string; // repo's visualization name
    items: IItem[];
}

interface IItem {
    chaining: IChaining[];
    cwes: ICWE[];
    direct_dependency_name: string;
    direct_dependency: {
        name: string;
        current_version: string;
        latest_version: string;
    };
    package: {
        name: string;
    };
    firstPatchedVersion: {
        identifier: string; // version: e.g. 1.9.0
    };
    vulnerableVersionRange: string; // version rager: e.g. < 1.9.0,
    severity: ADVISORY_SERVERITY_LEVEL;
    advisory: {
        permalink: string;
        ghsaId?: string;
        identifiers?: IAdvisoryIdentifier[];
    };
}
```

```

interface IChaining {
  source: string;
  target: string;
}

interface ICWE {
  cweId: string;
  name: string;
  link: string;
}

type ADVISORY_SERVERITY_LEVEL = 'LOW' | 'MODERATE' | 'HIGH' | 'CRITICAL';

interface IAdvisoryIdentifier {
  type: string;
  value: string;
}

```

Storing Vulnerability Report in MongoDB Atlas

After collecting the vulnerability information from dependency graph, we use the below API to send the vulnerability report to store in MongoDB Atlas.

```

const HOST_DOMAIN = 'Input your host domain';

POST `${HOST_DOMAIN}/api/v1/reports`;

```

A code snippet by using axios to store the vulnerability report is shown below.

```

const HOST_DOMAIN = 'Input your host domain';
const report = {};// Input report information by following the report format
→ in step 1
const JWT_TOKEN = 'jwt token that created by backend';

axios.post(` ${HOST_DOMAIN}/api/v1/reports`,
  { report },
  {
    Authorization: `Bearer ${JWT_TOKEN}`,
  }
);

```

The success response from this API will be a report ID that stored in database.

Achilles uses the report ID from the above API to request a specific report in order to render the vulnerability report in the report web page.

```
const HOST_DOMAIN = 'Input your host domain';

GET `${HOST_DOMAIN}/api/v1/reports/:reportId`;
```

A code snippet by using axios to get the vulnerability report by report ID which was generated by MongoDB Atlas is shown below.

```
const HOST_DOMAIN = 'Input your host domain';
const report = {};
const JWT_TOKEN = 'jwt token that created by backend';
const reportId = "report id";

axios.get(`${HOST_DOMAIN}/api/v1/reports${reportId}`,
{
  Authorization: `Bearer ${JWT_TOKEN}`,
}
);
```

Retrieving Report History

Finally, Achilles allows user to see the all the report history that users has created by the following API.

```
const HOST_DOMAIN = 'Input your host domain';

GET `${HOST_DOMAIN}/api/v1/reports`;
```

A code snippet by using axios to get the list of report history is shown below.

```
const HOST_DOMAIN = 'Input your host domain';
const JWT_TOKEN = 'jwt token that created by backend';

axios.get(`${HOST_DOMAIN}/api/v1/reports`,
{
  Authorization: `Bearer ${JWT_TOKEN}`,
}
);
```

4.4 Semver-Existing-Max

semver-existing-max is a npm package that finds the maximum version of given version range that exists in other npm packages or version list. It was created by our team

in order to assist Achilles website in creating dependency graph. In order to create the dependency graph, there was a step that requires to find the chain of dependencies. For example, package A has a chain of dependencies which are B and C (figure 4.2). This worked fine if the developers installed the exact version of package in the package.json file.

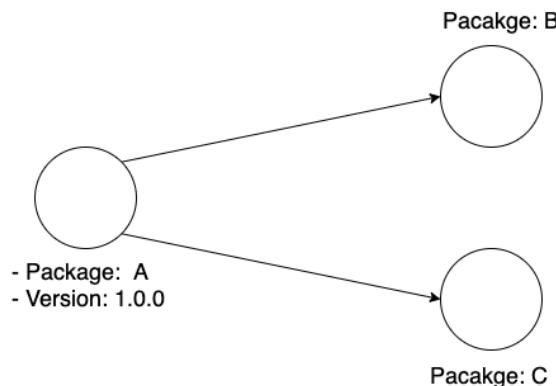


Figure 4.2: Chain of Dependencies of Package A

However, there was a problem when developers installed the package with range version. For instance, developers installed package A with the version 1.0.0 or $\wedge 1.0.0$. The problem with the range version is that we cannot find the the chain of dependencies with the exact version as we did above. This is because 1.0.0 can expand to be 1.0.1, 1.0.2, or 1.0.3 until reaching the maximum of minor change in version 1 (not the maximum version of the package). Moreover, as the observation from our team shows that different versions of package can lead to a different chain of dependencies in the package.

As shown in the Figure 4.3, package A with version 1.0.0 has B and C as a chain of dependencies whereas package A with version 1.0.1 has the B, C, and D as a chain of dependencies.

At first, our team decided to use minimum version of the package since we found that **semver** package provides a **minVersion** function to find the minimum version of the given range. Yet, after exploring the chain of dependencies and vulnerabilities that found in other tools which are **npm audit** and **dependabot**, we recognized that they both used the maximum version of the given range of the package. In addition, as the npm installation with the existing package range version, npm usually installs the latest

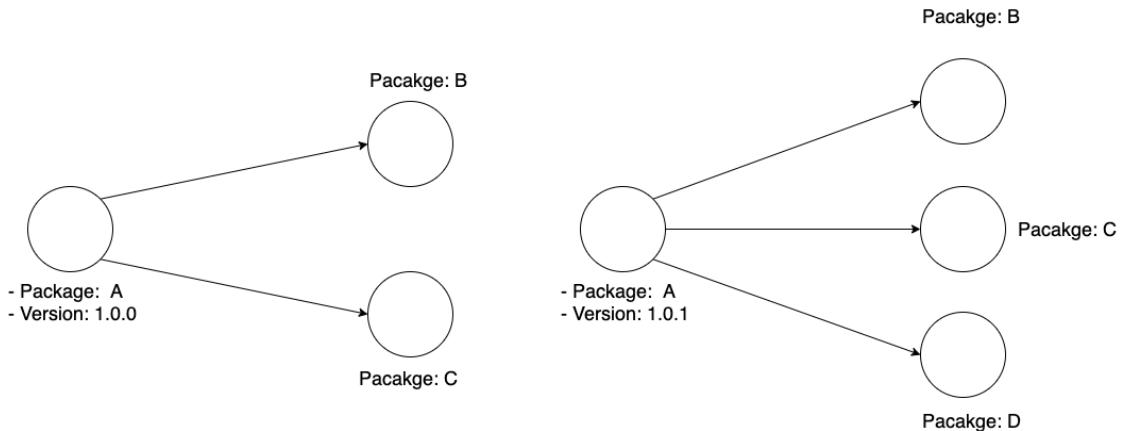


Figure 4.3: Chain of Dependencies of Package A with Different Version

version of the given range in the modules of the project. This behavior somehow can fix the vulnerability in the user's project, even though it is a minor change in version. Then, our team decided to change the methodology in finding chain dependencies by creating semver-existing-max package to help us in finding the maximum version of the given range.

There are three steps to build semver-existing-max package.

1. Get all the versions of the specific package by npm registry API.

```

const package = "package name";
const response = await axios.get(url:
  ↪ `https://registry.npmjs.cf/${package}`);
const versionList = Object.keys(response.data.versions);

// Example result:
// => versionList = [1.0.0, 1.0.1, 1.0.2, 2.1.1, 2.1.4];
  
```

2. Find the intersection version list from the given version range.

```

const semver = require('semver');
const givenRange = '^1.0.0';

const intersectedVersion = versionList.filter(version => {
  if (semver.intersects(version, givenRange)) return true;
  return false;
});

// => intersectedVersion = [1.0.0, 1.0.1, 1.0.2];
  
```

3. Find the intersection version list from the given version range.

```
const semverMax = require('semver-max');
const maxVersion = intersectedVersion.reduce(semverMax);

// => maxVersion = 1.0.2
```

After building this semver-existing-max package, we also published this package to npm registry on 31st March 2021, so that we can install and use this package in Achilles website. Currently, the package has a total of 65 downloads.

The semver-existing-max package can be found here: <https://www.npmjs.com/package/semver-existing-max>. The source code of the package can be found here: <https://github.com/KlintonICT/semver-existing-max>.

CHAPTER 5

EVALUATION RESULTS

In this chapter, we discuss the results of user study to evaluate the performance of the Achilles detecting and visualizing npm dependency vulnerabilities tool.

5.1 Evaluation Methodology

This chapter will focus on the evaluation of the project in several aspects. First, we performed an online survey to understand the awareness of the npm developers on security vulnerabilities, direct/indirect dependencies, and how they prioritize the updates of vulnerable npm packages. Second, we evaluate Achilles tool by performing a user study and compare it to the state-of-the-art tool, which is npm audit. Third, we applied Achilles to real-world GitHub projects in order to check its effectiveness in locating npm security vulnerabilities. We explain each of them in detail below.

5.1.1 The Online Survey

The online survey is created to evaluate developers and students' awareness and perception on security vulnerabilities and gather feedback for the visualization and vulnerability report of Achilles. We recruit participants who have experience in programming for more than six months. We contacted them directly via chat messages, and also posted the survey on the social media. We received nineteen responses; thirteen of them were students in ICT faculty and six were developers. The result of the online survey will be discussed in Section 5.2

5.1.2 The User Study

The objectives of the user study is to

1. Investigate how graph visualization (Achilles) support developer's decision on prioritizing vulnerability to fix.
2. Investigate how different types of visualization (Graph and Table) effect devel-

oper's decision on prioritizing vulnerability to fix.

The user study follows the guidelines from Ko et al.[18]. The guideline consists of ten key steps including Recruitment, Selection, Consent, Procedure, Demographic measurements, Group assignment, Training, Tasks, Outcome measurements, and Debrief and compensate. We followed the guideline as shown below.

1. **Recruitment** - We sent email to potential participants who are developers and we recruit students by contacting the instructors of the related faculty and directly contact potential participants.
2. **Selection** - The inclusion criteria for selecting the participants of this experiment is as followed:
 - Participants must be students in Information and Communication Technology program or full-time employees at a software development company.
 - Participants must have experience using npm to install packages in the projects but do not require to have an experience in using npm audit.
 - Participants must have the aforementioned experience at least six months. Due to the limitation of recruiting ideal participants, we anticipate that it would take six months for students and developers to learn new programming languages and have experience using third-party libraries. In addition, six months is used as the criteria in the previous user study in Gopstein, D. et al. [19]
 - Participants should understand and have fundamental knowledge in software vulnerability.
3. **Consent** - The user study has been approved by the Mahidol Central Institutional Review Board (IRB) - 12 March 2021 Number 084.1502
4. **Procedure** - Fig. 5.1 shows the flowchart of user study procedure between the controlled group and experimental group.
 - We let the participants read the consent form and ask for their confirmation to join the experiment.

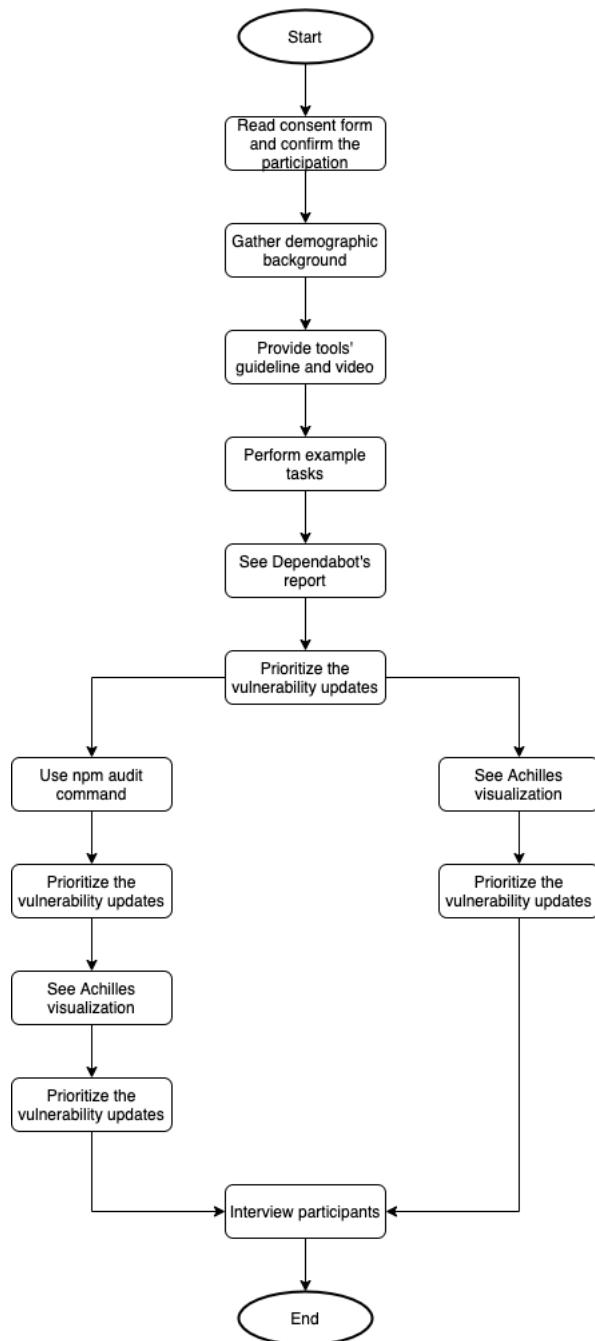


Figure 5.1: Procedures of User Study

- We gather their demographic background.
- We provide the guidelines and videos of the tools demonstration and ask them to perform example tasks.
- When the real experiment begin, both group of participants are asked to see the security vulnerability report from dependabot and prioritize the updates of the vulnerabilities. Then for the controlled group, they are asked to use npm audit, and for the experimental group, they are asked to use Achilles to find security vulnerability. After that they are asked to prioritize the updates of the vulnerabilities again.
- We interviewed the participants on their criteria that they use for the prioritization.

5. Demographic measurements - We asked participants the following questions prior the experiment in order to gain more understanding of participants' background.

- How long have you been using npm?
- What do you use npm for?
- How often do you check security vulnerabilities in the project?
- Do you know indirect dependencies?

6. Group assignment - By following the between-subject experiment, we randomly assign participants to two experimental groups.

- Controlled group: Ten participants are randomly assigned to the controlled group. They used npm audit to analyze security vulnerabilities.
- Experimental group: Ten participants are randomly assigned to the experiment group. They used Achilles to analyze security vulnerabilities.

7. Training - We provided the tools guidelines and videos for the tools demonstration. We also prepare example tasks to check their understanding of the tools before we begin the experiments.

Table 5.1: Characteristics of vulnerabilities in Test 1

		Severity	Complexity
1	three	high	no
2	type-graphql	low	yes
3	xmldom	low	no
4	Pug	high	yes

8. **Tasks** - In this experiment, the participants are asked to use two tools for security vulnerability check in npm projects. The goal that they have to do is prioritizing the updates of the vulnerabilities after using each tool.

To assess Achilles's usability in practice, we used Achilles to perform the vulnerability analysis on real-world GitHub projects and compare the analysis result with Dependabot and npm audit tools to see whether the results are different. The results showed that the vulnerabilities that were found in Achilles were also found in the Dependabot and npm audit tools. However, we spotted that there is a small different behavior between Achilles and npm audit. Some vulnerabilities that were detected in the npm audit were not found in Achilles. This is because npm audit is not just only finding vulnerabilities for dependencies that the project is using, but it also detects the vulnerabilities in **devDependencies** in the package.json file. The dependencies that are installed in devDependencies in the package.json file are the dependencies that are used in local development and testing. It provided some facilitating things in the development process. Nevertheless, those dependencies will not be used or affect the production stage once the project deploys. Since devDependencies are important during the development process, Achilles may be considered to detect vulnerabilities in devDependencies in the future work.

For the experimental group, they will be asked to see the vulnerability report of dependabot in GitHub first and then Achilles. For the controlled group, they will be asked to see the vulnerability report of dependabot in GitHub first and then npm audit. Additionally, we ask the controlled group to see Achilles visualization after they have finished the npm audit task. There are two tasks for the participants to perform.

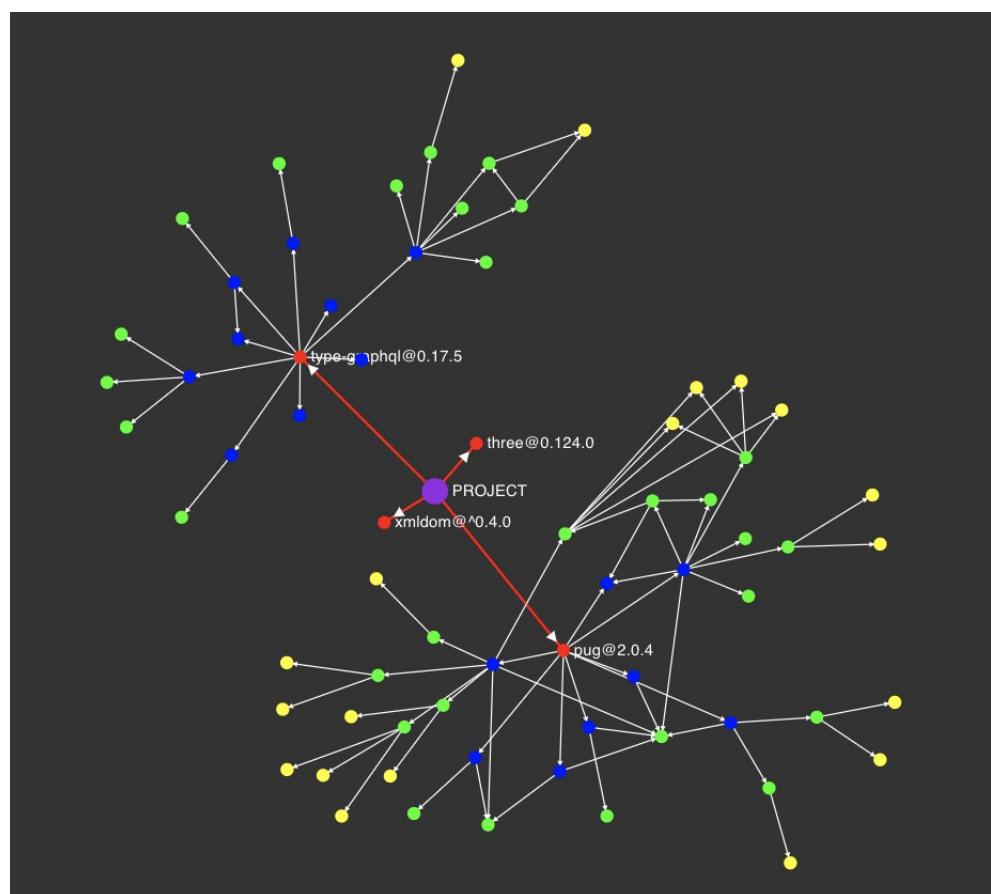


Figure 5.2: Achilles graph visualization of Test 1

Table 5.2: Characteristics of vulnerabilities in Test 2

		Severity	Types
1	Minimist@1.2.0 (karma-mocha@2.0.1)	low	indirect
2	netmask	High	direct
3	angular-expressions	low	direct
4	base64(uid-safe@2.1.5)	high	indirect

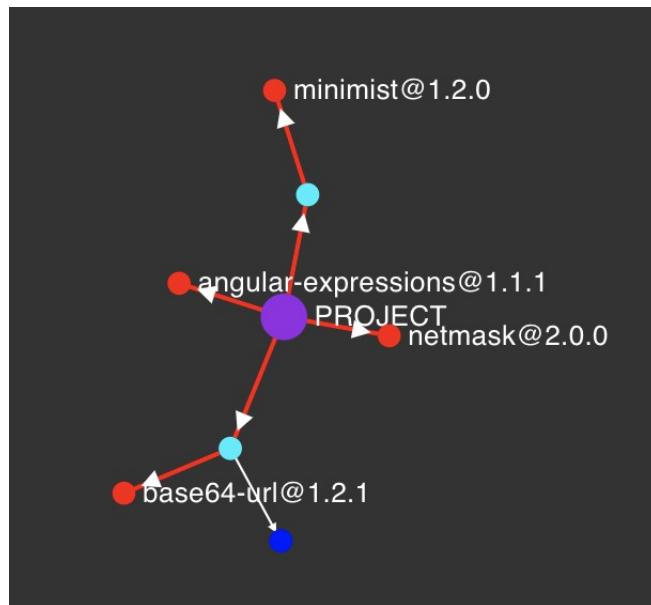


Figure 5.3: Achilles graph visualization of Test 2

Table 5.1 and Figure 5.2 shows the characteristics of vulnerabilities in Test 1. We would like to test whether the graph which shows the complexity of the vulnerability would affect the participants' decision on ranking.

Table 5.2 and Figure 5.3 shows the characteristics of vulnerabilities in Test 2. We would like to test whether the graph which shows the types of the vulnerabilities (direct/ indirect) would affect the participants' decision on ranking.

9. **Outcome measurements** - We compared the prioritization results and criteria that they use before (security vulnerability report from dependabot) and after they see the visualizations (npm audit or Achilles). We also compared the results between different types of visualization which are table by npm audit and graph by Achilles.
10. **Debrief** - We told the participants about the purpose of this experiment and interview them for them about the criteria that they used and the tool feedback.

5.2 Online Survey Result

We conduct an online survey as part of the evaluation process. This survey aims to gather information on developers' and students' awareness and perception of security vulnerabilities and their feedback on the visualization and vulnerability report.

We received the survey response from nineteen participants. There are six developers and thirteen students (Figure 5.4). Since these two participant groups may have different level of skills and experiences, we report their results separately.

5.2.1 Level of Concern Regarding Security Vulnerability Sources

First, we ask their level of concern regarding security vulnerability sources, i.e., vulnerabilities from third-party dependencies and indirect vulnerabilities.

From Figure 5.5, the graph shows the students' level of concern regarding different sources of vulnerability. Most students are very concerned about vulnerabilities from indirect dependencies. However, their concern regarding vulnerabilities from third-party dependencies is almost equal in different levels, from slightly concerned to extremely concerned. Even though two students are not worried about indirect vulnerabilities, all of them are worried about vulnerabilities from third-party dependencies.

From Figure 5.6, There is no developers who are not concerned about security vulnerabilities from third-party dependencies and indirect dependencies. There are 3, 2, and 1 developers who are moderately concerned, very concerned, and extremely concerned about vulnerabilities in third-party dependencies. Regarding indirect vulnerabilities, the result follows the same trend. The largest number of developers falls into slightly concerned (3), followed by moderately concerned, very concerned, and extremely concerned (1, 1, and 1).

We can see that the results from the two groups show that both students and developers are concerned about vulnerabilities from third-party dependencies. However, the student group is concerned more with indirect vulnerabilities than the developer group.

5.2.2 Prioritization Factors for Vulnerabilities Updates

Second, we ask their prioritization factors for vulnerabilities updates. There are five factors: number of vulnerabilities, severity, relevancy to business requirement, the

Occupation

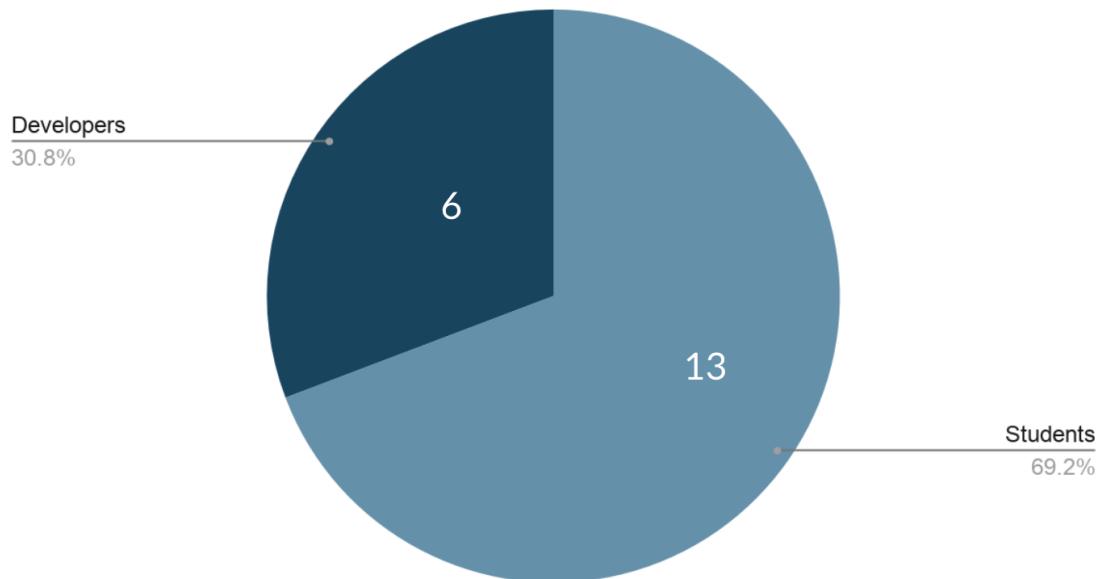


Figure 5.4: Participants' Survey

Students

Level of concern regarding security vulnerability sources

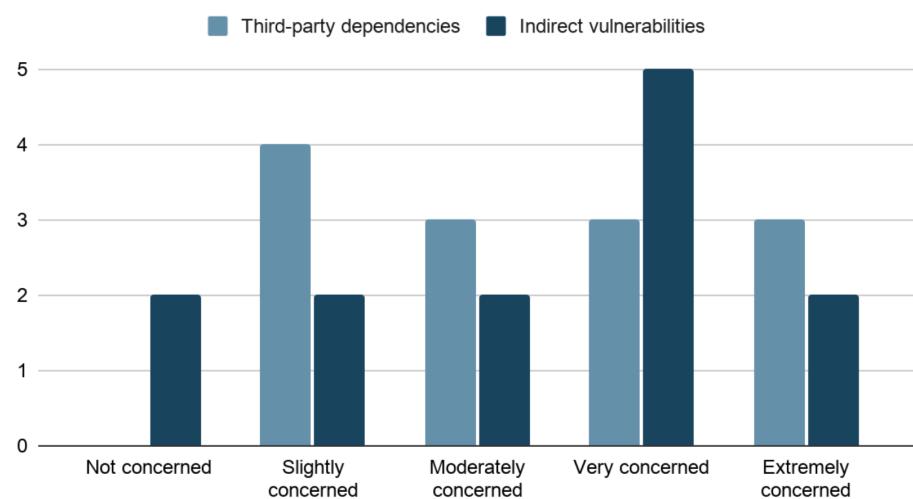


Figure 5.5: The Students' Level of Concern Regarding Different Sources

Developers

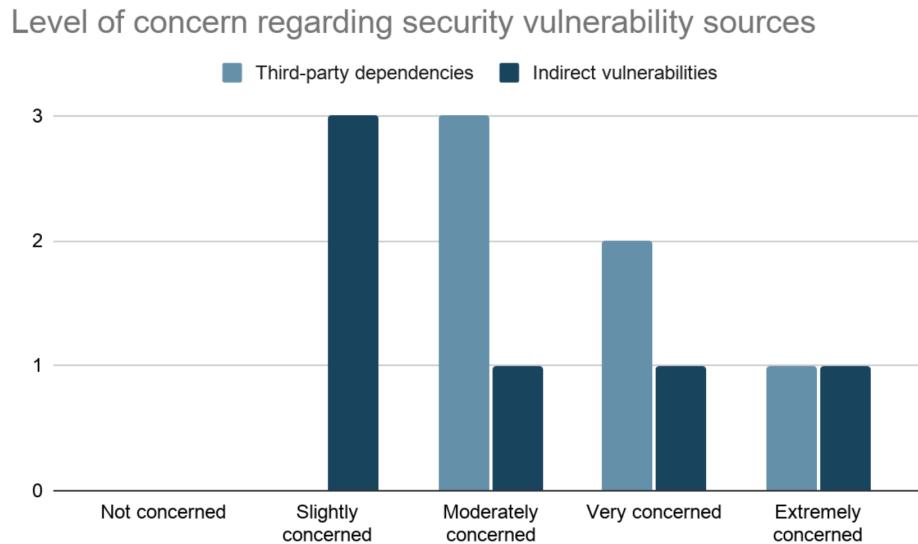


Figure 5.6: The Developers' Level of Concern Regarding Different Sources

gap between vulnerable version and patch version, and recency of the vulnerability. We ask how important each factor to consider when they decide to update vulnerabilities.

From Figure 5.7, seven out of thirteen students think that severity is extremely important and relevancy of that dependencies to the business requirement is very important for prioritizing vulnerabilities to update. These two factors range from moderately to extremely important, while other factors are scattered across the scale. Six students think that recency of vulnerability is moderately important.

From Figure 5.8, six developers think that severity, relevancy to business requirements, and recency of vulnerability are very important factors to consider when deciding which vulnerabilities to be updated. Interestingly, 3 participants rank severity, relevancy of business requirements, and recency of vulnerability equally as very important. Severity seems to be the major factor in this group as it is selected as very important and extremely important from 5 out of the 6 participants.

Even though both students and developers prioritize severity as the extremely level factor, their prioritization is different in other important levels. Students are more interested in relevancy to business requirements at the very important level, while developers emphasize severity, relevancy to business requirements, and recency of vulnerabil-

Students

Library update prioritization factors

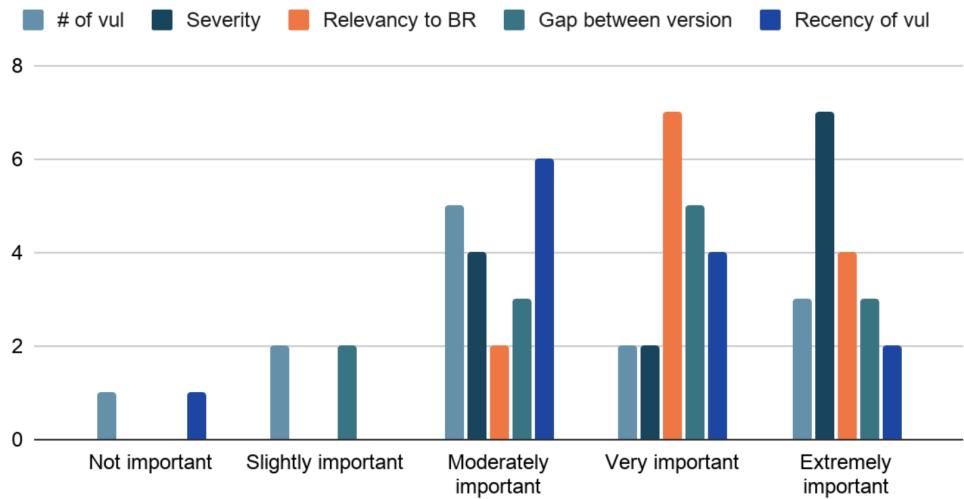


Figure 5.7: The Students' Library Update Prioritization Factors

Developers

Library update prioritization factors

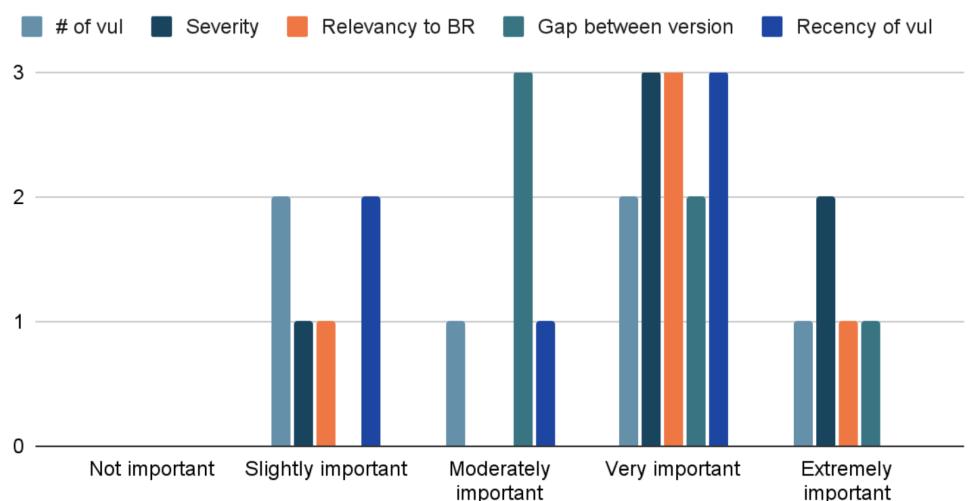


Figure 5.8: The Developers' Library Update Prioritization Factors

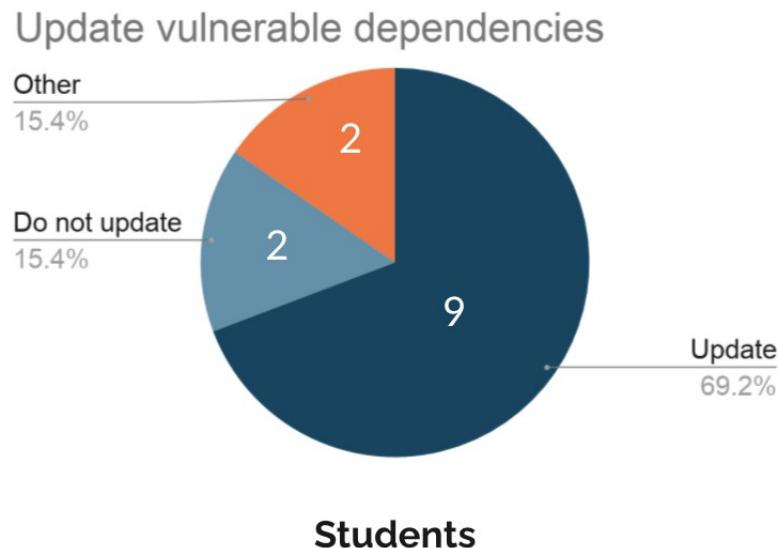


Figure 5.9: The Students' Decision on Updating Vulnerable Dependencies

ties. At the moderately important level, students prioritize the recency of vulnerabilities and the number of vulnerabilities. On the other hand, developers prioritize the gap between vulnerable versions and patch versions. These differences in result will be used as a guideline to train users to the tools according to each group prioritization factors.

5.2.3 Decision to Update Vulnerable Dependencies

Finally, we ask them whether they would update vulnerable dependencies.

From Figure 5.9 and Figure 5.10, nine out of thirteen students answered that they would update the vulnerabilities. Two students provide the reason for not updating the vulnerabilities as follows. One student declared that he or she does not know how to fix the vulnerabilities. Another student mentioned that fixing security vulnerabilities is not their priority job. There are other choices that two students would consider. First, they would choose to update vulnerabilities depending on the severity and the level of the vulnerability chain. Another student will update only significant vulnerabilities. If those vulnerabilities do not affect the project, they would not update them.

On the developer side, two out of six developers would update the vulnerabilities. One developer would not update vulnerabilities because it might cause conflict in the project, and for some projects, the developer no longer maintains that project. Fixing

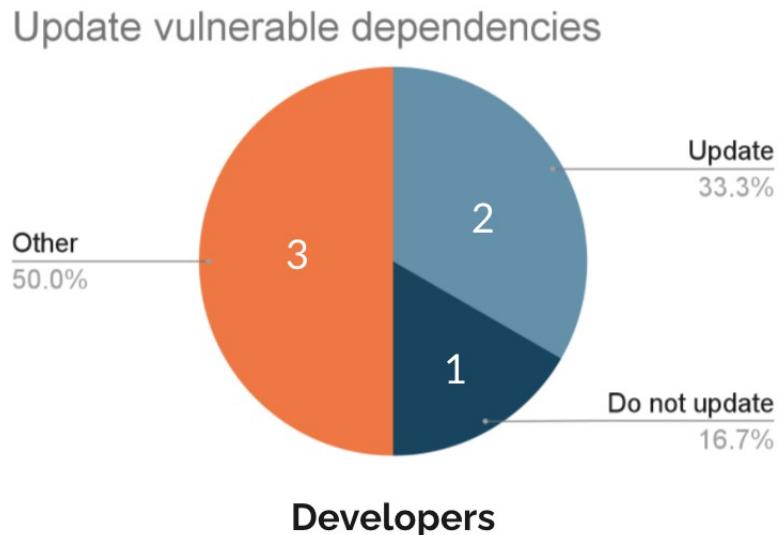


Figure 5.10: The Developers Decision on Updating Vulnerable Dependencies

vulnerabilities might lead to excessive expenses. Three developers consider other options. One developer mentioned that updating vulnerabilities depends on their severity and their effect on the overall application. He or she always update the package if it is not a breaking change. In case there is a breaking change, they would find the existing workaround first. Another developer stated that he or she would update only the matter vulnerabilities since conflict might occur after updating the vulnerabilities, and it is not their priority. The last developer mentioned that it depends on the nature of vulnerabilities and source. Some packages might be better to change than to update.

The full results of the online survey can be found in the Appendix B.

5.2.4 Feedback from Online Survey

As we received feedback from an online survey, there are two main parts that participants suggest to update which are visualization and vulnerability report. In the visualization, participants suggest to update and add some information as follow:

- Explain color of each node in each level
- Explain which node is direct or indirect dependency
- Add more information in tooltip

In the vulnerability report, there are a few suggestions from participants as follow:

- Provide a link that allows participants to link to vulnerability information
- Change summary section in the report to display as table row and provide more useful information
- Change vulnerability color representation

After we collected this information from participants, we decided to update some parts according to the participants' suggestions. In the visualization section, there are some changes in the following:

- In the top of the dependency graph, we added the short description that describes which node and color are represented to direct and indirect dependency, and also describes which node and color are represented to vulnerable nodes.
- In the tooltip sections, we provided the patch version for vulnerable nodes and provided the link for GHSA and CVE in order to allow participants to click on the vulnerability web page. Moreover, we changed the severity color that displays in tooltips according to the level of severity that displays in the report.

In the vulnerability report, we decided the changes are consists of some parts as the following:

- In each vulnerability dependency section, we added remarkable links for GHSA, CVE, and CWE that direct to vulnerability information web pages. We also provided a short description for CWE that helps participants to be aware of vulnerability type.
- In the summary section, instead of displaying the number of vulnerabilities that exist in the project, we decided to change more useful information for each vulnerability as a table row. There are four columns in summary section which are vulnerable dependency name, type of dependency whether it is direct or indirect

dependency, updating from vulnerable version to safer or patch version, and provide the level of severity column whether the dependency is critical, high, moderate or low.

- For the vulnerability color representation, we changed the low level color from yellow to blue color which is easy for users to notice.

5.3 Participants' demographic data

Table 5.3: Participants' Demographic

Participants	Demographics
A1	NAIST graduate student, little experience in npm, No prior knowledge of indirect dependency
N1	NAIST graduate student, proficient in npm, know indirect dependency
A2	NAIST graduate student, 3 years experience in npm, know indirect dependency
N2	Fourth-year undergraduate student, Little experience in npm, No prior knowledge of indirect dependency
A3	Fourth-year undergraduate student, 2-3 years experience in npm, know indirect dependency
N3	Developer, proficient in npm, know indirect dependency
A4	Developer, Little experience in npm, No prior knowledge of indirect dependency
N4	Fourth-year undergraduate student, Little experience in npm, No prior knowledge of indirect dependency
A5	Fourth-year undergraduate student, proficient in npm, know indirect dependency
N5	Fourth-year undergraduate student, Little experience in npm, No prior knowledge of indirect dependency
A6	Fourth-year undergraduate student, 3 years experience in npm, No prior knowledge of indirect dependency
N6	Fourth-year undergraduate student, Little experience in npm, know indirect dependency
A7	Fourth-year undergraduate student, little experience in npm, No prior knowledge of indirect dependency
N7	Fourth-year undergraduate student, little experience in npm, No prior knowledge of indirect dependency
A8	NAIST graduate student, little experience in npm, No prior knowledge of indirect dependency
N8	Fourth-year undergraduate student, little experience in npm, No prior knowledge of indirect dependency
A9	NAIST graduate student, little experience in npm, No prior knowledge of indirect dependency
N9	Fourth-year undergraduate student, little experience in npm, No prior knowledge of indirect dependency
A10	NAIST graduate student, little experience in npm, No prior knowledge of indirect dependency
N10	Developer, little experience in npm, No prior knowledge of indirect dependency

5.4 User Study Result

5.4.1 Results and Analysis

According to the defined methodology, we have performed a user study and found the following results.

Test 1 - Vulnerable Packages with and without Complexity

Test 1 contains direct vulnerabilities with different levels of severity and complexity (the number of dependencies that packages in the project are relied on). The participants of both the Achilles group and npm audit group see the same project with the same dependencies. We discuss their results below.

Achilles

According to Table 5.4, there are two groups of participants categorized by the prioritization order.

The changed group includes the participant A1, A2, A4, A5, A7, A8, and A9. They change the prioritization order after using the Achilles tool. We observed that they only emphasize the severity when they see the vulnerability report from Dependabot. However, after they use Achilles, they also take other factors into account. Within the changed group, six participants (A1, A2, A4, A5, A7, and A8) take **the package's complexity** into account. However, A9 is only concerned about the severity more after using Achilles, and the participant mentioned that the package's complexity does not have influenced the prioritization.

Within six participants, three participants (A1, A4, A5) use severity as the first priority factor and high complexity as the second priority factor. On the other hand, one participant (A7) uses low complexity as the second priority factor. For two participants who use complexity as the first priority factor and high severity as the second priority factor, A2 prioritizes high complexity first, while A8 prioritizes low complexity first.

The unchanged group (A3, A6, and A10) does not change their prioritization order. Nonetheless, they all mentioned that after they see Achilles, they get information about the package's complexity easier.

Table 5.4: The Result of Achilles Test Case No. 1 (Complexity)

Participants	Tool	Answers	Comparison	First Priority	Second Priority
A1	Dependabot	HC >= HS >LC >= LS	Different	High severity	-
	Achilles	HC >HS >LC >LS		High severity	High complexity
A2	Dependabot	HS >= HC >LC >LS	Different	High severity	-
	Achilles	HC >LC >HS >LS		High complexity	High severity
A3	Dependabot	HS >= HC >LC >= LS	Same	High severity	-
	Achilles	HS >= HC >LC >= LS		High severity	Low complexity
A4	Dependabot	HS >= HC >LC >= LS	Different	High severity	-
	Achilles	HC >HS >LC >LS		High severity	High complexity
A5	Dependabot	HC >HS >LS >LC	Different	High severity	-
	Achilles	HS >HC >LS >LC		High severity	Low complexity
A6	Dependabot	HC >= HS >LC >= LS	Same	Severity	
	Achilles	HC >= HS >LC >= LS		Severity	
A7	Dependabot	LS >LC >HS >HC	Different	Number of indirect dependencies	
	Achilles	HS >= LS >HC >LC		Number of indirect dependencies	Severity
A8	Dependabot	HS >HC >LC >LS	Different	High severity	Recent
	Achilles	HS >= LS >HC >LC		Low complexity	Severity
A9	Dependabot	HS >LC >LS >HC	Different	Version number	
	Achilles	HS >HC >LC >LS		High severity	Less version number
A10	Dependabot	HC >= HS >LC >= LS	Same	High severity	High complexity
	Achilles	HS >= HC >LC >= LS		High severity	Low complexity

H = High severity, L = Low severity

C = Complex (has several dependencies), S = Simple (no dependency)

Table 5.5: Factors for Prioritizing Package Updates

	First priority factor	Second priority factor	Participants	Amount
Dependabot	High Severity	Low Severity	A1, A2, A3, A4, A5, A6, A10	7
		Recent use	A8	1
	Other (Version number)		A9	1
	Other (Number of indirect dependencies)		A7	1
Total				10
Achilles	High Severity	Low Severity	A3, A6, A10	3
		High Complexity	A1, A4, A5	3
		Low Complexity	A7	1
		Version	A9	1
	High Complexity	High Severity	A2	1
	Low Complexity	High Severity	A8	1
	Total			10

According to Table 5.5, after seeing the vulnerability report from Dependabot, seven participants use severity as the significant factor for the prioritization. After witnessing the graph visualization in Achilles, there are six cases where the package complexity has become one of the factors when prioritizing the package update, either as the first or the second priority factor.

npm-audit

According to Table 5.6, there are two groups of participants categorized by the prioritization order.

There are three participants in the changed group, including N2, N4, and N8. They shift the prioritization order after using npm audit. Even though there is no significant change in the prioritization order, the report from Dependabot and npm audit provide vulnerability information at different granularity. It affects the prioritization order since these participants use vulnerability information (CVE) as the prioritization criteria.

On the other hand, there are six participants (N1, N3, N5, N6, N7, N9 and N10) in the unchanged group. Five of them (N1, N3, N5, N6 N10) use severity as the top priority factor for both tools' prioritization. Participant N7 uses the ease of patching the packages as the main priority factor for both tools.

Table 5.6: The Result of npm audit Test Case No. 1 (Complexity)

Participants	Tool	Answers	Comparison	First Priority	Second Priority
N1	Dependabot	HS > HC > LS > LC	Same	High severity	Alerted Time
	npm-audit	HS > HC > LS > LC		High severity	-
	Achilles	HS >= LS > HC >= LC	Different		
N2	Dependabot	HC > HS > LC > LS	Different	Severity	Impact on the server?
	npm-audit	HC > HS > LC >= LS		High severity	
	Achilles	HS > HC > LS > LC	Different	Severity	Less complex
N3	Dependabot	HC > HS > LC >= LS	Same	High severity	Impact with the project
	npm-audit	HC > HS > LC >= LS			
	Achilles	HS > HC > LS > LC	Different	High severity	Low complexity
N4	Dependabot	HS > HC > LC >= LS	Different	High severity	CVE
	npm-audit	HC > HS > LS > LC		High severity	npm description
	Achilles	HC > HS > LS > LC	Same	High severity	newer CVE
N5	Dependabot	HS > HC > LC > LS	Same	High severity	
	npm-audit	HS > HC > LC > LS		High severity	
	Achilles	HS > HC > LS > LC	Different	High severity	Less indirect dependencies
N6	Dependabot	HS >= HC > LS >= LC	Same	High severity	
	npm-audit	HS >= HC > LS >= LC		High severity	
	Achilles	HS > LS > HC > LC	Different	Less indirect dependencies	Severity
N7	Dependabot	LC > HS > LS > HC	Same	The ease of patching the packages	
	npm-audit	LC > HS > LS > HC		The ease of patching the packages	
	Achilles	LC > HS > LS > HC		The ease of patching the packages	
N8	Dependabot	HC > HS > LS > LC	Different	Severity	Vulnerability risk
	npm-audit	HS > HC > LC >= LS		Severity	expected error
	Achilles	HC > HS > LC > LS			More indirect dependencies
N9	Dependabot	HS >= HC >= LC > LS	Same	High severity	
	npm-audit	HS >= HC >= LC >= LS		High severity	
	Achilles	HS >= HC >= LC >= LS		High severity	
N10	Dependabot	HC >= HS > LC >= LS	Same	High severity	
	npm-audit	HC >= HS > LC >= LS		High severity	
	Achilles	HC > LC > HS > LS	Different	Complexity	Severity

H = High severity, L = Low severity

C = Complex (has several dependencies), S = Simple (no dependency)

Table 5.7: Factors for Prioritizing Package Updates

	First priority factor	Second priority factor	Cases	Number
Dependabot	High Severity	Low Severity	N5, N6, N9, N10	4
		Detail of vulnerability	N8	1
		Alert time	N1	1
		Impact with the project	N3	1
		CVE	N4	1
	Impact on the server	High Severity	N2	1
	The ease of patching the packages.		N7	1
Total				10
npm	High Severity	Low Severity	N1, N2, N4, N5, N6, N9, N10	7
		Expected error	N8	1
		Impact with the project	N3	1
	The ease of patching the packages.	Update packages with no breaking changes	N7	1
	Total			10

According to Table 5.7, after using both Dependabot and npm audit, nine participants use severity as the first priority factor.

Test 2 - Direct and indirect vulnerabilities

Test 2 contains both direct and indirect vulnerabilities with different levels of severity. Similar to Test 1, the participants of both the Achilles group and npm audit group see the same project with the same dependencies. We discuss their results below.

Achilles

According to table 5.8, there are two groups of participants categorized by the prioritization order.

The changed group, including seven participants A1, A2, A4, A6, A7, A8, and A10, change the prioritization order after seeing Achilles visualization. When they see the vulnerability report from Dependabot, participants A1, A2, A4, and A10 only emphasize on severity of the vulnerable packages. Participant A6 also considers **issue type**. Participant A8 takes **the recency** of the vulnerability into account. However, Participant A10 does not concern about the severity. The number of indirect dependencies is the only factor that A10 considers.

Nonetheless, after using Achilles, **they also take types of dependency (whether directly or indirectly) into account**. Within the changed group, four participants A1, A6, A8, and A10, use severity as their first priority factor and for the second priority factor, consider updating direct dependencies first. Two participants, A2 and A4, choose to update direct dependencies first and select high severity as the second priority factor. On the other hand, participant A7 updates the indirect dependencies first before considering the severity level.

The unchanged group (A3, A5, A9) does not change their prioritization order. Participant A3 choose to update direct dependencies first since seeing the vulnerabilities report from Dependabot. Participant A5 updates the high severity vulnerabilities first and takes direct dependencies into account since seeing the report, but without changing the prioritization. Participant A9 considers the severity level and the version number. Nevertheless, they mentioned that Achilles allows them to differentiate between direct and indirect vulnerabilities easier, similarly to Test 1.

Table 5.8: The Result of Achilles Test Case No. 2 (Direct/ Indirect)

Participants	Tool	Answers	Comparison	First Priority	Second Priority
A1	Dependabot	HD >= HI >LI >= LD	Different	High severity	-
	Achilles	HD >HI >LD >LI		High severity	Direct
A2	Dependabot	HD >= HI >= LI >= LD	Different	High severity	-
	Achilles	HD >LD >LI >HI		Direct	High severity
A3	Dependabot	HD >LD >HI >= LI	Same	Direct	High severity
	Achilles	HD >LD >HI >= LI		Direct	High severity
A4	Dependabot	HD >= HI >LI >= LD	Different	High severity	
	Achilles	HD >LD >HI >LI		Direct	High severity
A5	Dependabot	HD >HI >LD >LI	Same	High severity	-
	Achilles	HD >HI >LD >LI		High severity	Direct
A6	Dependabot	HD >= HI >LD >LI	Different	High severity	-
	Achilles	HD >HI >LD >LI		High severity	Direct
A7	Dependabot	HD >HI >LD >LI	Different	High severity	number of indirect dependencies
	Achilles	HD >LD >LI >HI		Direct	-
A8	Dependabot	HD >HI >LI >= LD	Different	High severity	Recent
	Achilles	HD >= HI >LI >= LD		High severity	
A9	Dependabot	LI >= LD >= HI >= HD	Same	Low severity	number of Indirect dependencies
	Achilles	HI >= HD >= LI >= LD		High severity	Less version number
A10	Dependabot	HD >= HI >LI >= LD	Different	High severity	-
	Achilles	HD >LD >HI >LI		Direct	Severity

H = High severity, L = Low severity

D = Direct dependency, I = Indirect dependency

Table 5.9: Factors for Prioritizing Package Updates

	First priority factor	Second priority factor	Cases	Number
Dependabot	High Severity	Low Severity	A1,A2, A4, A10	4
		Direct	A5	1
		Issue type	A6	1
		Recent use	A8	1
	Direct	High Severity	A3	1
	Indirect	Severity	A9	1
	Number of indirect dependencies		A7	1
Total				10
Achilles	High Severity	Direct	A1, A5, A6, A8, A10	5
		Version	A9	1
	Direct	High Severity	A2(with third factor of low severity), A3, A4	3
	Indirect	Severity	A7	1
	Total			10

According to Table 5.9, after seeing the vulnerability report from Dependabot, six participants use severity as the major factor for the prioritization. After seeing the graph visualization in Achilles, there are seven participants whose types of dependency have become the factor when they are prioritizing the package update as the first or the second priority factor.

npm-audit

According to table 5.10, there are two groups of participants categorized by the prioritization order.

Five participants in the changed group are N3, N4, N6, N7, N8 and N10. When they see the vulnerability report from Dependabot, they prioritize severity as the first or second factor. Participants N3 and N6 use severity as their only factors. Participant N4 also considers the CVE, and participant N7 prioritizes the ease of fixing. Participant N8 only takes vulnerabilities information into account. Participant N10 considers the risk of vulnerabilities from attacker. However, after they use npm audit, participants N3 and N6 consider the types of vulnerabilities as a factor for prioritization. N4, N7 and N8 mentioned that short description that provided by npm affect their decision on the prioritization. Participant N10 only considers the severity.

There are four participants in the unchanged group, which are N1, N2, N5, and N9. Even though participant N1 does not change the order, the criteria is different. After seeing dependabot report, participant N1 uses existing solving pull request and severity as the factor, and after seeing the npm audit report, the participant considers direct dependency and severity. Participants N2, N5, and N9 only assess the severity.

According to Table 5.11, after seeing the vulnerability report from Dependabot, eight participants use severity as the major factor for prioritization. After seeing the graph visualization in Achilles, there are four participants whose types of dependency have become the factor when they prioritize the package update as the first or second priority factor.

Table 5.10: The Result of npm audit Test Case No. 2 (Direct/ Indirect)

Participants	Tool	Answers	Comparison	First Priority	Second Priority
N1	Dependabot	HD > LD > HI > LI	Same		
	npm-audit	HD > LD > HI > LI		Direct	Severity
	Achilles	HD > LD > HI > LI			
N2	Dependabot	HD > HI > LI >= LD	Same	High severity	
	npm-audit	HD > HI > LI >= LD			
	Achilles	HD >= HI > LD >= LI			
N3	Dependabot	HI >= HD > LD >= LI	Different	High severity	
	npm-audit	HD > LD > HI > LI		Direct	severity
	Achilles	HD > LD > HI > LI	Same	Type of dependency	Severity
N4	Dependabot	HD > HI > LI > LD	Different	High severity	CVE, impact
	npm-audit	HD > LD > HI > LI		Direct	Severity
	Achilles	LD > LI > HD >= HI	Different	CVSS score	
N5	Dependabot	HD >= HI > LD >= LI	Same	High severity	-
	npm-audit	HD >= HI > LD >= LI		High severity	-
	Achilles	HD > LD > HI > LI		Direct	
N6	Dependabot	HD >= HI > LI >= LD	Different	High severity	
	npm-audit	HD > LD > HI > LI		Direct	Severity
	Achilles	HD > LD > HI > LI	Same	Direct	Severity
N7	Dependabot	HD > LD >= LI > HI	Different	Ease of fixing	Severity
	npm-audit	HD > LD > LI > HI		Ease of fixing	Severity
	Achilles	HD > LD > LI > HI	same	Ease of fixing	Severity
N8	Dependabot	HD > LI >= LD > HI	Different	Details of vulnerabilities	
	npm-audit	HD > HI > LI >= LD		Severity	Effect of vulnerabilities
	Achilles	HD > HI > LD > LI	Different	High severity	Direct
N9	Dependabot	HD >= HI >= LI >= LD	Same	High severity	
	npm-audit	HI >= HD > LI >= LD		High severity	
	Achilles	HD > HI > LD > LI	Different	High severity	Direct
N10	Dependabot	HD > HI > LD > LI	Same	High severity	
	npm-audit	HD > HI > LD > LI		High severity	
	Achilles	HI > HD > LI > LD	Different	High severity	Indirect

H = High severity, L = Low severity

D = Direct dependency, I = Indirect dependency

Table 5.11: Factors for Prioritizing Package Updates

	First priority factor	Second priority factor	Cases	Amount
Dependabot	High Severity	Low Severity	N3, N5, N6, N9	4
		how it can impact the project	N2	1
		CVE	N4	1
	Existing solving pull request	High Severity	N1	1
	Ease of fixing	High Severity	N7	1
	Detail of vulnerability		N8	1
	Chance of getting attack from the outsider		N10	1
Total				10
npm	High Severity	Low Severity	N2, N5, N8, N9, N10	5
		Direct	N6	1
		Impact the project	N4	1
	Direct	High Severity	N1, N3	2
	Patch the packages with no breaking changes	High Severity	N7	1
	Total			

Table 5.12: Comparison of Developers' Decisions

	Dependabot - Achilles	Dependabot - npm audit
Same	3	7
Different	7	3

5.4.2 Answer to Research Questions

From the results, we can answer the two research questions as follows.

RQ1: How graph visualization (Achilles) support developer's decision on prioritizing vulnerability to fix?

From the user study, **we found that the graph visualization of Achilles helps supporting developers' decisions on prioritizing vulnerability to fix by providing more information about complexity and direct/indirect dependencies compared to the traditional list of vulnerabilities provided by Dependabot**. Six participants take complexity into account after seeing the graph visualization. Nine out of ten participants consider types of dependencies (direct or indirect dependencies) as the factor for prioritization after seeing the graph visualization.

RQ2: How different types of visualization (Graph and Table) effect developer's decision on prioritizing vulnerability to fix?

The types of visualization affect developers' decision on prioritizing vulnerabilities to fix as below.

Table 5.12 Comparison of developers' decisions after seeing the graph and table visualization on the task having vulnerabilities with complexity

We can see from table 5.12, for test 1 where vulnerable packages have different complexities, the number of participants who use Achilles and change their prioritization order is larger than the number of participants who use npm audit. The table visualization can affect the developers' decision three out of nine cases, while the graph visualization can affect the developers' decision to update vulnerabilities seven out of ten cases.

Table 5.13 Comparison of developers' decisions after seeing the graph and table visualization on the task with indirect vulnerabilities

We can see from table 5.13, for test 2, where vulnerable packages have different

Table 5.13: Comparison of Developers' Decisions

	Dependabot - Achilles	Dependabot - npm audit
Same	3	5
Different	7	5

Table 5.14: showing the prioritization results after using npm audit and Achilles

	Test 1	Test 2
Same	3	5
Different	7	5

types of dependencies. **The number of participants who use Achilles and change the prioritization order is also larger than participants who use npm audit.** We can see that the graph visualization in Achilles does outperform npm audit in providing information about the types of dependencies since both of the tools represent direct and indirect dependencies differently. However, the differences between the two group is not as large as in Test 1. The table visualization can affect the developers' decision five out of nine cases, while the graph visualization can affect the developers' decision to update vulnerabilities seven out of ten cases.

We also asked participants who use npm audit to see Achilles visualization of Test 1 and Test 2 and asked for their prioritization again. What we found are as followed.

Table 5.14 shows the prioritization results after participants using npm audit and Achilles. In Test 1, **there are seven participants change their prioritization since they consider complexity of the packages which introduced by Achilles as a prioritization factor. There is no significant differences in Test 2 since both Achilles and npm audit provide types of dependencies (whether direct or indirect dependencies).**

5.5 Analysis of GitHub Project

To assess its usability in practice, we used Achilles to perform the vulnerability analysis on real-world GitHub projects to see if the tool can detect any existing security vulnerabilities. There are a few criteria that we chose to sample the repositories.

1. The repositories must be developed by using npm.
2. The repositories must have the dependencies in package.json file.

Then, we choose two sets pf projects. First, we retrieved the repositories based on the

Table 5.15: 10 Most Stars GitHub Project Used in the Study

Project	Description	No. of Stars
np	A better npm publish	6k
sinopia	A private/caching npm repository server	5.4k
nwb	A toolkit for React, Preact, Inferno & vanilla JS apps	5.3k
concurrently	Command line	4.4k
npm	JavaScript package manager	17.3k
npm-run-all	A CLI tool to run multiple npm-scripts in parallel or sequential	4.1k
node-semver	The semver parser for node	3.6k
cnpmjs.org	Private npm registry and web for Enterprise	3.4k
windows-build-tools	Install C++ Build Tools for Windows using npm	3.2k
npx	Execute npm package binaries	2.6k

Table 5.16: 10 Most Dependent GitHub Project Used in the Study

Project	Description	No. of Stars
mocha	Javascript test framework for node.js & the browser	2074
chai	BDD / TDD assertion framework for node.js and the browser that can be paired with any testing framework	1577
grunt	The JavaScript Task Runner	1623
eslint	Find and fix problems in JavaScript code	916
gulp	A toolkit of automate & enhance workflow	2758
request	HTTP request client	15820
istanbul	JS code coverage tool	391
should	Test framework for node.js	612
express	Fast, unopinionated, minimalist web framework for node	10250
sinon	Test spies, stubs and mocks for JavaScript	511

number of stars. Then, we selected the top 10 projects with the highest number of stars in the study. The information of the 10 projects is shown in the Table 5.15.

Second, we retrieved the repositories based on the number of dependent packages using the information from npm registry. This is done using the all-the-package-names[20] package in npm registry. Then, we picked the 10 projects with the highest number of dependent packages. The information of the 10 projects is shown in the Table 5.16.

5.5.1 Most Starred GitHub Projects

As shown in Figure 5.11, we found that Achilles could find the vulnerabilities in 4 most-starred GitHub repositories including sinopia, cnpmjs.org, windows-build-tools, and npx.

For sinopia, Achille found 9 direct vulnerabilities (1 low, 1 medium, 6 high, and 1 critical), and 3 indirect vulnerabilities (2 low and 1 high).

For cnpmjs.org, Achilles found 4 direct vulnerabilities (1 medium and 3 high), and 7 indirect vulnerabilities (4 low, 1 medium, and 2 high).

For windows-build-tools, Achilles found 1 indirect vulnerability (1 low).

For npx, Achilles found 1 indirect vulnerability (1 low).

The full Achilles analysis report of the 4 vulnerable projects can be found in the Appendix F.

5.5.2 Most Dependent npm Projects

As shown in Figure 5.12, Achilles did not report any vulnerability in the 10 most dependent npm projects. This shows that these projects are actively maintained with regular security checks.

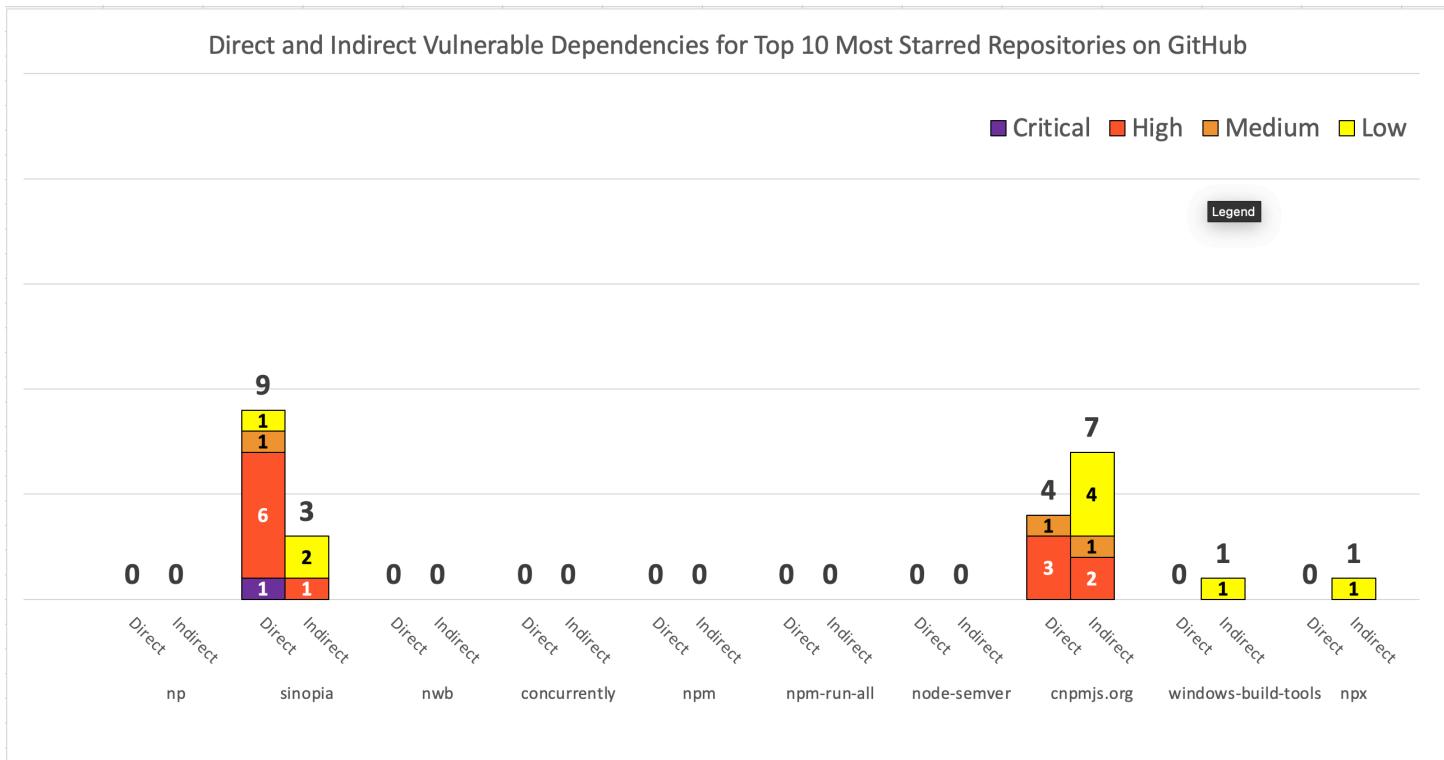


Figure 5.11: Bar graphs showing Direct and Indirect Vulnerable Dependency for Top 10 Most starred Repositories on GitHub

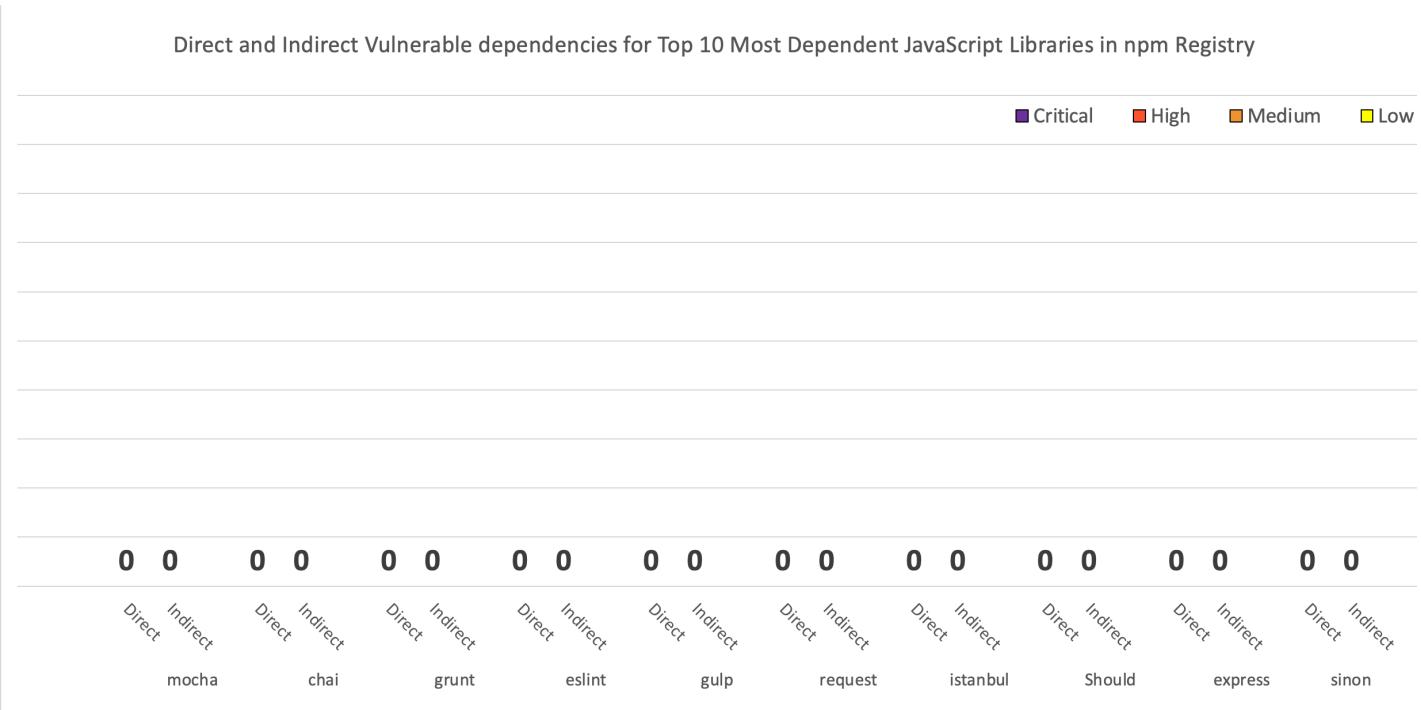


Figure 5.12: Bar graphs showing Direct and Indirect Vulnerable dependencies for Top 10 Most Dependent JavaScript Libraries in npm Registry

CHAPTER 6

CONCLUSIONS

This chapter summarizes the project and discusses the limitations and the future directions of this project

6.1 Problems and limitations

There is a limitation while selecting the user's repository to analyze. Achilles takes the repository name to search and verify whether the project is an npm project and has dependencies in package.json. However, in GitHub's code search API, there is a limitation that the API is not allowed to search the repository's content that hasn't had activity on the repository over a year. Thus, Achilles will not be able to analyze and draw dependency graph of the project that is not maintained for over a year [21].

A limitation during the visualization development includes scarcity in D3 documentation, leading to some problems during the visualization implementation. We cannot handle some factors properly, such as the force simulation when the graph contains a large number of dependencies. The performance of the graph drops significantly when it has to handle a complex relationship between dependencies. In the report part, the pdf version cannot be downloaded on mobile.

6.2 Threats to validity

The result of user study may not be fully representative for the npm audit group. This is because some of the participants in the study may be inexperienced in using npm audit. We mitigate this problem by having a short training for participants to understand how to use and interpret the result of npm audit. The result from the online survey suggested that students and developers are interested in different prioritization factors. We will use that information as a guideline to design the training process according to their interests. For students, we will mainly focus on severity, recency and number of vulnerabilities. Contrarily, for developers, we will focus on severity, recency of vulnerabilities,

and gap between patch versions and vulnerable versions.

6.3 Future work

In the future, we aim to have the tool be integrated into GitHub repositories and be able to open a pull request for updating the vulnerabilities from the visualization. We also intend to improve the graph navigation for easy understanding and the ability to filter information that the user only wants to know. Performance of the visualization is also one of the areas that we aim to enhance to handle projects with more extensive dependencies and provide insightful information for developers to manage the projects. We anticipate that Achilles can be used with not only npm projects but also other package managers, including Composer, Maven, NuGet, etc. There are also many other unexplored aspects of D3 that remain potential key factors to better performance and represent information of the visualization.

Since the COVID-19 and financial support prohibiting us from recruiting more specific and expected target user groups, our future work is to contact companies and industries which mainly focus on security vulnerabilities in order to hear comments from real user experiences and get feedback to develop our tool to meet the needs of the users.

Moreover, if our tool can be integrated with GitHub directly, it would be more convenient for users to see the graph visualizations in the GitHub repositories without opening a separate browser and be able to manage their packages instantly. This can be done as an issue report or a pull request comment.

6.3.1 Potential performance optimization

It is feasible to optimize the performance of the querying and visualization process by separating these two processes. After querying the dependencies and vulnerabilities information, we can keep this information in the data structure which supports the graph format. Hence, it will not be required to query the information when the web page is refreshed. We also compare the current and new methods for the querying process and visualizing process in the Table 6.1.

Table 6.1: Comparing the current and the proposed methods

	Pros	Cons
Current method (New query every time)	<ul style="list-style-type: none"> - Information is up to date - Use the existing GraphQL API provided by GitHub 	<ul style="list-style-type: none"> - Expensive computational power - Slower information retrieval
Local database - Indexing	<ul style="list-style-type: none"> - Cheap computational power - Faster information retrieval 	<ul style="list-style-type: none"> - Information is not updated - Need to design new database

6.3.2 Potentially better visualization method

We plan to change the default view to show only vulnerable nodes and provide other options for users to visualize according to their preferences. The possibilities include showing an entire graph, only the direct dependencies or indirect dependencies. We also plan to investigate other visualization which would better represent dependencies and vulnerabilities information, e.g., hierarchical visualization.

6.4 Conclusion

The main goal of this project is to provide developers a tool that can help them detect, visualize and report vulnerabilities in their projects. We create an automated tool called Achilles to detect both direct and indirect vulnerabilities and visualize the dependencies graph with identified vulnerable node. Users can also see and download the vulnerabilities report.

Achilles requires a user to log in with a GitHub account in order to retrieve the list of user's repositories. After the user selects a repository and package.json file to analyze, Achilles will check the packages vulnerability from GitHub Security Advisory, retrieve the chain of dependencies of the packages from npm registry, and visualize the dependencies graph. Achilles also provides the security vulnerability report, which can be downloaded in pdf format.

We evaluate the Achilles tool by three methods: gathering information from an online survey, conducting a user study, and analyzing GitHub projects. The responses from the online survey suggested that most students and developers use severity as a highly important factor that they consider when updating vulnerabilities. Comparing the prioritization for vulnerabilities update after participants use the Dependabot report, npm audit, and Achilles, we found that most participants change their prioritization order after

using Achilles due to information about packages' complexity and types of dependencies (direct and indirect vulnerabilities) Achilles introduced. We also used Achilles to analyze the vulnerability in the real-world GitHub projects. We tested with the Top 10 most starred repositories on GitHub. Achilles can detect vulnerabilities in four repositories. Direct and indirect vulnerabilities are detected in two repositories which are sinopia and cnpmjs.org. An indirect vulnerability is detected in windows-build-tools and npx. We also tested Achilles with Top 10 Most Dependent JavaScript Libraries in npm registry. Achilles did not report any vulnerabilities.

The evaluation results show that Achilles tool is a beneficial addition to contemporary software development to detect indirect dependencies and comprehend potential vulnerabilities via the dependency graph visualization and analysis report.

APPENDIX A
IRB DOCUMENT



COE No. MU-CIRB 2021/085.1103

Mahidol University Central Institutional Review Board*Certificate of Exemption***Title of Project:** Achilles: An automated tool for analyzing npm library vulnerabilities**Protocol Number:** MU-CIRB 2021/084.1502**Principal Investigator:** Lect. Dr. Chaiyong Ragkhitwetsagul**Co- Investigators:** 1) Miss Vipawan Jarukitpipat

2) Miss Wachirayana Wanprasert

3) Mr. Klinton Chhun

Affiliation: Faculty of Information and Communication Technology, Mahidol University**The criteria of Exemption:** Research involving the use of survey and interview procedures and:

- Recorded information CANNOT readily identify the subject (directly or indirectly/linked) OR
- Any disclosure of responses outside of the research would NOT place subject at risk (criminal, civil liability, financial, employability, educational advancement, reputation)

MU-CIRB is in full compliance with International Guidelines for Human Research Protection such as Declaration of Helsinki, The Belmont Report, CIOMS Guidelines and the International Conference on Harmonization on Good Clinical Practice (ICH-GCP)

Date of Determination: 11 March 2021**Signature of Chairperson:**

(Emeritus Professor Dr. Wariya Chinwanno)

MU-CIRB Chair

MU-CIRB Address: Office of the President, Mahidol University, 4th Floor, Room Number 411

999 Phutthamonthon 4 Road, Salaya, Nakhonpathom 73170, Thailand

Tel: 66 (0) 2849 6224, 6225 Fax: 66 (0) 2849 6224

E-mail: mucirb@gmail.com Website: <http://www.sp.mahidol.ac.th>

MU-CIRB	คณะกรรมการจัดซื้อจัดจ้างในคณลักษณะ มาตรฐานที่ดี		แก้ไขวันที่ 21/02/2021
เอกสารชี้แจงสำหรับผู้เข้าร่วมวิจัยที่อายุ ๑๘ ปีขึ้นไป (Participants Information sheet (age 18 years or older))		หน้าที่ 1 ของ 2 หน้า	
เอกสารชี้แจงสำหรับผู้เข้าร่วมวิจัยที่อายุ ๑๘ ปีขึ้นไป (Participants Information sheet (age 18 years or older))			
<input type="checkbox"/> ต้นฉบับ <input type="checkbox"/> การปรับเปลี่ยนครั้งที่ 		วันที่/...../.....	
<p>ในเอกสารนี้อาจมีข้อความที่ท่านอ่านแล้วยังไม่เข้าใจ โปรดสอบถามหัวหน้าโครงการวิจัย หรือผู้แทนให้ช่วยอธิบายจนกว่าจะเข้าใจดี ท่านจะได้รับเอกสารนี้ 1 ฉบับ นำกลับไปอ่านที่บ้านเพื่อปรึกษาหารือกับญาติพี่น้อง เพื่อสนับสนุน แพทย์ประจำตัวของท่าน แพทย์ท่านอื่น หรือผู้ที่ท่านต้องการปรึกษา เพื่อช่วยในการตัดสินใจเข้าร่วมการวิจัย</p>			

ชื่อโครงการ อดีตส.: เกรื่องมือเพื่อการวิเคราะห์ซ่องโหว่ด้านความปลอดภัยของอีนพีอีนไลนารี่

ชื่อผู้วิจัย ดร. ชัยวงศ์ รักขิตเวชสกุล

สถานที่วิจัย สถานที่ทำงานและหมายเลขอรหัสพทที่คิดต่อได้ทั้งในและนอกเวลาราชการ ได้ตลอด 24 ชั่วโมง

คณะเทคโนโลยีสารสนเทศและการสื่อสาร ม.มหิดล โทรศัพท์ 089-1763372

ผู้ให้ทุน ไม่มี

โครงการวิจัยนี้เป็นส่วนหนึ่งของวิชา ITCS492 Senior Project II คณะเทคโนโลยีสารสนเทศและการสื่อสาร มหาวิทยาลัยมหิดล คณะผู้จัดทำได้พัฒนาเครื่องมือในการค้นหาและแสดงผลช่องโหว่ด้านความปลอดภัยของ npm dependency ดังนั้นจุดประสงค์ของแบบสอบถามนี้จึงมีเพื่อสอบถามความคิดเห็นจากผู้ที่อาจเกิดขึ้นจากช่องโหว่ด้านความปลอดภัยของ npm dependency รวมถึงความคิดเห็นต่อตัวอย่างการใช้งานเครื่องมือดังกล่าว

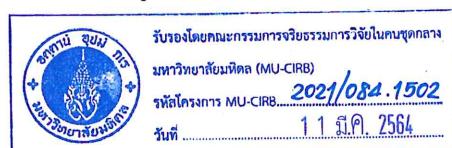
ท่านได้รับเชิญให้เข้าร่วมวิจัยนี้ เพราะท่านเป็นนักพัฒนาซอฟต์แวร์ที่มีลักษณะเหมาะสมกับโครงการวิจัยนี้

ท่านอาจจะไม่ได้รับประโยชน์จากการวิจัยนี้โดยตรง แต่หากงานวิจัยนี้ได้ผลดีจะเป็นประโยชน์เนื่องจากเครื่องมือคิดส์ที่พัฒนาขึ้น จะสามารถช่วยเหลือนักพัฒนาซอฟต์แวร์ให้สามารถวิเคราะห์และเข้าใจปัญหาด้านของโหว่ด้านความปลอดภัยของ npm dependency ได้อย่างอัตโนมัติ

จะมีผู้เข้าร่วมการวิจัยนี้ทั้งสิ้นประมาณ 40 คน ระยะเวลาที่ใช้ในการวิจัย ประมาณ 1 ชั่วโมง ต่อผู้เข้าร่วมวิจัย 1 ท่าน

หากท่านตัดสินใจเข้าร่วมการวิจัยแล้ว จะมีขั้นตอนการวิจัยดังต่อไปนี้คือ

- 1) ทีมผู้วิจัยจะแบ่งผู้เข้าร่วมการวิจัยออกเป็น 2 กลุ่ม ได้แก่ กลุ่มควบคุม (controlled group) และกลุ่มทดลอง (experimental group) โดยกลุ่มควบคุมจะเป็นกลุ่มผู้เข้าร่วมวิจัยที่ได้รับข้อมูล software vulnerability ผ่านเครื่องมือ GitHub dependabot และกลุ่มทดลองจะเป็นกลุ่มผู้เข้าร่วมวิจัยที่ได้รับข้อมูล software vulnerability ผ่านเครื่องมือ Achilles
- 2) ผู้วิจัยให้ผู้เข้าร่วมวิจัยทั้งสองกลุ่มทำความเข้าใจกับปัญหาและผลกระทบที่เกิดขึ้นจาก software vulnerability และทำการสัมภาษณ์ผู้เข้าร่วมวิจัยถึงความเข้าใจ ข้อมูลที่ได้ และวิธีการแก้ปัญหา



MU-CIRB	คณะกรรมการจัดการวิจัยในคนส่วนกลาง มหาวิทยาลัยมหิดล เอกสารชี้แจงสำหรับผู้เข้าร่วมวิจัยที่อายุ ๑๘ ปีขึ้นไป (Participants Information sheet (age 18 years or older))	แก้ไขขั้นที่ 21/02/2021 หน้าที่ 2 จาก 2 หน้า
---------	--	---

3) ทีมผู้วิจัยทำการเปรียบเทียบผลการทดลอง เพื่อศึกษาว่าเครื่องมือ Achilles สามารถช่วยให้นักพัฒนาโปรแกรมเข้าใจและระบุปัญหาด้าน software vulnerability ได้ดีกว่า dependabot หรือไม่

ความเสี่ยงที่อาจจะเกิดขึ้นเมื่อเข้าร่วมการวิจัย คือ ท่านอาจรู้สึกอึดอัด ไม่สบายใจ เครียด กับบางคำถ้า ท่านมีสิทธิ์ที่จะไม่ตอบคำถามเหล่านี้ได้ หรือเสียเวลา

หากท่านไม่เข้าร่วมในการวิจัยนี้จะไม่มีผลต่อการเรียนการสอนหรือหน้าที่การทำงานของท่านแต่อย่างใด

หากมีอาการผิดปกติ รู้สึกไม่สบายกาย หรือมีผลกระแทบตื้อจิตใจของท่านเกิดขึ้นระหว่างการวิจัย ท่านจะแจ้งผู้วิจัยโดยเร็วที่สุด และหากท่านมีข้อสงสัยที่จะสอบถามที่เกี่ยวข้องกับการวิจัย หรือหากเกิดการบาดเจ็บ/เจ็บป่วย หรือหากเกิดเหตุการณ์ไม่พึงประสงค์จากการวิจัยกับท่าน ท่านสามารถติดต่อได้ที่ ดร. ชัยยงค์ รักขิตเวชสกุล หมายเลขโทรศัพท์ 089-1763372 ได้ตลอด 24 ชั่วโมง

หากเกิดผลข้างเคียงที่ไม่พึงประสงค์จากการวิจัย ทีมผู้วิจัยจะติดต่อกับผู้เชี่ยวชาญเพื่อตรวจสอบอาการข้างเคียงที่ไม่พึงประสงค์จากการวิจัยดังกล่าว

หากมีข้อสงสัยที่จะสอบถามที่เกี่ยวข้องกับการวิจัย หรือเมื่อบาดาเจ็บ/เจ็บป่วยจากการวิจัยท่านสามารถติดต่อได้ที่ ดร. ชัยยงค์ รักขิตเวชสกุล หมายเลขโทรศัพท์ 089-1763372 ได้ตลอด 24 ชั่วโมง

ค่าตอบแทนที่จะได้รับ

ไม่มี

ค่าใช้จ่ายที่ผู้เข้าร่วมการวิจัยจะต้องรับผิดชอบเอง

ไม่มี

หากมีข้อมูลเพิ่มเติมทั้งด้านประ โยชน์และไทยที่เกี่ยวข้องกับการวิจัยนี้ ผู้วิจัยจะแจ้งให้ทราบโดยรวดเร็วไม่ล�บัง

ข้อมูลส่วนตัวของผู้เข้าร่วมการวิจัยจะถูกเก็บรักษาไว้ ไม่เปิดเผยต่อสาธารณะเป็นรายบุคคล แต่จะรายงานผลการวิจัยเป็นข้อมูลส่วนรวม ข้อมูลของผู้เข้าร่วมการวิจัยเป็นรายบุคคลอาจมีความบุคคลบานกถล่มเข้ามาตรวจสอบได้ เช่น ผู้ให้ทุนวิจัย, สถาบัน หรือองค์กรของรัฐที่มีหน้าที่ตรวจสอบ, คณะกรรมการจริยธรรมฯ เป็นต้น

ผู้เข้าร่วมการวิจัยมีสิทธิ์ถอนตัวออกจากโครงการวิจัยเมื่อใดก็ได้ โดยไม่ต้องแจ้งให้ทราบล่วงหน้า และการไม่เข้าร่วมการวิจัยหรือถอนตัวออกจากโครงการวิจัยนี้ จะไม่มีผลกระทบต่อการบริการและการรักษาที่สมควรจะได้รับแต่ประการใด

โครงการวิจัยนี้ได้รับการพิจารณาอนุมัติจากโครงการวิจัยเมื่อใดก็ได้ โดยไม่ต้องแจ้งให้ทราบล่วงหน้า และการไม่เข้าร่วมการวิจัยนี้จะไม่มีผลกระทบต่อการบริการและการรักษาที่สมควรจะได้รับแต่ประการใด

โครงการวิจัยนี้ได้รับการอนุมัติจากคณะกรรมการวิจัยในคนส่วนกลาง มหาวิทยาลัยมหิดล ซึ่งมี สำนักงานอยู่ที่ สำนักงานอธิการบดีมหาวิทยาลัยมหิดล ถนนพุทธมณฑล สาย 4 ตำบลศาลายา อำเภอพุทธมณฑล จังหวัดนครปฐม ๗๓๑๗๐ หมายเลขโทรศัพท์ 02-849-6224, 6225 โทรสาร 02-849-6224 หากท่านได้รับการปฏิบัติไม่ตรงตามที่ระบุไว้ ท่านสามารถติดต่อกับประธานคณะกรรมการฯ หรือผู้แทน ได้ตามสถานที่และหมายเลขโทรศัพท์ข้างต้น

ข้าพเจ้าได้อ่านรายละเอียดในเอกสารนี้ครบถ้วนแล้ว

ลงชื่อ..... ผู้เข้าร่วมวิจัย

(.....)

วันที่...../...../.....

MU-CIRB	คณะกรรมการจัดยุทธศาสตร์การวิจัยในคนส่วนกลาง มหาวิทยาลัยมหิดล	แก้ไขวันที่ 10/07/2020
หนังสือแสดงเจตนาข้อมูลของผู้เข้าร่วมวิจัยที่อายุ ๑๘ ปี ขึ้นไป (Informed consent form for research participants (age 18 years or older))		หน้าที่ 1 ของ 2 หน้า
หนังสือแสดงเจตนาข้อมูลของผู้เข้าร่วมวิจัยที่อายุ ๑๘ ปี ขึ้นไป (Informed consent form for research participants (age 18 years or older))		
<input type="checkbox"/> ต้นฉบับ <input type="checkbox"/> การปรับเปลี่ยนครั้งที่.....		วันที่...../...../.....

วันที่..... เดือน..... พ.ศ.....

ข้าพเจ้า..... อายุ..... ปี อาศัยอยู่บ้านเลขที่.....

ถนน..... ตำบล..... อำเภอ..... จังหวัด.....

รหัสไปรษณีย์..... โทรศัพท์

ขอแสดงเจตนาข้อมูลของ โครงการวิจัยเรื่อง อดีตส.: เครื่องมือเพื่อการวิเคราะห์ช่องโหว่ด้านความปลอดภัย
ของอุปกรณ์ในเครือข่าย

โดยข้าพเจ้าได้รับทราบรายละเอียดเกี่ยวกับที่มาและจุดมุ่งหมายในการท้าววิธารายละเอียดข้อตอนต่างๆ ที่จะต้อง^ก
ปฏิบัติหรือได้รับการปฏิบัติ ประ โยชน์ที่คาดว่าจะได้รับของการวิจัยและความเสี่ยงที่อาจเกิดขึ้นจากการเข้าร่วมการวิจัย
รวมทั้งแนวทางป้องกันและแก้ไขหากเกิดอันตรายขึ้น ค่าตอบแทนที่จะได้รับ ค่าใช้จ่ายที่ข้าพเจ้าจะต้องรับผิดชอบจ่ายเอง
โดยได้อ่านข้อความที่มีรายละเอียดอยู่ในเอกสารนี้ แจ้งผู้เข้าร่วมการวิจัยโดยตลอด อีกทั้งยังได้รับคำอธิบายและตอบข้อ^ก
สงสัยจากหัวหน้าโครงการวิจัยเป็นที่เรียบร้อยแล้ว โดยไม่มีสิ่งใดปิดบังซ่อนเร้น

ข้าพเจ้าจึงสมัครใจเข้าร่วมในโครงการวิจัยนี้ :

ข้าพเจ้าได้ทราบถึงสิทธิ์ที่ข้าพเจ้าจะได้รับข้อมูลเพิ่มเติมทั้งทางด้านประ โยชน์และไทยจากการเข้าร่วมการวิจัย
และสามารถถอนตัวหรือดูแลตัวเองได้โดยไม่มีผลกระทบใดๆ ให้กับผู้อื่น ได้รับค่าใช้จ่ายที่สูงกว่าค่าเดินทาง แต่จะไม่เสียค่าเดินทาง
ข้อมูลให้ผู้วิจัยใช้ข้อมูลส่วนตัวของข้าพเจ้าที่ได้รับจากการวิจัย แต่จะไม่เผยแพร่ต่อสาธารณะเป็นรายบุคคล โดยจะ^ก
นำเสนอเป็นข้อมูลโดยรวมจากการวิจัยเท่านั้น

หากข้าพเจ้ามีอาการผิดปกติ รู้สึกไม่สบายกาย หรือมีผลกระทบต่อจิตใจของข้าพเจ้าเกิดขึ้นระหว่างการวิจัย
ข้าพเจ้าจะแจ้งผู้วิจัยโดยเร็วที่สุด และหากข้าพเจ้ามีข้อข้องใจเกี่ยวกับข้อตอนของการวิจัย หรือหากเกิดผลข้างเคียงที่ไม่
พึงประสงค์จากการวิจัยขึ้นกับข้าพเจ้า ข้าพเจ้าจะสามารถติดต่อ ก.ด.ร.ช.ย.ย.ก. รักษาเวชสกุล หมายเลขโทรศัพท์ 089-
1763372 ตลอด 24 ชั่วโมง



MU-CIRB	คณะกรรมการจัดการวิจัยในคนส่วนกลาง มหาวิทยาลัยมหิดล หนังสือแสดงเจตนาข้อมูลของผู้เข้าร่วมวิจัยที่อายุ ๑๘ ปี ขึ้นไป (Informed consent form for research participants (age 18 years or older))	แก้ไขวันที่ 10/07/2020 หน้าที่ 2 จาก 2 หน้า
---------	---	--

หากเข้าพเจ้า ได้รับการปฏิบัติไม่ตรงตามที่ได้ระบุไว้ในเอกสารชี้แจงผู้เข้าร่วมการวิจัย เข้าพเจ้าจะสามารถติดต่อ กับประธานคณะกรรมการจัดการวิจัยในคนหรือผู้แทน ได้ที่สำนักงานคณะกรรมการจัดการวิจัยในคน ส่วนกลาง สำนักงานอธิการบดี มหาวิทยาลัยมหิดล หมายเลขอุทธรรศพที่ 02-849-6224 , 6225 โทรศัพท์ 02-849-6224

เข้าใจข้อความในเอกสารชี้แจงผู้เข้าร่วมการวิจัย และหนังสือแสดงเจตนาข้อมูลนี้โดยตลอดแล้ว จึงลง ลายมือชื่อไว้

ลงชื่อ.....

ลงชื่อ.....

(.....)

(ดร. ชัยยงค์ รักษาเวชสกุล)

ผู้เข้าร่วมการวิจัย/ผู้แทน โดยชอบธรรม

ผู้ให้ข้อมูลและขอความยินยอม/หัวหน้าโครงการวิจัย

วันที่...../...../.....

วันที่...../...../.....

ในกรณีผู้เข้าร่วมการวิจัยไม่สามารถอ่านหนังสือได้ผู้ที่อ่านข้อความทั้งหมดแทนผู้เข้าร่วมการวิจัยคือ.....
จึงได้ลงลายมือชื่อไว้เป็นพยาน

ลงชื่อ..... พยาน

(.....)

วันที่...../...../.....

MU-CIRB	คณะกรรมการจัดทำแบบสอบถามผู้เข้าร่วมวิจัยในคนส่วนกลาง มหาวิทยาลัยมหิดล	แก้ไขวันที่ 10/07/2020
เอกสารชี้แจงผู้เข้าร่วมวิจัย โดยการตอบแบบสอบถาม (Self-Administered Questionnaire Participant Information Sheet)		หน้าที่ 1 ของ 1 หน้า
เอกสารชี้แจงผู้เข้าร่วมวิจัย โดยการตอบแบบสอบถาม (Self-Administered Questionnaire Participant Information Sheet)		
<input type="checkbox"/> ต้นฉบับ <input type="checkbox"/> การปรับเปลี่ยนครั้งที่.....		วันที่...../...../.....

เรียน ผู้ดูแลแบบสอบถามทุกท่าน

ด้วย กระผม ดร. ชัยยงค์ รักพิชเวศสกุล อาจารย์ประจำคณะเทคโนโลยีสารสนเทศและการสื่อสาร มหาวิทยาลัยมหิดล มีความประสงค์ที่จะดำเนินการวิจัย เรื่อง อคิลีส: เครื่องมือเพื่อการวิเคราะห์ช่องโหว่ด้านความปลอดภัยของอีเมลในโลกออนไลน์ ซึ่งประเมินช่วงเวลาที่คาดว่าจะได้รับคือ (1) ผลที่ได้จากการวิจัยจะถูกใช้เพื่อวัดประสิทธิภาพของเครื่องมือ Achilles และความทึบตอนในการใช้งานครื่องมือในการค้นหาแหล่งข้าวไฟ software vulnerability และ (2) คำแนะนำและข้อเสนอแนะจากการตอบแบบสอบถาม จะถูกนำมาพิจารณาเพื่อปรับปรุงเครื่องมือ Achilles ให้ดียิ่งขึ้น

ท่านได้รับเชิญให้เข้าร่วมการวิจัยนี้ เพราะท่านเป็นนักพัฒนาซอฟต์แวร์ที่มีลักษณะเหมาะสมกับโครงการวิจัยนี้ ในกรณีนี้ ผู้วิจัยมีความจำเป็นต้องเก็บรวบรวมข้อมูลโดยใช้แบบสอบถามตามเรื่อง “แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไอลิบรารี่ของบุคคลที่ 3 (Third-party library) และเครื่องมือในการตรวจจับ” ซึ่งประกอบด้วยคำถาม 5 ส่วน จำนวน 15 ข้อ ใช้เวลาในการตอบประมาณ 15 - 20 นาที ผู้วิจัยจะขอรับแบบสอบถามคืนโดยการส่ง อีเมลมาบ้างทีมผู้วิจัย

เนื่องจากแบบสอบถามประกอบด้วยคำถามหลายส่วน จึงขอความกรุณารายให้ท่านพิจารณาตอบตามความรู้สึกของท่านให้มากที่สุด โดยข้อมูลและคำตอบทั้งหมดจะถูกปกปิดเป็นความลับ และจะนำมายังการวิเคราะห์ผลการศึกษาครั้งนี้โดยอกมา เป็นภาพรวมของการวิจัยเท่านั้น จึงไม่มีผลกระทบใด ๆ ต่อผู้ตอบหรือหน่วยงานของผู้ตอบ เมื่อจากไม่สามารถนำมารีบุนเดิน เจาะจงหากผู้ตอบได้ ท่านมีสิทธิ์ที่จะไม่ตอบคำถามข้อใดข้อหนึ่ง หากท่านไม่สนใจให้วิธีอีกด้วยที่จะตอบคำถามนั้น หรือไม่ต้อง แบบสอบถามทั้งหมดก็ได้ โดยไม่มีผลกระทบต่อการปฏิบัติงานใด ๆ ของท่าน ท่านมีสิทธิ์ที่จะไม่เข้าร่วมการวิจัยก็ได้โดยไม่ต้องแจ้งเหตุผล

หากผู้เข้าร่วมวิจัยมีข้อสงสัยเกี่ยวกับการวิจัยหรือแบบสอบถาม สามารถติดต่อสอบถามได้ที่ สถานที่ติดต่อ ดร. ชัยยงค์ รักพิชเวศสกุล ในวันและเวลาราชการ หรือ โทรศัพท์ที่ติดต่อได้ 089-1763372

โครงการวิจัยนี้ได้รับการพิจารณาไว้รองจาก คณะกรรมการจัดทำแบบสอบถาม สำหรับการวิจัยในคนของมหาวิทยาลัยมหิดล สำนักงาน อุปถัมภ์ สำนักงานอธิการบดีมหาวิทยาลัยมหิดล ถนนพุทธมณฑล สาย 4 ตำบลศาลายา อำเภอพุทธมณฑล จังหวัดนครปฐม 73170 หมายเลขโทรศัพท์ 02-849-6224 , 6225 โทรสาร 02-849-6224 หากท่านได้รับการปฏิบัติไม่ตรงตามที่ระบุไว้ ท่านสามารถติดต่อ ประธานกรรมการฯหรือผู้แทน ได้ตามสถานที่และหมายเลขอโทรศัพท์ข้างต้น

ขอขอบพระคุณที่กรุณาสละเวลาในการตอบแบบสอบถาม

ขอแสดงความนับถือ

ดร. ชัยยงค์ รักพิชเวศสกุล



APPENDIX B
ONLINE SURVEY

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และเครื่องมือในการตรวจจับ

แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และเครื่องมือในการตรวจจับ

Survey about the software vulnerability from third-party library and Achilles tool for detecting security vulnerabilities

* Required

1. กรุณาเลือกภาษาของแบบสอบถาม *

Please choose the language of the survey

Mark only one oval.

ภาษาไทย

Skip to section 2 (แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และเครื่องมือในการตรวจจับ)

English

Skip to section 8 (Survey on security vulnerability from third-party library and detection tool)

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) ฯ...

แบบสอบถามนี้เป็นส่วนหนึ่งของวิชา ITCS492 Senior Project II คณะเทคโนโลยีสารสนเทศและการสื่อสาร มหาวิทยาลัยมหิดล คณะผู้จัดทำได้พัฒนาเครื่องมือในการค้นหาและแสดงผลช่องโหว่ด้านความปลอดภัยของ npm dependency ดังนั้นจุดประสงค์ของแบบสอบถามนี้จึงมีเพื่อสอบถามความตระหนักรู้ในปัญหาที่อาจเกิดขึ้นจากช่องโหว่ด้านความปลอดภัยของ npm dependency รวมถึงความคิดเห็นต่อตัวอย่างการใช้งานเครื่องมือดังกล่าว

**แบบสอบถาม
เกี่ยวกับช่องโหว่
ด้านความ
ปลอดภัย ของ
ซอฟต์แวร์
(Security
Vulnerability) ที่
เกิดขึ้นจาก
ไลบรารีของบุคคล
ที่ 3 (Third-party
library) และ
เครื่องมือในการ
ตรวจสอบ**

เนื่องจากแบบสอบถามประกอบด้วยคำตามหล่ายส่วน จึงขอความกรุณาให้ท่านพิจารณาตอบตามความเข้าใจท่านให้มากที่สุด โดยข้อมูลและค่าตอบทั้งหมดจะถูกปิดเป็นความลับ และจะนำมายังนักวิเคราะห์ผลการศึกษาครั้งนี้โดยอุปกรณ์เป็นภาพรวมของ การวิจัยเท่านั้น จึงไม่มีผลกระทบใด ๆ ต่อผู้ตอบหรือหน่วยงานของผู้ตอบ เป็นเชิงๆจากไม่สามารถนำมาสืบค้นจะทางผู้ตอบได้ ท่านมีสิทธิ์ที่จะไม่ตอบคำถามข้อใดข้อหนึ่ง หากท่านไม่สบายใจหรืออิดอก็ที่จะตอบคำว่า “ไม่” หรือไม่ตอบแบบสอบถามทั้งหมดเลก็ได้ โดยไม่มีผลกระทบต่อการปฏิบัติงานใด ๆ ของท่าน ท่านมีสิทธิ์ที่จะไม่เข้าร่วมการวิจัย ก็ได้โดยไม่ต้องแจ้งเหตุผล

หากผู้เข้าร่วมวิจัยมีข้อสงสัยเกี่ยวกับการวิจัยหรือแบบสอบถาม สามารถติดต่อสอบถามได้ที่ ดร. ชัยยอง วงศิตเวชสกุล (อาจารย์ที่ปรึกษา) สถานที่ติดต่อ คณะเทคโนโลยีสารสนเทศและการสื่อสาร มหาวิทยาลัยมหิดล 999 ถนนพุทธมณฑลสาย 4 ตำบลศาลา อำเภอพุทธมณฑล, จังหวัดนครปฐม 73170 ประเทศไทย ในวันและเวลาทำการ หรือโทรศัพท์ที่ติดต่อได้ (02) 441-0909 หรืออีเมล chaiyong.rag@mahidol.ac.th

โครงการวิจัยนี้ได้รับการพิจารณารับรองจาก คณะกรรมการจริยธรรมการวิจัย ในคุณของ มหาวิทยาลัยมหิดล เลขที่ MU-CIRB 2021/085.1103 สำนักงานอธิบดี สำนักงาน อธิการบดีมหาวิทยาลัยมหิดล ถนนพุทธมณฑล สาย 4 ตำบลศาลา อำเภอพุทธมณฑล จังหวัดนครปฐม 73170 หมายเลขโทรศัพท์ 02-849-6224, 6225 โทรสาร 02-849-6224 หากท่านได้รับการปฏิบัติไม่ตรงตามที่ระบุไว้ ท่านสามารถติดต่อประธานกรรมการการวิจัยเพื่อแก้ไข ได้ตามสถานที่และหมายเลขอโทรศัพท์ข้างต้น

ขอขอบพระคุณที่กรุณาสละเวลาในการตอบแบบสอบถาม
นางสาว วิภาวรรณ จารุกิจพิพัฒน์
นางสาว วชิรญาณ วันประเสริฐ
นาย คลินเด้น ชน
ดร. ชัยยอง วงศิตเวชสกุล (อาจารย์ที่ปรึกษา)

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ

เอกสารชี้แจงผู้เข้าร่วมวิจัย โดยการตอบแบบสอบถาม

MU-CIRB	คณะกรรมการร่วมธรรมการวิจัยในคนส่วนกลาง มหาวิทยาลัยนเรศวร	แก้ไขวันที่ 10/07/2020
เอกสารชี้แจงผู้เข้าร่วมวิจัย โภคภารตแบบสอบถาม (Self-Administered Questionnaire Participant Information Sheet)		หน้าที่ 1 จาก 1 หน้า
เอกสารชี้แจงผู้เข้าร่วมวิจัย โภคภารตแบบสอบถาม (Self-Administered Questionnaire Participant Information Sheet)		
<input checked="" type="checkbox"/> ต้นฉบับ	<input type="checkbox"/> การปรับเปลี่ยนครั้งที่.....	วันที่...11.../..3../2564

เรียน ผู้ดูแลแบบสอบถามทุกท่าน

ด้วย กรมธรรม ดร. ขัยคงต์ รัชวิจิเวชสกุล อาจารย์ประจำคณะเทคโนโลยีสารสนเทศและการท่องเที่ยว มหาวิทยาลัยมหิดล มีความประทับใจที่อาจารย์วิจัยนี้เป็นอุดมสังคม เกี่ยวกับเรื่องความปลอดภัยของอาชญากรรมทางไซเบอร์ ที่ชี้ให้เห็นถึงความสำคัญของการเฝ้าระวังและป้องกันภัยคุกคามที่อาจเกิดขึ้นในโลกดิจิทัล อย่างเช่น การโจมตีทางไซเบอร์ หรือการลักขโมยข้อมูลส่วนบุคคล ซึ่งเป็นภัยคุกคามที่เพิ่มมากขึ้นในยุคดิจิทัล ท่านได้แสดงให้เห็นถึงความสามารถในการวิเคราะห์และแก้ไขปัญหาด้านความปลอดภัยอย่างเชิงลึก ที่สอดคล้องกับความต้องการของผู้ใช้งานจริง ทำให้เกิดประโยชน์อย่างสูงต่อสังคม ท่านยังได้เสนอแนวทางการดำเนินการต่อไป เช่น การพัฒนาเครื่องมือในการเฝ้าระวังและป้องกันภัยคุกคาม ที่สามารถนำไปใช้ในภาคอุตสาหกรรม ภาครัฐ และภาคประชาชน ได้เป็นอย่างดี

ท่านได้รับเชิญให้เข้าร่วมการวิจัยขั้นพื้นฐานเพื่อท่านเป็นนักพัฒนาซอฟต์แวร์ที่มีลักษณะเหมาะสมกับโครงการวิจัยนี้ ในกรณีนี้ ผู้รับเชิญมีความจำเป็นต้องเก็บรวบรวมข้อมูลโดยใช้แบบสอบถามตามเรื่อง “แบบสอบถามเกี่ยวกับช่องทางคุณภาพของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารี่ของบุคคลที่ 3 (Third-party library) และเครื่องมือในการตรวจสอบ” ซึ่งจะประกอบด้วยคำถาม 5 สรุป จำนวน 15 ข้อ ใช้เวลาในการตอบประมาณ 15 - 20 นาที ผู้รับเชิญขอรับแบบสอบถามคืนโดยการส่งอีเมลมาบังคับผู้รับเชิญ

หากผู้เข้าร่วมวิจัยนี้ข้องรับยกเว้นการวิจัยหรือหมายเหตุใดๆ ตาม สามารถออดิตต่อสอนตามได้ที่ สถานที่ดังต่อ คร. ขั้ยงค์ รักพิชธรรมาก ในวันและเวลาราชการ หรือ โทรศัพท์ติดต่อได้ 089-1763372

โครงการวิจัยนี้ได้รับการพิจารณาอนุมัติจาก คณะกรรมการวิจัยและนวัตกรรมการวิจัยในศูนย์เทคโนโลยีทางการแพทย์ มหาวิทยาลัยหอดหินพิทิด สำนักงานอุปนายก สำนักงานอธิการบดีมหาวิทยาลัยหอดหินพิทิด ถนนนฤทธมณฑลสาย 4 ตำบลศาลาฯ อําเภอบุษราคัม จังหวัดคุรุภูมิ 73170 หมายเลขโทรศัพท์ 02-849-6224 , 6225 โทรศัพท์ 02-849-6224 หากท่านได้รับการปฏิเสธไม่ตรวจสอบตามที่ระบุไว้ ท่านสามารถติดต่อ ประธานกรรมการฯหรือผู้แทน ได้ตามสถานที่และหมายเลขโทรศัพท์ข้างต้น

ขอขอบพระคุณที่กรุณาสละเวลาในการตอบแบบสอบถาม

ขอแสดงความนับถือ

ดร. ชัยยงค์ รักขิตเวชสกุล



Self-Administered Questionnaire Participant Information Sheet version 10/02/2021

ข้อมูลเบื้องต้น

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

2. คุณประกอบอาชีพใด *

Mark only one oval.

- นักเรียน/ นักศึกษา
- ผู้เชี่ยวชาญด้านความปลอดภัย (Security specialist)
- นักพัฒนาซอฟต์แวร์ (Software developer)
- นักทดสอบโปรแกรมคอมพิวเตอร์ (Software tester)
- ผู้ดูแลและจัดการเครือข่ายคอมพิวเตอร์ (System administrator)
- นักวิเคราะห์ช่องโหว่ในซอฟต์แวร์ (Vulnerability analyst)
- นักวิจัย (Researcher)
- Other: _____

3. คุณมีประสบการณ์การเขียนโปรแกรมนานเท่าใด *

Mark only one oval.

- 6 เดือน - 1 ปี
- 1 - 2 ปี
- 3 - 4 ปี
- 5 - 6 ปี
- มากกว่า 6 ปี

4. คุณใช้ภาษาใดในการพัฒนาซอฟต์แวร์ *

Check all that apply.

- PHP
- Java
- JavaScript
- .NET
- Python
- Ruby
- Other: _____

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และช่องโหว่ด้านความปลอดภัยที่มีอยู่ในซอฟต์แวร์ของผู้ใช้งาน เช่น การติดต่อสื่อสาร กับเซิร์ฟเวอร์ หรือการเชื่อมต่อฐานข้อมูล เป็นต้น

5. คุณใช้การจัดการแพ็กเกจ (Package manager ecosystem) ใด *

Check all that apply.

- Composer
- Maven
- npm
- NuGet
- pip
- RubyGems

Other: _____

โปรดอ่าน ก่อนเริ่มทำ แบบสอบถาม

การซ่องโหว่ด้านความปลอดภัย (Security vulnerability) สามารถเกิดขึ้นได้จากหลายแหล่ง ไม่ว่าจะเกิดจากภายในโค้ดของโปรแกรมเอง หรือเกิดจากการเรียกใช้ไลบรารีของบุคคลที่สาม (Third-party dependency)

ปัจจุบันช่องโหว่ด้านความปลอดภัยที่เกิดจากการใช้ไลบรารีของบุคคลที่สามนั้น อาจเกิดขึ้นได้ จาไลบรารีของบุคคลที่สามโดยตรง (Direct vulnerability) หรือจากการที่ไลบรารีของบุคคลที่สามนั้นเรียกใช้ไลบรารีอีกตัว โดยเดาการณ์ตัวก่อภัยที่ทำให้เกิดเป็นสายโยงของไลบรารี (Chain of library dependency) หรือเรียกอีกอย่างหนึ่งว่าช่องโหว่ด้านความปลอดภัยทาง อ้อม (Indirect vulnerability)

แบบสอบถามนี้มุ่งเน้นความสนใจที่ช่องโหว่ด้านความปลอดภัย (Security vulnerability) ซึ่ง เกิดจากการใช้ไลบรารีของบุคคลที่สาม (Third-party dependency) ทั้งทางตรง (Direct vulnerability) และทางอ้อม (Indirect vulnerability) เพื่อศึกษาถึงความตระหนักรู้ (Awareness) ของนักพัฒนาซอฟต์แวร์

6. โปรดตอบคำถามเบื้องต้นเกี่ยวกับช่องโหว่ด้านความปลอดภัย (Security vulnerability) *

Mark only one oval per row.

	ไม่ สำคัญ	สำคัญ น้อย	สำคัญปาน กลาง	สำคัญ มาก	สำคัญ ที่สุด
คุณให้ความสำคัญกับช่องโหว่ด้านความปลอดภัยของซอฟต์แวร์ (Security vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่สาม (Third-party dependency) ในระดับใด	<input type="radio"/>				
คุณให้ความสำคัญกับ Security vulnerability ที่เกิดขึ้นจากสายโยงของ ไลบรารี (Chain of dependencies) ของ Third-party library ในระดับใด	<input type="radio"/>				

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

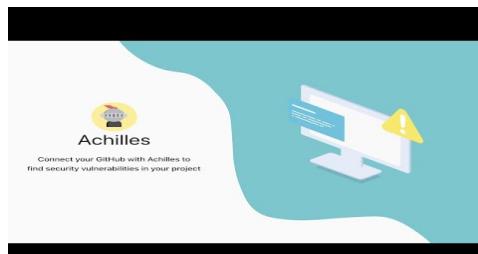
7. ในการจัดอันดับความสำคัญ การอัปเดต Third-party library คุณให้ความสำคัญกับสิ่งใด *

Mark only one oval per row.

	ไม่สำคัญ	สำคัญน้อย	สำคัญปานกลาง	สำคัญมาก	สำคัญที่สุด
จำนวนของข้อมูลช่องโหว่ด้านความปลอดภัยที่พบและปรากฏในฐานข้อมูล Common Vulnerabilities and Exposures (https://cve.mitre.org) หรือ GitHub Security Advisory (https://github.com/advisories)	<input type="radio"/>				
ระดับความรุนแรงของช่องโหว่ด้านความปลอดภัย	<input type="radio"/>				
ความเกี่ยวข้องของช่องโหว่ด้านความปลอดภัยกับฟังก์ชันหลักของซอฟต์แวร์	<input type="radio"/>				
ความท่าหะระหว่างเวอร์ชันที่มีช่องโหว่ (Vulnerable version) เทียบกับเวอร์ชันที่ได้รับการแก้ไข (First patch version)	<input type="radio"/>				
เป็นช่องโหว่ด้านความปลอดภัยที่เพียงค้นพบ (ความใหม่ของช่องโหว่ฯ)	<input type="radio"/>				

คำถามเกี่ยวกับการแสดงผลกราฟ

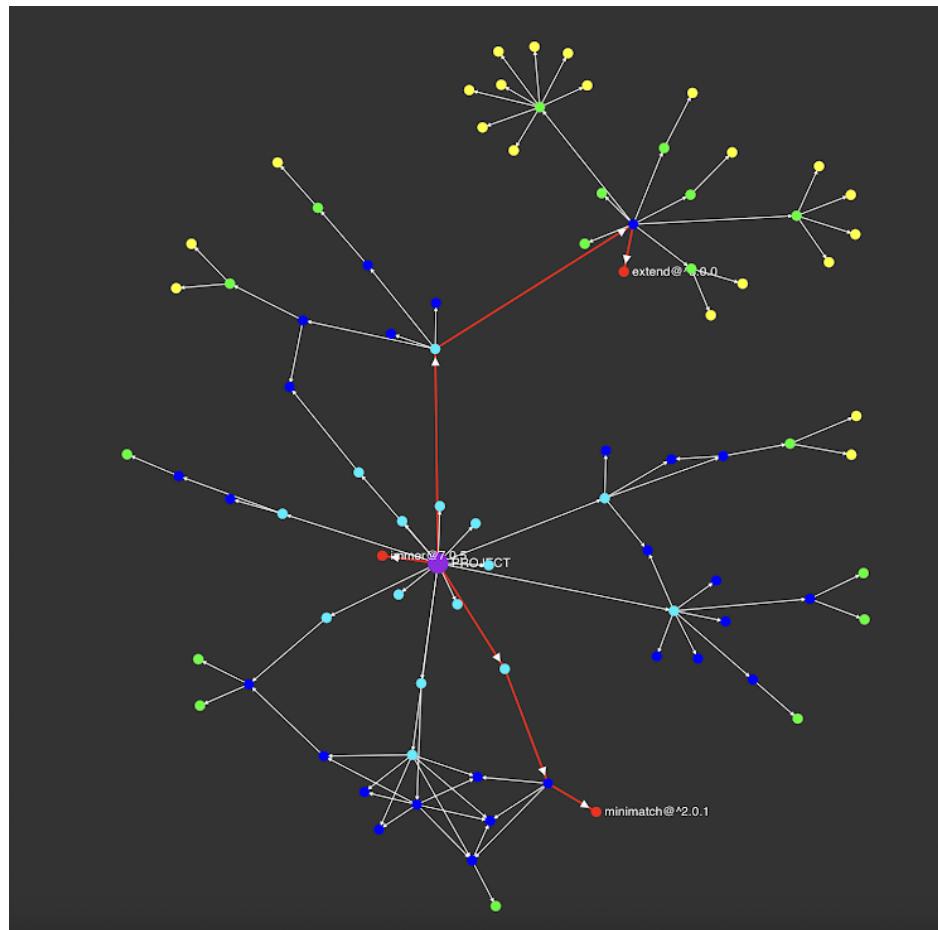
โปรดรับชมวิดีโอสาธิตการใช้เครื่องมือ Achilles และตอบคำถามด้านล่าง (วิดีโอขนาดเต็มสามารถดูได้ที่ https://youtu.be/BRMhT_vmu_0) คุณสามารถทดลองใช้งาน Achilles Tool ได้ที่ <https://achilles-sp.azurewebsites.net/>



http://youtube.com/watch?v=BRMhT_vmu_0

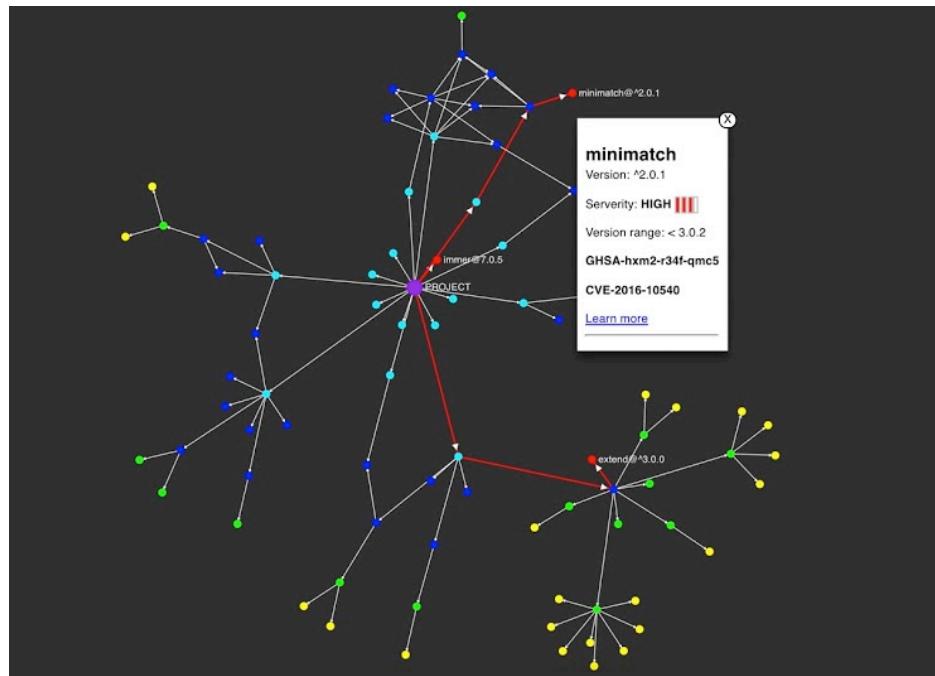
4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

ภาพตัวอย่างการแสดงผลช่องโหว่ด้านความปลอดภัย (Vulnerable dependency visualization)
(ภาพขนาดเต็มสามารถดูได้ที่ <http://bit.ly/achilles-visualization>)



4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

ภาพตัวอย่างการแสดงผลช่องโหว่ด้านความปลอดภัย (Vulnerable dependency visualization)
พร้อมคำอธิบาย (ภาพขนาดเต็มสามารถดูได้ที่ <http://bit.ly/achilles-tooltip>)



8. คุณเข้าใจการแสดงผลของช่องโหว่ด้านความปลอดภัยจากการแสดงผลด้วยกราฟด้านบน (Vulnerability dependency visualization) ในระดับใด *

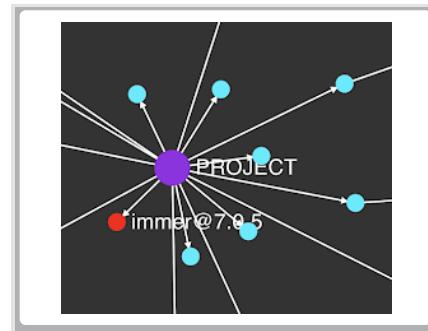
Mark only one oval.

1	2	3	4	5	
ไม่เข้าใจ	<input type="radio"/> เข้าใจทั้งหมด				

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

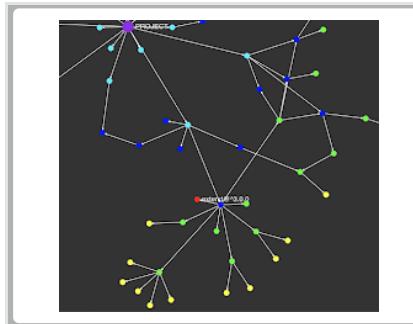
9. ส่วนใดที่คุณคิดว่าข้างต้นการคำนวณเพิ่มเติม

Check all that apply.

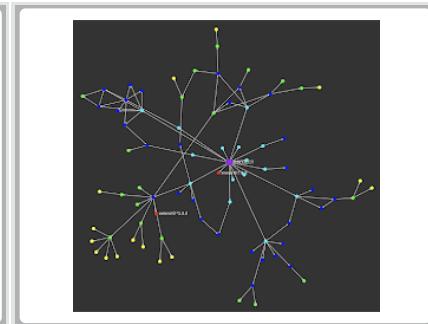


ไม่ต้องการคำนวณเพิ่มเติม

ช่องโหว่ด้านความปลอดภัยทางตรง
(Direct vulnerability)

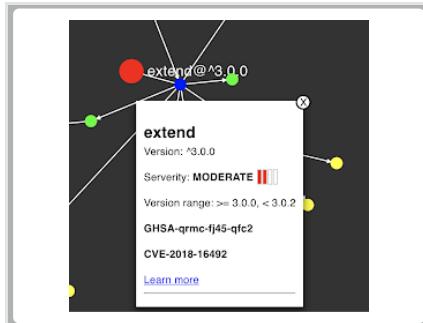


ช่องโหว่ด้านความปลอดภัยทางอ้อม
(Indirect vulnerability)



สิ่งไหนแยกกันไม่ชัดเจน

Other: _____



คำอธิบายเพิ่มเติม (Tooltips)

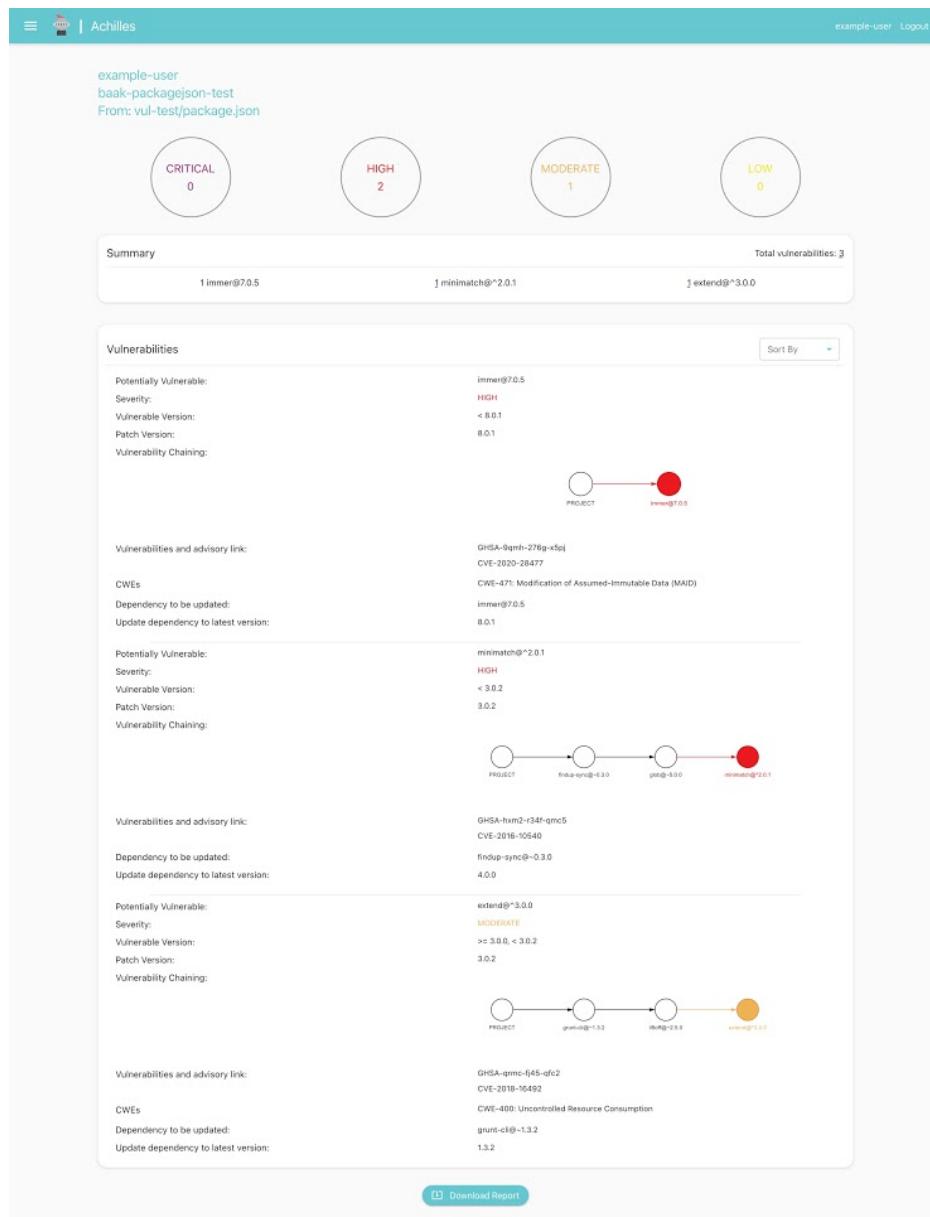
4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

10. คุณมีคำแนะนำเพิ่มเติมอื่นสำหรับการแสดงผล Vulnerability dependency visualization หรือไม่

แบบสอบถามเกี่ยวกับรายงานช่องโหว่ด้านความปลอดภัย

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ

โปรดศึกษาภาพด้วยว่ารายงานช่องโหว่ด้านความปลอดภัยของ Achilles แล้วตอบคำถามด้านล่าง (ภาพขนาดเต็มสามารถดูได้ที่ <http://bit.ly/achilles-report>)



4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ

11. ระดับของความเข้าใจเมื่อคุณดูรายงานช่องโหว่ด้านความปลอดภัยด้านบน *

Mark only one oval.

1 2 3 4 5

ไม่เข้าใจเลย เข้าใจทั้งหมด

12. คำแนะนำเพิ่มเติมสำหรับรายงานฯ

การวางแผนหลังการใช้เครื่องมือ Achilles

13. จากการรับชมวิดีโอสาธิตและตัวอย่างรายงานของเครื่องมือตรวจสอบและรายงานช่องโหว่ด้านความปลอดภัย Achilles คุณคิดว่าเครื่องมือดังกล่าวมีประโยชน์ต่องานคุณมากน้อยเพียงใด *

Mark only one oval.

1 2 3 4 5

น้อยที่สุด หากมากที่สุด

14. คุณจะตัดสินใจอย่างไร หากใช้เครื่องมือ Achilles และพบว่า โปรเจ็คของคุณมีช่องโหว่ด้านความปลอดภัย *

Mark only one oval.

วางแผนอัปเดตไลบรารีที่มีช่องโหว่

เลือกที่จะไม่อัปเดตไลบรารีที่มีช่องโหว่

Other: _____

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

15. โปรดอธิบายเพิ่มเติมว่าทำไม่คุณถึงเลือกที่จะไม่อัปเดตไลบรารีที่มีช่องโหว่

Check all that apply.

- หากอัปเดตไลบรารีที่มีช่องโหว่ว่าอาจทำให้โค้ดในโปรเจ็คเกิดความขัดแย้งได้ (Conflict)
- การแก้ไขช่องโหว่ด้านความปลอดภัยไม่ใช่งานที่มีความสำคัญอันดับแรก
- ไม่ทราบว่าจะต้องแก้ไขไลบรารีที่มีช่องโหว่อย่างไร

Other:

**Survey on
security
vulnerability
from third-
party
library and
detection
tool**

This survey is part of the ITCS492 Senior Project II in the Faculty of Information and Communication Technology, Mahidol University. We developed an Achilles tool for finding and visualizing security vulnerability for npm dependency. This survey aims to perceive the awareness of npm dependencies security vulnerabilities issues and ask for suggestions and feedback for the Achilles tool.

Since the survey consists of multiple parts, thus we ask you to consider answering based on your utmost understanding. Your survey answers will be sent to a link at Google Forms and Google Sheets where data will be stored in a password-protected electronic format. Google Forms and Google Sheets do not collect identifying information such as your name, email address, or IP address. Therefore, your responses will remain anonymous. No one will be able to identify you or your answers, and no one will know whether you participated in the survey. The survey responses will be used to analyze the overview of this research project. It will not have any effect on your career or your organization. Your participation in this survey is voluntary. You may refuse to take part in the research or exit the survey at any time without penalty. You are free to decline to answer any particular question you do not wish to answer for any reason.

If you have questions at any time about the study or the procedures, you may contact my research supervisor, Dr. Chaiyong Ragkhitwetsagul, at the Faculty of Information and Communication Technology on official days and hours or via phone at (02) 441-0909 or via email at chaiyong.rag@mahidol.ac.th.

This research study has been approved by Mahidol University Center of Ethical Reinforcement for Research No. MU-CIRB 2021/085.1103 at Office of the President Mahidol University, 2nd floor, 999 Phuttamonthon 4 Road, Salaya, Nakhon Pathom 73170, Thailand or via phone at (02) 849-6220, (02) 849-6223 If you feel you have not been treated according to the descriptions in this form, or that your rights as a participant in research have not been honored during the course of this project, or you have any questions, concerns, or complaints that you wish to address to someone other than the investigator, you may contact MUCERif Administrative at the address and phone mentioned above.

Thank you for dedicating your time to answering this survey.

Miss Vipawan Jarukitpipat

Miss Wachirayana Wanprasert

Mr. Klinton Chhun

Dr. Chaiyong Ragkhitwetsagul (Research advisors)

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) ฯ...

Self-Administered Questionnaire Participant Information Sheet

MU-CIRB	คณะกรรมการวิจัยและพัฒนา มหาวิทยาลัยมหิดล เอกสารแจ้งผู้เข้าร่วมวิจัย โดยการตอบแบบสอบถาม (Self-Administered Questionnaire Participant Information Sheet)	แฟ้มวันที่ 10/07/2020 หน้าที่ 1 จาก 1 หน้า
เอกสารแจ้งผู้เข้าร่วมวิจัย โดยการตอบแบบสอบถาม (Self-Administered Questionnaire Participant Information Sheet)		
<input checked="" type="checkbox"/> ต้นฉบับ	<input type="checkbox"/> การปรับเปลี่ยนครั้งที่.....	วันที่ 11/.../2564

เรียน ผู้ดูแลแบบสอบถามทุกท่าน

ด้วย กระหม่ำ ดร. ชัยยงค์ รักขิตเวชสกุล อาจารย์ประจำคณะเทคโนโลยีสารสนเทศและการสื่อสาร มหาวิทยาลัยมหิดล มีความประสงค์ที่จะดำเนินการวิจัย เรื่อง อดีต: เครื่องมือเพื่อการวิเคราะห์ช่องโหว่ด้านความปลอดภัยของบุคคลที่ 3 (Third-party library) ซึ่งประเมินว่ามีความเสี่ยงต่อ Achille และความเสี่ยงของ Achille ให้อธิบาย

ท่านได้รับเชิญให้เข้าร่วมการวิจัยนี้ เพราะท่านเป็นนักพัฒนาซอฟต์แวร์ที่มีลักษณะเหมาะสมกับโครงการวิจัยนี้ ในการนี้ ผู้จัดนิยามจำเป็นต้องเก็บรวบรวมข้อมูลโดยใช้แบบสอบถามเรื่อง “แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และเรื่องมือในการตรวจสอบ” ซึ่งประกอบด้วยค่าถูกต้อง 5 ตัววน จำนวน 15 ข้อ ใช้เวลาในการตอบประมาณ 15 - 20 นาที ผู้ที่ร่วมของรับแบบสอบถามคืนโดยการส่ง อีเมลมาทั้งที่มีสิ่งที่

เพื่อจัดแบบสอบถามตามประกอบด้วยค่าถูกต้อง จึงขอความกรุณาให้ท่านพิจารณาตอบตามความรู้สึกของท่านให้มากที่สุด โดยขออนุญาตและคำตกลงทั้งหมดจะถูกปฏิบัติเป็นความลับ และจะนำมายังการวิเคราะห์ผลการศึกษาครั้นนี้โดยอ่อนโยน เป็นภาพรวมของการวิจัยท่านนี้ จึงไม่มีผลกระทบใด ๆ ต่อค่าตอบแทนหรือหน่วยงานของผู้ตอบ เมื่อจากไม่สามารถนำมายังค้น เจาะของทางผู้ตอบได้ ท่านมีสิทธิที่จะไม่ตอบค่าถูกต้องให้ข้อเท็จจริง หากท่านไม่สามารถใช้เครื่องดัดที่จะตอบค่าถูกต้องนั้น หรือไม่ตอบแบบสอบถามทั้งหมดเดียวกันได้ โดยไม่มีผลกระทบต่อการปฎิบัติงานใด ๆ ของท่าน ท่านมีสิทธิที่จะไม่เข้าร่วมการวิจัยที่ได้โดยไม่ต้องแจ้งเหตุผล

หากผู้เข้าร่วมวิจัยมีข้อสงสัยเกี่ยวกับการวิจัยหรือแบบสอบถาม สามารถติดต่อสอบถามได้ที่ สถานที่ติดต่อ ดร. ชัยยงค์ รักขิตเวชสกุล ในวันและเวลาราชการ หรือ โทรศัพท์ที่ติดต่อได้ 089-1763372

โครงการจัดขึ้นได้รับการพิจารณา/approval ของ คณะกรรมการวิจัยในคนของมหาวิทยาลัยมหิดล สำนักงานอัญเชิญ สำนักงานอธิการบดีมหาวิทยาลัยมหิดล ถนนพุทธมณฑล สาย 4 ตำบลศาลายา อำเภอพุทธมณฑล จังหวัดนครปฐม 73170 หมายเลขโทรศัพท์ 02-849-6224 , 6225 โทรสาร 02-849-6224 หากท่านได้รับการปฏิบัติไม่ตรงตามที่ระบุไว้ ท่านสามารถติดต่อประธานกรรมการหรือผู้แทน ได้ตามสถานที่และหมายเลขโทรศัพท์ข้างต้น

ขอขอบพระคุณที่กรุณาสละเวลาในการตอบแบบสอบถาม

ขอแสดงความนับถือ

ดร. ชัยยงค์ รักขิตเวชสกุล



Self-Administered Questionnaire Participant Information Sheet version 10/02/2021

Basic Information

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

16. What is your occupation? *

Mark only one oval.

- Students
- Security specialist
- Software developer
- Software tester
- System administrator
- Vulnerability analyst
- Researcher
- Other: _____

17. How long is your experience in programming? *

Mark only one oval.

- 6 months - 1 year
- 1 - 2 years
- 3 - 4 years
- 5 - 6 years
- More than 6 years

18. What programming language do you use? *

Check all that apply.

- PHP
- Java
- JavaScript
- .NET
- Python
- Ruby

Other: _____

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

19. What package manager ecosystem do you use? *

Check all that apply.

- Composer
- Maven
- npm
- NuGet
- pip
- RubyGems

Other: _____

Please read
before
starting the
survey

Security vulnerability can occur from several sources including vulnerability from source code and from third-party dependency.

Vulnerability from third-party dependency can be direct vulnerability and indirect vulnerability which occur from the usage of other libraries causing a chain of dependency.

This survey focuses on security vulnerability from third-party dependency both direct and indirect vulnerability. We would like to study the awareness of software developers regarding this vulnerability.

20. Please answer the following basic questions about software vulnerabilities. *

Mark only one oval per row.

	Not concerned	Slightly concerned	Moderately concerned	Very concerned	Extremely concerned
How much are you concerned about security vulnerabilities from third-party dependencies in your software project?	<input type="radio"/>				
How much are you concerned of security vulnerabilities caused by chain of dependencies in your software project (indirect vulnerabilities)	<input type="radio"/>				

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

21. How do you prioritize each factor when you decide to update the Third-party library? *

Mark only one oval per row.

	Not important	Slightly important	Moderately important	Very Important	Extremely important
The number of vulnerabilities that found in CVE (https://cve.mitre.org) or GitHub Security Advisory (https://github.com/advisories)	<input type="radio"/>				
Severity of Vulnerability	<input type="radio"/>				
Relevancy to the business requirement	<input type="radio"/>				
The gap between your vulnerable library version and first patch version to fix that vulnerability is large	<input type="radio"/>				
Recency of vulnerability	<input type="radio"/>				

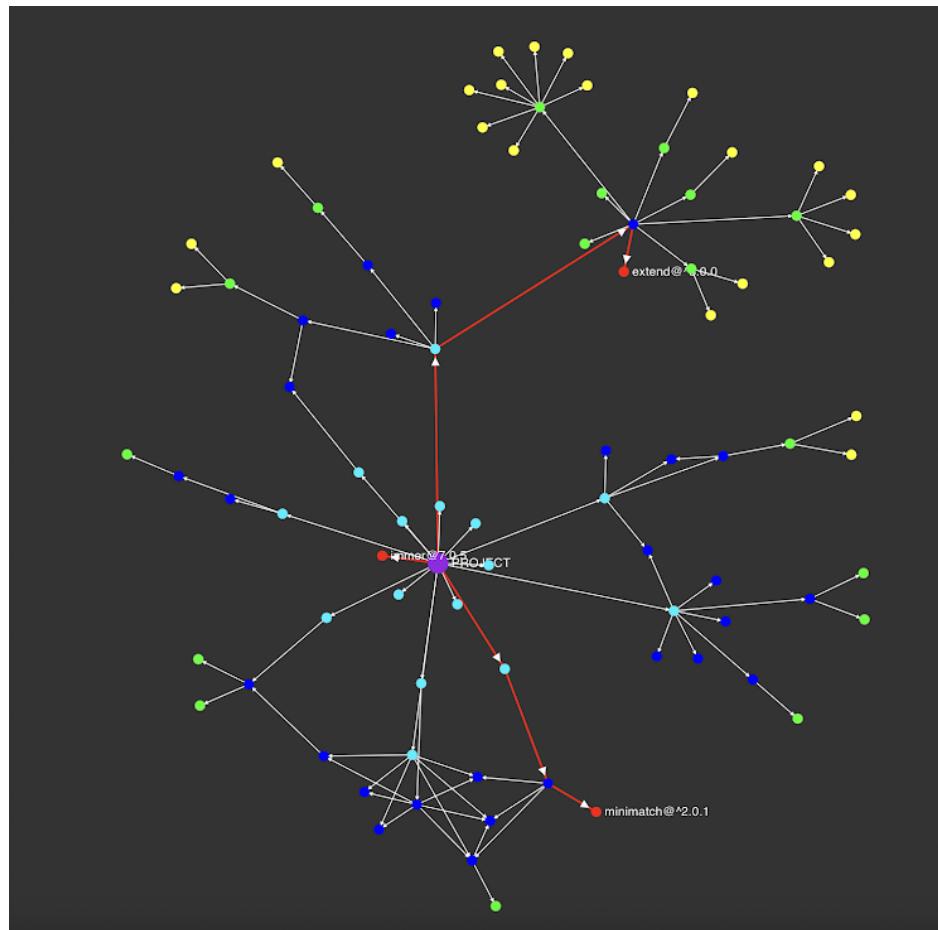
Questions for visualization

Please watch the Achilles tool demonstration and answer the following questions (Link for the video: https://youtu.be/BRMhT_vmu_0) We would like to invite you to use Achilles tool at <https://achilles-sp.azurewebsites.net>.



4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

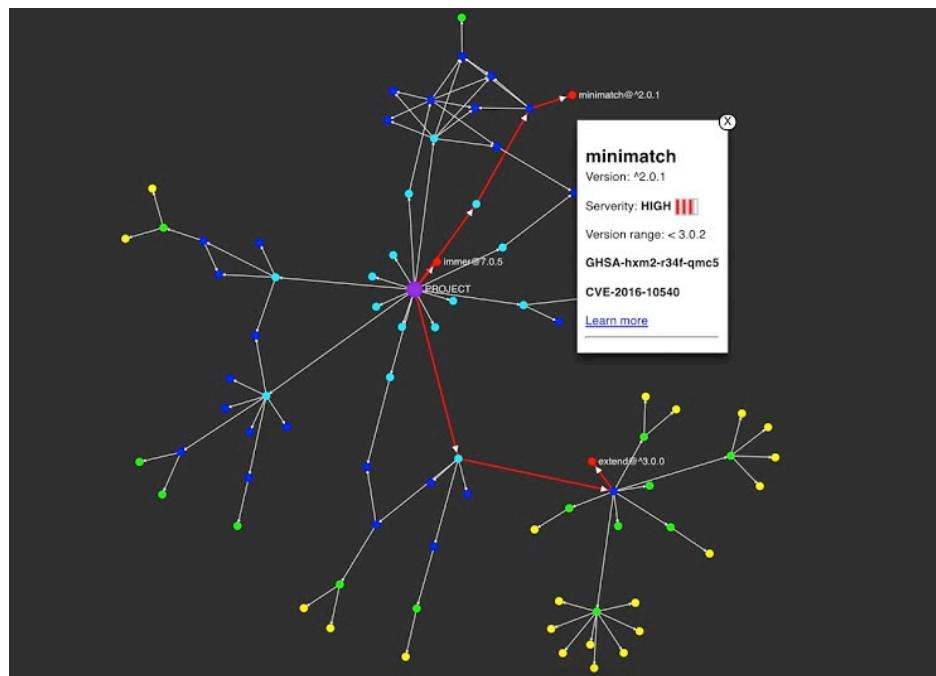
Visualization of vulnerable dependencies (See full image: <http://bit.ly/achilles-visualization>)



4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

Visualization of vulnerable dependencies with tooltips (See full image:

<http://bit.ly/achilles-tooltip>)



22. How much do you understand this visualization? *

Mark only one oval.

1 2 3 4 5

Do not understand Understand completely

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

23. Are there any parts that you would like more explanation?

Check all that apply.

None

Direct vulnerability

Indirect vulnerability

Color of the nodes in each level

Other:

Tooltips

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

24. Do you have any suggestion for improving the vulnerability dependency visualization?

Questions for security vulnerability report

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ

Please see the security vulnerability report and answer the following questions (See full picture: <http://bit.ly/achilles-report>)

Achilles

example-user
baak-packagejson-test
From: vul-test/package.json

CRITICAL 0 **HIGH 2** **MODERATE 1** **LOW 0**

Summary Total vulnerabilities: 3

immer@7.0.5 minimatch@^2.0.1 extend@^3.0.0

Vulnerabilities Sort By

Potentially Vulnerable:
Severity: HIGH
Vulnerable Version: < 8.0.1
Patch Version: 8.0.1
Vulnerability Chaining:

Vulnerabilities and advisory link:
GHSA-9qmh-276g-x5pj
CVE-2020-26477

CWEs
Dependency to be updated:
Update dependency to latest version:

Potentially Vulnerable:
Severity: HIGH
Vulnerable Version: < 3.0.2
Patch Version: 3.0.2
Vulnerability Chaining:

Vulnerabilities and advisory link:
GHSA-hzm2-r34f-qmc5
CVE-2018-10540

Dependency to be updated:
Update dependency to latest version:

Potentially Vulnerable:
Severity: MODERATE
Vulnerable Version: >= 3.0.0, < 3.0.2
Patch Version: 3.0.2
Vulnerability Chaining:

Vulnerabilities and advisory link:
GHSA-qmc-fj45-afc2
CVE-2018-16492

CWEs
Dependency to be updated:
Update dependency to latest version:

Download Report

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

25. What is your level of understanding when reading the vulnerability report *

Mark only one oval.

1 2 3 4 5

Do not understand Understand completely

26. Suggestion for the security vulnerability report

Questions for the plan after using the Achilles tool

27. After watching the Achilles tool demonstration and vulnerability report, how do you find this vulnerability visualization and report useful? *

Mark only one oval.

1 2 3 4 5

Least useful Most useful

28. What would be your decision if you use the Achilles tool and find that your project has security vulnerabilities. *

Mark only one oval.

Update vulnerable dependencies

Do not update vulnerable dependencies

Other: _____

4/18/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

29. Please provide some reasons why you would not update vulnerable dependencies

Check all that apply.

- Conflict in the code might occur after updating the library
- Updating vulnerable dependencies is not the first priority
- Do not know how to update the vulnerable dependencies

Other:

This content is neither created nor endorsed by Google.

Google Forms

APPENDIX C

ONLINE SURVEY RESULT

4/16/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และเครื่องมือในการตรวจจับ

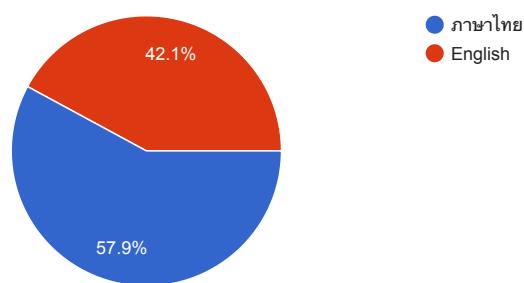
แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และเครื่องมือในการตรวจจับ

19 responses

[Publish analytics](#)

กรุณาเลือกภาษาของแบบสอบถาม

19 responses



แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และเครื่องมือในการตรวจจับ

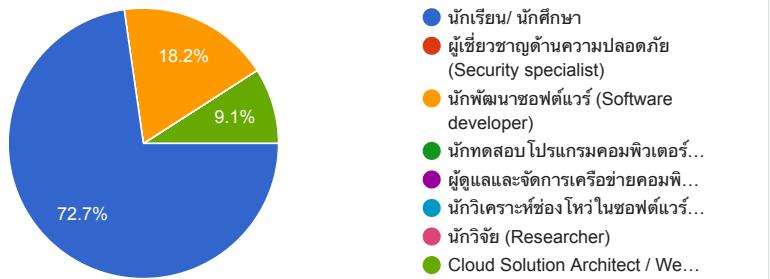
ข้อมูลเบื้องต้น



4/16/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

คุณประกอบอาชีพได้

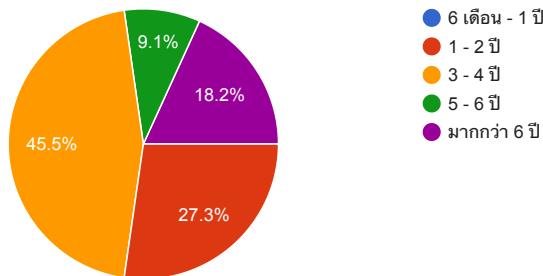
11 responses



- นักเรียน/นักศึกษา
- ผู้เชี่ยวชาญด้านความปลอดภัย (Security specialist)
- นักพัฒนาซอฟต์แวร์ (Software developer)
- นักทดสอบโปรแกรมคอมพิวเตอร์...
- ผู้ดูแลและดักการเครือข่ายคอมพิวเตอร์...
- นักวิเคราะห์ช่องโหว่ในซอฟต์แวร์...
- นักวิจัย (Researcher)
- Cloud Solution Architect / We...

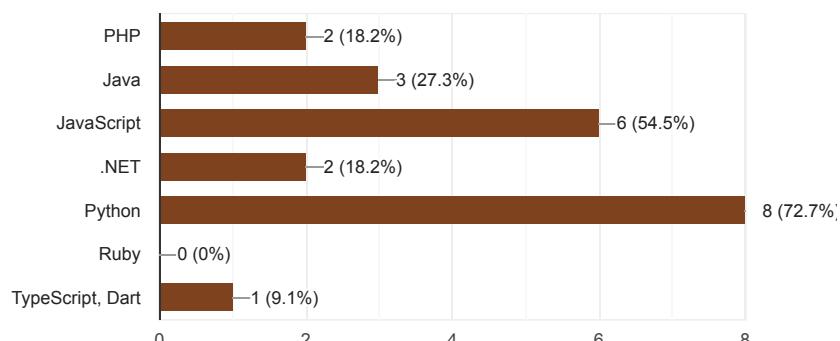
คุณมีประสบการณ์การเขียนโปรแกรมนานเท่าใด

11 responses



คุณใช้ภาษาใดในการพัฒนาซอฟต์แวร์

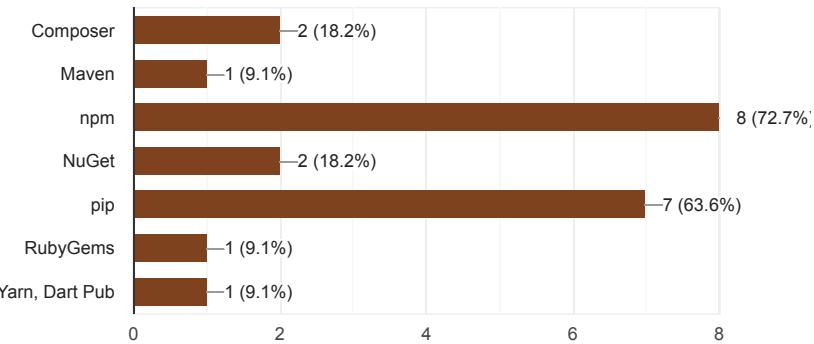
11 responses



4/16/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

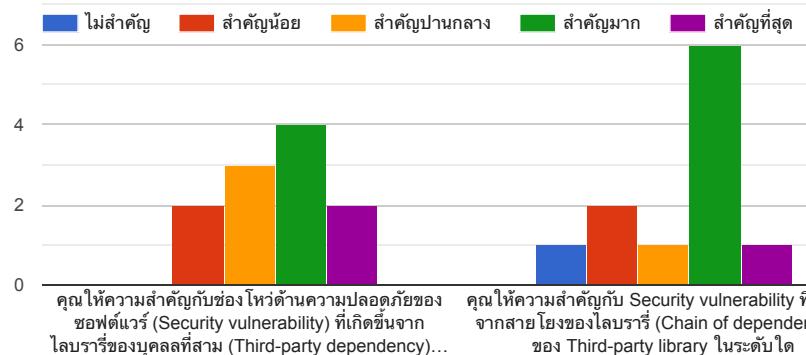
คุณใช้การจัดการแพ็คเกจ (Package manager ecosystem) ใด

11 responses



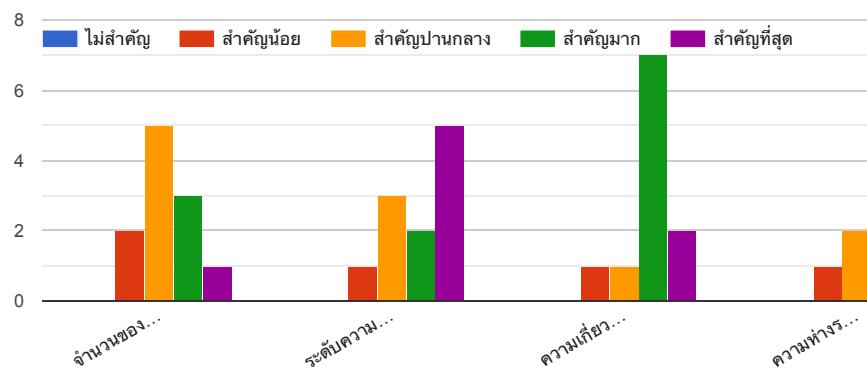
โปรดอ่านก่อนเริ่มทำแบบสอบถาม

โปรดตอบคำถามเบื้องต้นเกี่ยวกับช่องโหว่ด้านความปลอดภัย (Security vulnerability)



4/16/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

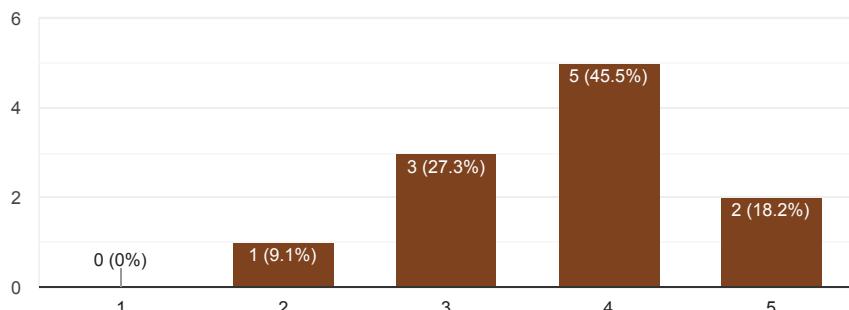
ในการจัดอันดับความสำคัญ การอัปเดต Third-party library คุณให้ความสำคัญกับสิ่งใด



คำถามเกี่ยวกับการแสดงผลกราฟ

คุณเข้าใจการแสดงผลของช่องโหว่ด้านความปลอดภัยจากการแสดงผลด้วยกราฟต้นแบบ (Vulnerability dependency visualization) ในระดับใด

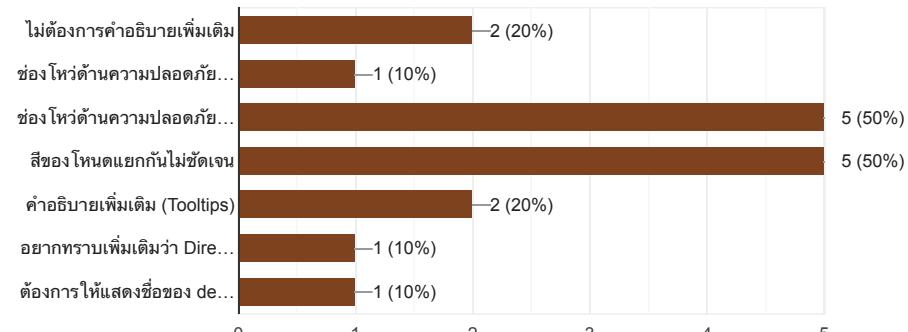
11 responses



4/16/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

ส่วน ได้ที่คุณคิดว่า ยังต้องการคำอธิบายเพิ่มเติม

10 responses



คุณมีคำแนะนำเพิ่มเติมอื่นสำหรับการแสดงผล Vulnerability dependency visualization หรือไม่

2 responses

ความหมายของเลข level ของ indirect dependency ยังไม่ชัดเจน

- node ควรมีขนาดใหญ่ขึ้น
- ควรปรับสีของ node ให้สอดคล้องกับ บี
- อาจจะมี option ให้เลือกในการ display ว่าอยากให้ vulnerability dependency show ที่ node หรือที่ path (ให้ user เลือกได้ว่าจะ highlight ที่ไหน)

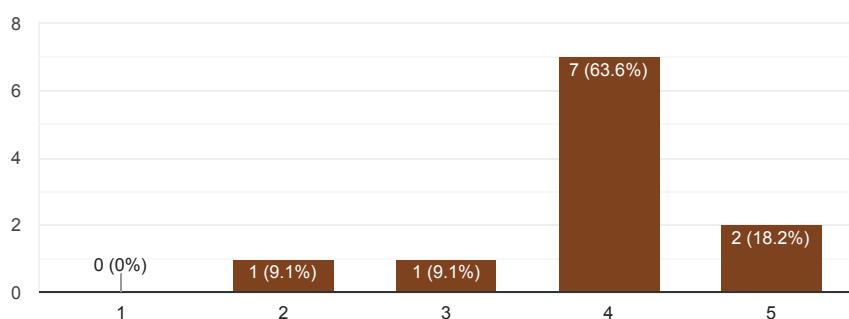
แบบสอบถามเกี่ยวกับรายงานช่องโหว่ด้านความปลอดภัย



4/16/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

ระดับของความเข้าใจเมื่อคุณดูรายงานช่องโหว่ด้านความปลอดภัยด้านบน

11 responses



คำแนะนำเพิ่มเติมสำหรับรายงานฯ

3 responses

ถ้าให้เดิน อาจจะมีลิงค์ไปที่ version ที่ไม่มี vulnerability เพื่อให้อัพเดตได้เลย อาจจะสะดวกกับ dev

อยากให้มีลิงค์พาไปหน้า NPM ของ dependency

อยากให้แยก severity เป็น section มากกว่ารวมกันใน section

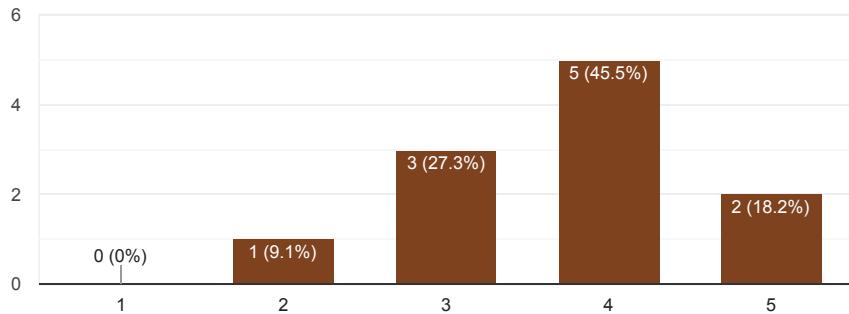
การวางแผนหลังการใช้เครื่องมือ Achilles



4/16/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

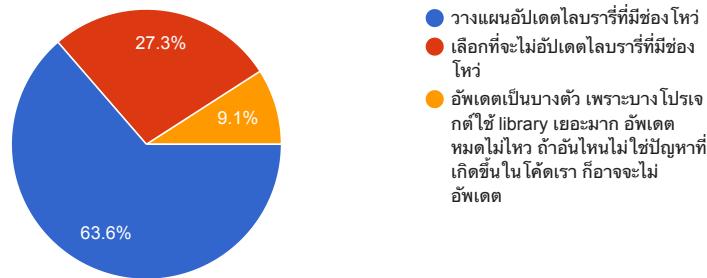
จากการรับชมวิดีโอสาธิตและตัวอย่างรายงานของเครื่องมือตรวจสอบและรายงานช่องโหว่ด้านความปลอดภัย Achilles คุณคิดว่าเครื่องมือดังกล่าวมีประโยชน์ต่องานคุณมากน้อยเพียงใด

11 responses



คุณจะตัดสินใจอย่างไร หากใช้เครื่องมือ Achilles และพบว่า โปรเจ็คของคุณมีช่องโหว่ด้านความปลอดภัย

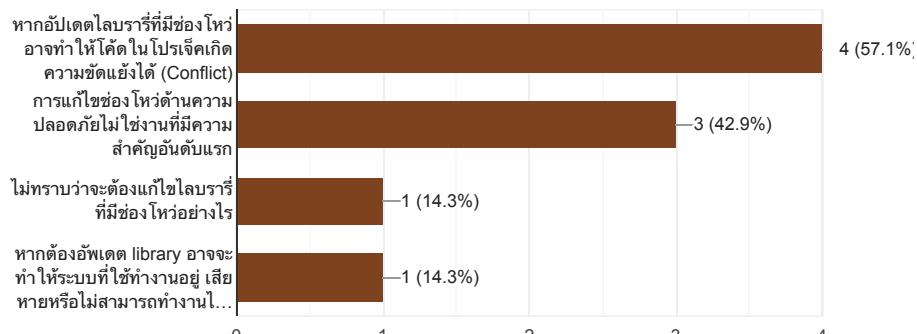
11 responses



4/16/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ

โปรดอธิบายเพิ่มเติมว่าทำไม่คุณถึงเลือกที่จะไม่อัปเดตไลบรารีที่มีช่องโหว่

7 responses

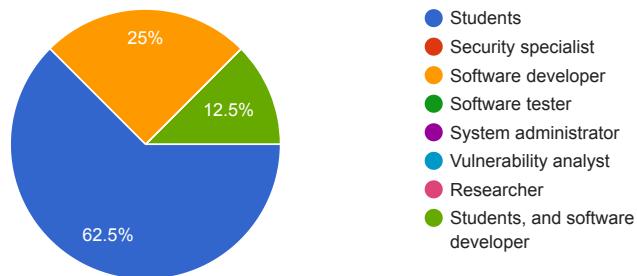


Survey on security vulnerability from third-party library and detection tool

Basic Information

What is your occupation?

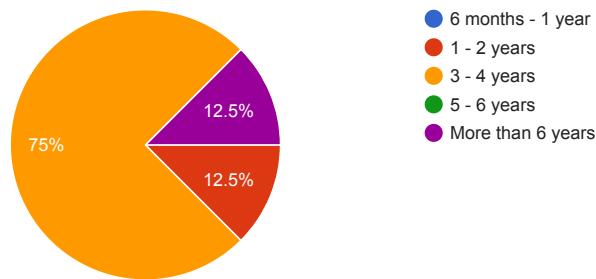
8 responses



4/16/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

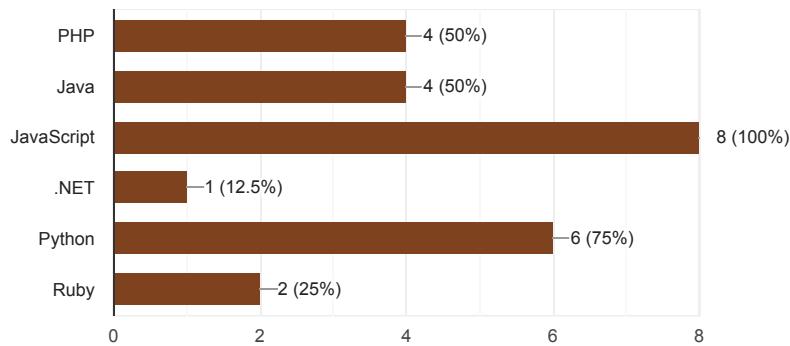
How long is your experience in programming?

8 responses



What programming language do you use?

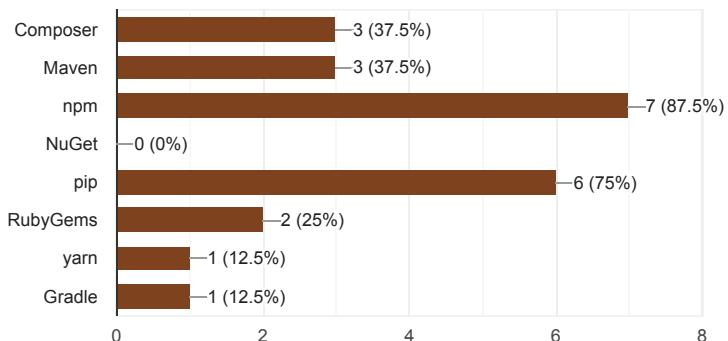
8 responses



4/16/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

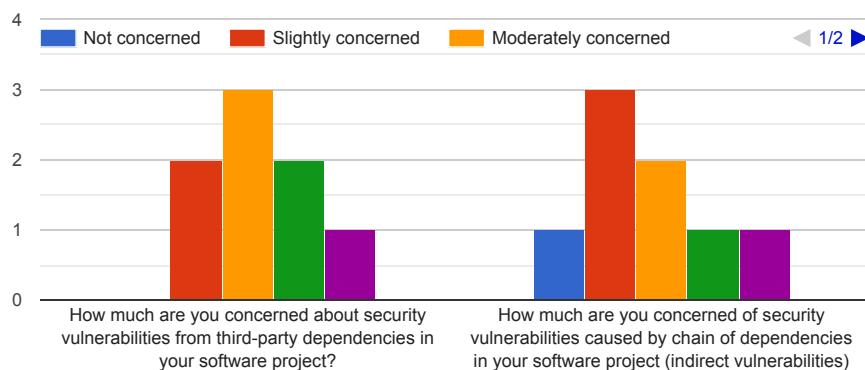
What package manager ecosystem do you use?

8 responses



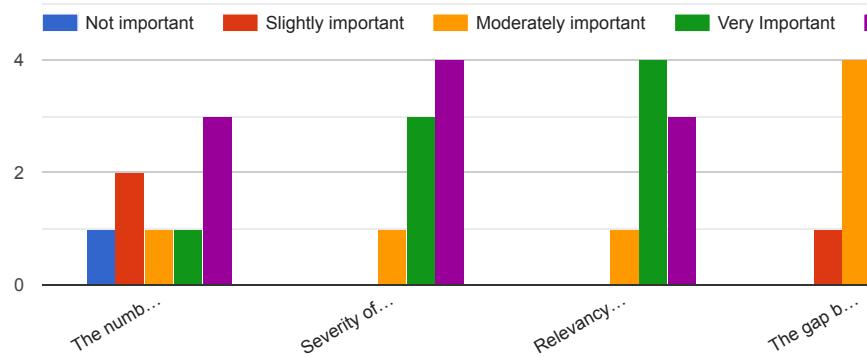
Please read before starting the survey

Please answer the following basic questions about software vulnerabilities.



4/16/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ

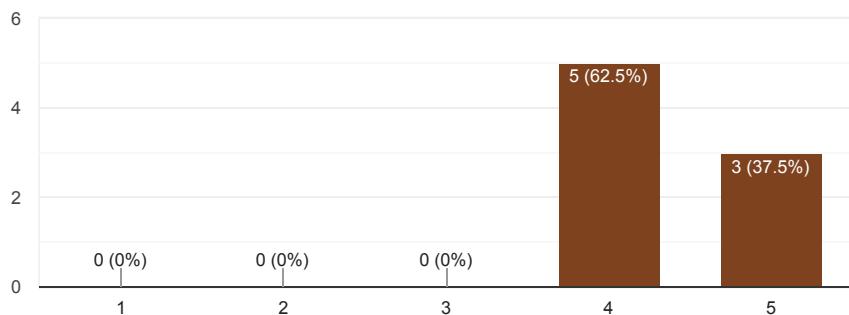
How do you prioritize each factor when you decide to update the Third-party library?



Questions for visualization

How much do you understand this visualization?

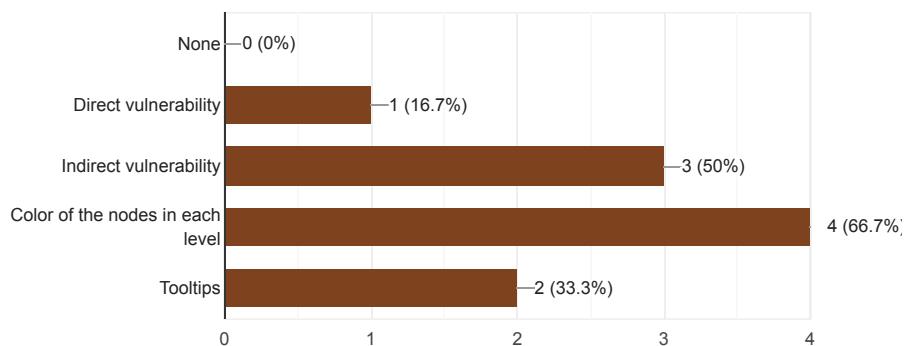
8 responses



4/16/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

Are there any parts that you would like more explanation?

6 responses



Do you have any suggestion for improving the vulnerability dependency visualization?

3 responses

as the graph grew bigger, it got really slow. It was slow to the point that moving is hard.

May add label of each colour beside the visualization.

In the tool tips, it would be more insightful if there is a short description of what type the vulnerability is.

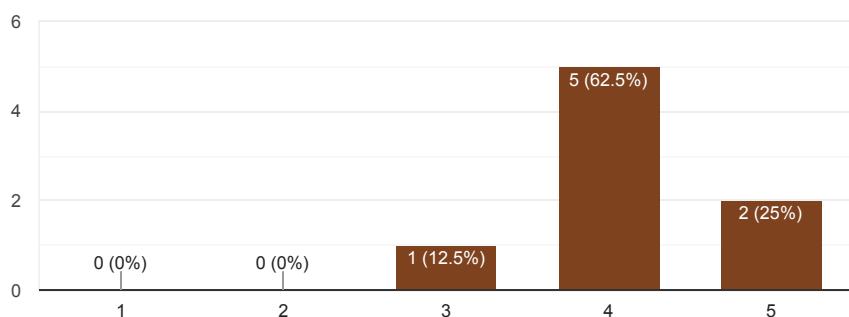
Questions for security vulnerability report



4/16/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

What is your level of understanding when reading the vulnerability report

8 responses



Suggestion for the security vulnerability report

2 responses

Improve the design like using more color visualization to show how important is. ex. CRITICAL -> using dark purple bg color / bar over the section of that library.

In the summary section, I think that it would be easier for me to read if the data is displayed in a table view containing columns such as the vulnerable packages, dependency, the level of severity, etc.

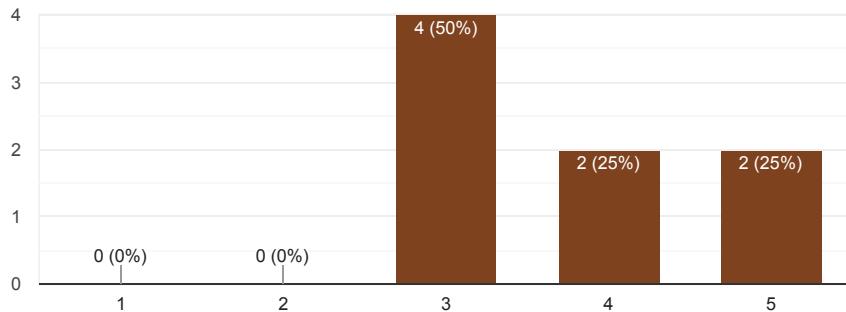
Questions for the plan after using the Achilles tool



4/16/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

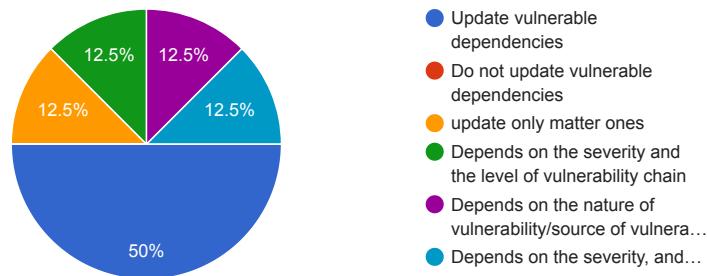
After watching the Achilles tool demonstration and vulnerability report, how do you find this vulnerability visualization and report useful?

8 responses



What would be your decision if you use the Achilles tool and find that your project has security vulnerabilities.

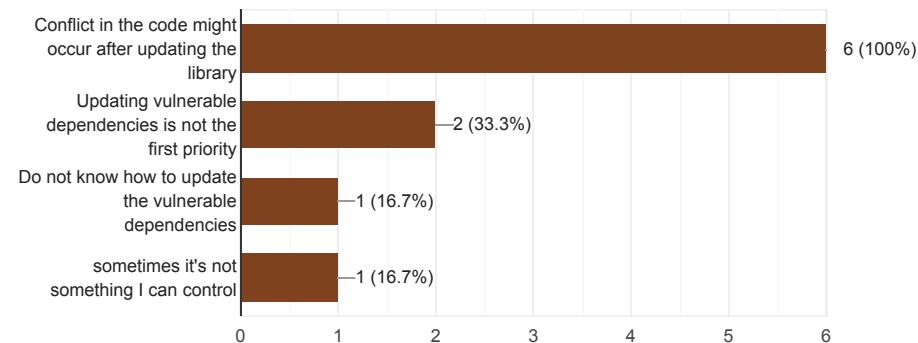
8 responses



4/16/2021 แบบสอบถามเกี่ยวกับช่องโหว่ด้านความปลอดภัย ของซอฟต์แวร์ (Security Vulnerability) ที่เกิดขึ้นจาก ไลบรารีของบุคคลที่ 3 (Third-party library) และ...

Please provide some reasons why you would not update vulnerable dependencies

6 responses



This content is neither created nor endorsed by Google. [Report Abuse](#) - [Terms of Service](#) - [Privacy Policy](#)

Google Forms



APPENDIX D

USER STUDY MATERIALS

Next page

<สิ่ง consent form>
<http://bit.ly/achilles-consent>

Hello, my name is <your name>. I'm a researcher in the project "Achilles: Automated tool for detecting and visualizing npm dependency vulnerabilities"

I would like to thank you for joining our user study today. This user study is a part of our senior project which studies security vulnerabilities in npm projects. The project also involves the tools that detect and report security vulnerabilities in such projects.

Now, we would like to ask you to read the participant information form and the consent forms that we have sent to you in the chat. Once you have finished reading and agree to join the study, please let us know by saying "I agree to join the study".

Regarding the study, you will be asked to use a tool for security vulnerability check in npm projects on the projects that we prepared for you. The tool that you'll use today is <achilles>

Before starting the study, we would like to ask you some questions.

<demographic questions>

1. How long have you been using npm?
2. What do you use npm for?
3. How often do you check security vulnerability in your project? And how?
4. Have you used this tool before? If yes - for how long?
5. Do you know Indirect Dependencies? How much you give importance to them? (สำคัญมาก
เป็นส่วนต่อหน้าที่ indirect dependencies)

<ส่ง Document>
เปิดเมลเดรีบยนส่ง verification code

May we ask you to share your screen please?

Next, we'll give you a mock task to try. You will use the two tools (dependabot report and achilles) and answer the questions.

We have the guide videos on how to use dependabot and achilles and also the mock repository here.

Video: <https://youtu.be/4ppCoCDtFeo>

[อธิบาย] Dependabot Report จะให้หัวข้อ Direct และ Indirect vulnerabilities ถ้าอยากรายงานก็สามารถดาวน์โหลด package.json file ได้เลย

For now, we have 2 questions, you can let us know when you have finished answering the questions. If you have any inquiries while answering the questions, please feel free to let us know.

Next, we will move to the actual user study.

For this user study, we have 2 test cases for you.
<บอกให้ participants เปิด GitHub overview ดู repo>

There are 2 repositories for the 2 test cases, which are Test 1 and Test 2.

<บอกให้เปิด Document หน้าที่ 5>

The task that you have to do is prioritizing the updates of the vulnerabilities. In each test, you will see the vulnerability report of dependabot in GitHub. You can take notes while seeing the report. After that, you'll use <npm audit> and prioritize the updates of the vulnerabilities again.

Before we proceed, do you allow us to record the video from further analysis?

Test1: [index.js — test1 — code-server \(cdr.co\)](#)

Test2: [yarn.lock — test2 — code-server \(cdr.co\)](#)

Test 1

		Severity	Complexity
1	three	high	no
2	type-graphql	low	yes

3	xmldom	low	no
4	Pug	high	yes

Test 2

		Severity	Types
1	Minimist (karma-mocha)	low	indirect
2	netmask	high	direct
3	angular-expressions	low	direct
4	base64(uid-safe)	high	indirect

Debrief [Ampere]

For this user study, we would like to evaluate if a security vulnerability tool affects understanding and prioritization of vulnerabilities.

- Can you please explain what were the criteria that you used to prioritize the vulnerabilities to fix for each tool?
- In the future, is there any chance that you would use these tools? Will they be used in the same or different scenario?
- Is there other factors that you would update the vulnerability?
- Do you have any feedback or suggestions for Achilles?

[Achilles]

We have another tool to analyze security vulnerabilities and we would like you to try using it.

[Achilles \(achilles-sp.azurewebsites.net\)](http://Achilles.achilles-sp.azurewebsites.net)

- Can you please try it on the Test 1 and Test 2 projects again and let us know how you would prioritize the packages for updates?
- This is the tool that we have developed. Do you have any suggestions for the Achilles tool?

That is the end of this study. We would like to thank you again for your participation in this user study.

<สิ่ง consent form>

<http://bit.ly/achilles-consent>

Hello, my name is <your name>. I'm a researcher in the project "Achilles: Automated tool for detecting and visualizing npm dependency vulnerabilities"

I would like to thank you for joining our user study today. This user study is a part of our senior project which studies security vulnerabilities in npm projects. The project also involves the tools that detect and report security vulnerabilities in such projects.

Now, we would like to ask you to read the participant information form and the consent forms that we have sent to you in the chat. Once you have finished reading and agree to join the study, please let us know by saying "I agree to join the study".

Regarding the study, you will be asked to use a tool for security vulnerability check in npm projects on the projects that we prepared for you. The tool that you'll use today is <npm audit>

Before starting the study, we would like to ask you some questions.

<demographic questions>

1. How long have you been using npm?
2. What do you use npm for?
3. How often do you check security vulnerability in your project? And how?
4. Have you used this tool before? If yes - for how long?
5. Do you know Indirect Dependencies? How much you give importance to them? (สำคัญมาก indirect dependencies)

<ส่ง Document>
เปิดเมลเดรีบยนส่ง verification code

May we ask you to share your screen please?

Next, we'll give you a mock task to try. You will use the two tools (dependabot report and npm audit) and answer the questions.

We have the guide videos on how to use dependabot and npm audit and also the mock repository here.

Video: <https://youtu.be/4ppCoCDtFeo>

[อธิบาย] Dependabot Report จะให้หัวข้อ Direct และ Indirect vulnerabilities ถ้าอยากรายงานก็สามารถดาวน์โหลดที่ package.json file ได้เลย

For now, we have 2 questions, you can let us know when you have finished answering the questions. If you have any inquiries while answering the questions, please feel free to let us know.

Next, we will move to the actual user study.

For this user study, we have 2 test cases for you.
<บอกให้ participants เปิด GitHub overview ดู repo>

There are 2 repositories for the 2 test cases, which are Test 1 and Test 2.

<บอกให้เปิด Document หน้าที่ 5>

The task that you have to do is prioritizing the updates of the vulnerabilities. In each test, you will see the vulnerability report of dependabot in GitHub. You can take notes while seeing the report. After that, you'll use <npm audit> and prioritize the updates of the vulnerabilities again.

Before we proceed, do you allow us to record the video from further analysis?

Test1: [index.js — test1 — code-server \(cdr.co\)](#)

Test2: [yarn.lock — test2 — code-server \(cdr.co\)](#)

Test 1

		Severity	Complexity
1	three	high	no
2	type-graphql	low	yes

3	xmlDom	low	no
4	Pug (template engine)	high	yes

Test 2

		Severity	Types
1	Minimist@1.2.0 (karma-mocha@2.0.1)	low	indirect
2	netmask	High	direct
3	angular-expressions	low	direct
4	base64(uid-safe@2.1.5)	high	indirect

Debrief [Ampere]

For this user study, we would like to evaluate if a security vulnerability tool affects understanding and prioritization of vulnerabilities.

- Can you please explain what were the criteria that you used to prioritize the vulnerabilities to fix for each tool?

[Achilles]

We have another tool to analyze security vulnerabilities and we would like you to try using it.

<https://docs.google.com/document/d/1YN5WTWqWM7DO5zsTnb7EDcotUC2Elb2JPeL4SVdBPIg/edit?usp=sharing>

- Can you please try it on the Test 1 and Test 2 projects again and let us know how you would prioritize the packages for updates?
- This is the tool that we have developed. Do you have any suggestions for the Achilles tool?

That is the end of this study. We would like to thank you again for your participation in this user study.

APPENDIX E
PARTICIPANTS ANSWER SHEET

Participant A1**Date: 09 April 2021**

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	D1 pug >= D2 three > D3 type-graphql >= D4 xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	I checked these severity. I think the most high priority dependency is pug and three because this severity is high
[Option] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: pug Level of severity: high Other note: 9 days ago by GitHub
D2: Dependency 2	Name of vulnerable package: three Level of severity: high Other note: 5 days ago by GitHub
D3: Dependency 3	Name of vulnerable package: type-graphql Level of severity: low Other note: 5 days ago by GitHub
D4: Dependency 4	Name of vulnerable package: xmldom Level of severity: low Other note: 9 days ago by Github
Test 1 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	D1 pug > D2 three > D3 type-graphql > D4 xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?

Participant A1**Date: 09 April 2021**

Answer	I checked these severity and the number of indirect dependencies. I think the highest priority is pug because this severity is high and it has many indirect dependencies. Next one is three because it is high severity. Type-graphql has many dependencies but that severity is low. So I think severity is an important factor than the number of indirect dependencies.
---------------	---

Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use \geq sign
Answer	D1 netmask \geq D2 base64-url > D3 minimist \geq D4 angular-expressions
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	I checked their severity.
[Option] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package: netmask Level of severity: high Other note: 5 days ago
D2: Dependency 2	Name of vulnerable package: base64-url Level of severity: high Other note: 9 days ago
D3: Dependency 3	Name of vulnerable package: minimist Level of severity: low Other note: 4 days ago
D4: Dependency 4	Name of vulnerable package: angular-expressions Level of severity: low Other note: 9 days ago
Test 2 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first.

Participant A1**Date: 09 April 2021**

	For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	D1 netmask > D2 base64-url > D3 angular-expressions > D4 minimalist
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	I checked their severity and the dependency whether direct or not. D1 and D2 are high severity but D1 is direct dependency. I think direct dependency is easier to fix than indirect dependency, then I think it is the highest priority than others.

Participant A2**Date: 09 April 2021**

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	D1>=D4>D2>D3
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Work on high level dependencies first and low afterwards
[Option] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: three Level of severity: high Other note:
D2: Dependency 2	Name of vulnerable package: type-graphql Level of severity: low Other note:
D3: Dependency 3	Name of vulnerable package: xmldom Level of severity: low Other note:
D4: Dependency 4	Name of vulnerable package: pug Level of severity: high Other note:
Test 1 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	D4>D2>D1>D3
Questions	What criteria do you use to prioritize these vulnerability updates?

Participant A2**Date: 09 April 2021**

Answer	Fixing those dependencies with a larger number of interdependencies and higher level of severity first and moving to fewer number of interdependencies
---------------	--

Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	D2>=D4>D1>=D3
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Same as before giving more priority to higher risk vulnerabilities first

[Option] Test 2 Note Section

D1: Dependency 1	Name of vulnerable package: minimst Level of severity: low Other note:
D2: Dependency 2	Name of vulnerable package: netmask Level of severity: high Other note:
D3: Dependency 3	Name of vulnerable package: angular-expressions Level of severity: low Other note:
D4: Dependency 4	Name of vulnerable package: base64-url Level of severity: high Other note:

Test 2 Part 2 - Achilles

Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
-----------------	--

Participant A2**Date: 09 April 2021**

Answer	D2>D3>D1>=D4
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	I'm prioritizing D2 and D3 since they are direct vulnerabilities and from level of severity, for D1 and D4 thanks to Achilles I can see that those libraries are indirect dependencies so I wouldn't be able to actually update those directly and will have to update the direct dependencies instead

Participant A3**Date: 08 April 2021**

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Three >= pug > type-graphql >= xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Vulnerability level
[Optional] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: three Level of severity: high Other note: Upgrade three to version 0.125.0 or later
D2: Dependency 2	Name of vulnerable package: pug Level of severity: high Other note: Upgrade pug to version 3.0.1 or later
D3: Dependency 3	Name of vulnerable package: type-graphql Level of severity: low Other note: Upgrade type-graphql to version 0.17.6 or later
D4: Dependency 4	Name of vulnerable package: xmldom Level of severity: low Other note: Upgrade xmldom to version 0.5.0 or later
Test 1 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Three >= pug > xmldom >= type-graphql
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Vulnerability level

Participant A3**Date: 08 April 2021**

Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	D1 [Netmask] > angular-expressions > base-64url >= minimalist
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Type of dependency, vulnerability level
[Optional] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package: netmask Level of severity: high Other note: Upgrade netmask to version 2.0.1 or later
D2: Dependency 2	Name of vulnerable package: angular-expressions Level of severity: low Other note: Upgrade angular-expressions to version 1.1.2 or later
D3: Dependency 3	Name of vulnerable package: base-64url Level of severity: high Other note: Upgrade base64-url to version 2.0.0 or later
D4: Dependency 4	Name of vulnerable package: minimalist Level of severity: low Other note: Upgrade minimalist to version 1.2.3 or later
Test 2 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Netmask > angular-expressions > base-64url >= minimalist
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Type of dependency, vulnerability level

Participant A4**Date: 08 April 2021**

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Three >= pug > type-graphql >= xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity
[Optional] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: three Level of severity: high Other note:
D2: Dependency 2	Name of vulnerable package: pug Level of severity: high Other note:
D3: Dependency 3	Name of vulnerable package: type-graphql Level of severity: low Other note:
D4: Dependency 4	Name of vulnerable package: xmldom Level of severity: low Other note:
Test 1 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Pug > three > type-graphql > xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity and level of indirect dependency complexity?

Participant A4**Date: 08 April 2021**

Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Netmask >= base64-url > minimalist >= angular-expressions
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity
[Optional] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package: netmask Level of severity: high Other note:
D2: Dependency 2	Name of vulnerable package: base64-url Level of severity: high Other note:
D3: Dependency 3	Name of vulnerable package: minimalist Level of severity: low Other note:
D4: Dependency 4	Name of vulnerable package: angular-expressions Level of severity: low Other note:
Test 2 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Netmask > angular-expressions > base64-url > minimalist
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Direct first > severity

Participant A5**Date: 08 April 2021**

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Pug > Three.js > xmldom > typegraphql
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity and complexity of package & direct and indirect dependency
[Option] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: pug Level of severity: hight Other note:
D2: Dependency 2	Name of vulnerable package: three.js Level of severity:high Other note:
D3: Dependency 3	Name of vulnerable package: xmldom Level of severity:low Other note:
D4: Dependency 4	Name of vulnerable package:typegraphql Level of severity:low Other note:
Test 1 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	three.js > pug > xmldom > typegraphql
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity and complexity of package & direct and indirect dependency

Participant A5**Date: 08 April 2021**

Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Netmask > base64-url > angular-expressions > minimalist
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity and complexity of package & direct and indirect dependency
[Option] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package: Netmask Level of severity: high Other note:
D2: Dependency 2	Name of vulnerable package: base64-url Level of severity: high Other note:
D3: Dependency 3	Name of vulnerable package: angular-expressions Level of severity: low Other note:
D4: Dependency 4	Name of vulnerable package: minimalist Level of severity: low Other note:
Test 2 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Netmask > base64-url > angular-expressions > minimalist
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity and complexity of package & direct and indirect dependency

Participant A5**Date: 08 April 2021**

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	pug >= three > type-graphql >= xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	severity
[Option] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: pug Level of severity: high severity Other note: -
D2: Dependency 2	Name of vulnerable package: three Level of severity: high severity Other note: -
D3: Dependency 3	Name of vulnerable package: type-graphql Level of severity: low severity Other note: -
D4: Dependency 4	Name of vulnerable package: xmldom Level of severity: low severity Other note: -
Test 1 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	pug >= three > type-graphql >= xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	severity

Participant A5**Date: 08 April 2021**

Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	netmask >= base64-url > angular-expressions > minimalist
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity and issue type
[Option] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package: netmask Level of severity: high severity Other note: -
D2: Dependency 2	Name of vulnerable package: base64-url Level of severity: high severity Other note: -
D3: Dependency 3	Name of vulnerable package: angular-expressions Level of severity: low severity Other note: -
D4: Dependency 4	Name of vulnerable package: minimalist Level of severity: low severity Other note: -
Test 2 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	netmask > base64-url > angular-expressions > minimalist
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity and direct/indirect graph

Participant A7**Date: 08 April 2021**

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	D4 [xmldom] > D3 [type-graphql] > D2 [three.js] > D1 [pug]
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Number of indirect dependencies
[Option] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: pug Level of severity: high Other note:
D2: Dependency 2	Name of vulnerable package: three.js Level of severity: high Other note:
D3: Dependency 3	Name of vulnerable package: type-graphql Level of severity: low Other note:
D4: Dependency 4	Name of vulnerable package: xmldom Level of severity: low Other note:
Test 1 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	D1 [three.js] >= D2 [xmldom] > D3 [pug] > D4 [type-graphql]
Questions	What criteria do you use to prioritize these vulnerability updates?

Participant A7**Date: 08 April 2021**

Answer	Number of indirect dependencies in each library. If the number is high, it may interrupt other libraries once updated.
---------------	---

Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	D1 [node-netmask] > D2 [uid-safe] > D3 [angular-expressions] > D4 [karma-mocha]
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	The number of indirect dependencies

[Option] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package: Level of severity: Other note:
D2: Dependency 2	Name of vulnerable package: Level of severity: Other note:
D3: Dependency 3	Name of vulnerable package: Level of severity: Other note:
D4: Dependency 4	Name of vulnerable package: Level of severity: Other note:

Test 2 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign

Participant A7**Date: 08 April 2021**

Answer	D1 [netmask] > D2 [angular-expressions] > D3 [minimist] > D4 [base64-url]
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Number of indirects dependencies and severity

Participant A8**Date: 09 April 2021**

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	three>pug>type-graphql>xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	severity (high first) then time (recent use first)
[Option] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: three Level of severity: direct Other note:
D2: Dependency 2	Name of vulnerable package: pug Level of severity: direct Other note:
D3: Dependency 3	Name of vulnerable package: type-graphql Level of severity: direct Other note:
D4: Dependency 4	Name of vulnerable package: xmldom Level of severity: direct Other note:
Test 1 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	three>xmldom>type-graphql>pug
Questions	What criteria do you use to prioritize these vulnerability updates?

Participant A8**Date: 09 April 2021**

Answer	How big of the dependency graph (small first because it might take shorter time for fixing)
---------------	--

Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	netmask>base64-url>minimist>angular-expressions
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	High severity first then more recent time

[Option] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package: Level of severity: Other note:
D2: Dependency 2	Name of vulnerable package: Level of severity: Other note:
D3: Dependency 3	Name of vulnerable package: Level of severity: Other note:
D4: Dependency 4	Name of vulnerable package: Level of severity: Other note:

Test 2 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	netmask>=base64-url>minimist>angular-expressions

Participant A8**Date: 09 April 2021**

Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	ค่อยดูจาก severity แล้วดูว่าแต่ละตัวมีตัวต่อ曳อะแครไหน เอาตัวที่มีตัวต่อ น้อยก่อน

Participant A9

Date: 09 April 2021

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	mrdoob / three.js >= MichalLytek / type-graphql >= xmldom / xmldom >= pugjs / pug
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	The number of versions. If the package has many, it should be less vulnerable. So, it should be the last priority.
[Option] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: Level of severity: Other note:
D2: Dependency 2	Name of vulnerable package: Level of severity: Other note:
D3: Dependency 3	Name of vulnerable package: Level of severity: Other note:
D4: Dependency 4	Name of vulnerable package: Level of severity: Other note:
Test 1 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and

Participant A9**Date: 09 April 2021**

	dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Three >= Pug >= Michał Łytek / type-graphql >= xmldom / xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Use the severity and version. High severity and Less version

Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	karma-runner / karma-mocha >= peerigon / angular-expressions >= crypto-utils / uid-safe >= rs / node-netmask
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	The number of indirect dependencies. High > Low
[Option] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package: Level of severity: Other note:
D2: Dependency 2	Name of vulnerable package: Level of severity: Other note:
D3: Dependency 3	Name of vulnerable package: Level of severity: Other note:
D4: Dependency 4	Name of vulnerable package: Level of severity: Other note:
Test 2 Part 2 - Achilles	

Participant A9**Date: 09 April 2021**

Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	<u>crypto-utils / uid-safe >= rs / node-netmask >= karma-runner / karma-mocha >= peerigon / angular-expressions</u>
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Use the severity and version. High severity and Less version

Participant A10**Date: 09 April 2021**

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	pug>= three > type-graphql >= xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Level of severity
[Option] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: pug Level of severity: high Other note: Vulnerable versions: < 3.0.1
D2: Dependency 2	Name of vulnerable package: three Level of severity: high Other note: Vulnerable versions: < 0.125.0
D3: Dependency 3	Name of vulnerable package: type-graphql Level of severity: low Other note: Vulnerable versions: < 0.17.6
D4: Dependency 4	Name of vulnerable package: xmldom Level of severity: low Other note: Vulnerable versions: < 0.5.0

Participant A10

Date: 09 April 2021

Test 1 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	three>=pug>xmldom>=type-graphq
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Level of Severity

Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	netmask>=base64-url> minimist>=angular-exp ressions
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Level of Severity
[Option] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package: Level of severity: Other note:
D2: Dependency 2	Name of vulnerable package: Level of severity: Other note:

Participant A10**Date: 09 April 2021**

D3: Dependency 3	Name of vulnerable package: Level of severity: Other note:
D4: Dependency 4	Name of vulnerable package: Level of severity: Other note:
Test 2 Part 2 - Achilles	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	netmask>angular-expressions>base64-url>minimist
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Direct before indirect and Level of Severity

Participant N1**Date: 09 April 2021**

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Three > pug > xmldom > typegraphql
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity and alerted time
[Option] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: three Level of severity:high Other note:
D2: Dependency 2	Name of vulnerable package: typegraphql Level of severity:low Other note:
D3: Dependency 3	Name of vulnerable package: xmldom Level of severity:low Other note:
D4: Dependency 4	Name of vulnerable package: pug Level of severity:high Other note:
Test 1 Part 2 - npm audit (https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test1)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Three > pug > xmldom > typegraphql
Questions	What criteria do you use to prioritize these vulnerability updates?

Participant N1

Date: 09 April 2021

Answer	Severity
Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	netmask > angular-expressions > Base64-url > minimalist
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Existing solving pull request and severity
[Option] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package: minimalist Level of severity: low Other note:
D2: Dependency 2	Name of vulnerable package: netmask Level of severity: high Other note:
D3: Dependency 3	Name of vulnerable package: angular-expressions Level of severity: low Other note:
D4: Dependency 4	Name of vulnerable package: base64-url Level of severity: high Other note:
Test 2 Part 2 - npm audit (https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test2)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	netmask > angular-expressions > base64-url > minimalist

Participant N1**Date: 09 April 2021**

Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Direct dependency and severity

Participant N2

Date: 09 April 2021

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Pug > three > type-graphQI > xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	I see the impact on the server first and then use level of severity as criteria
[Option] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: Level of severity: Other note:
D2: Dependency 2	Name of vulnerable package: Level of severity: Other note:
D3: Dependency 3	Name of vulnerable package: Level of severity: Other note:
D4: Dependency 4	Name of vulnerable package: Level of severity: Other note:
Test 1 Part 2 - npm audit (https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test1)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Pug > three > graphql > xml

Participant N2**Date: 09 April 2021**

Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Level of severity

Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	netmask> base64-url > minimist > = angular-expression
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Level of severity and how it can impact the project

[Option] Test 2 Note Section

D1: Dependency 1	Name of vulnerable package: Level of severity: Other note:
D2: Dependency 2	Name of vulnerable package: Level of severity: Other note:
D3: Dependency 3	Name of vulnerable package: Level of severity: Other note:
D4: Dependency 4	Name of vulnerable package: Level of severity: Other note:

Test 2 Part 2 - npm audit**(https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test2)**

Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
-----------------	--

Participant N2**Date: 09 April 2021**

Answer	netmask> base64-url > minimalist > = angular-expression
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Level of severity

Participant N3

Date: 08 April 2021

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	pug > three.js > type-graphql >= xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Level of severity, impact with the project, ease of modification
[Option] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: pug Level of severity: High Other note: Main renderer library
D2: Dependency 2	Name of vulnerable package: three.js Level of severity: High Other note: Consist with rendering part
D3: Dependency 3	Name of vulnerable package: type-graphql Level of severity: Low Other note:
D4: Dependency 4	Name of vulnerable package: xmldom Level of severity: Low Other note:
Test 1 Part 2 - npm audit (https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test1)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	pug > three.js > type-graphql >= xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?

Participant N3

Date: 08 April 2021

Answer	Level of severity, impact with the project
Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	base64-url >= node-netmark > angular-expressions >= minimist
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Level of severity
[Option] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package: base64-url Level of severity: High Other note:
D2: Dependency 2	Name of vulnerable package: node-netmark Level of severity: High Other note:
D3: Dependency 3	Name of vulnerable package: angular-expressions Level of severity: Low Other note:
D4: Dependency 4	Name of vulnerable package: minimist Level of severity: Low Other note:
Test 2 Part 2 - npm audit	
https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test2	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	node-netmark > angular-expressions > base64-url > minimist

Participant N3**Date: 08 April 2021**

Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Type of dependency, Level of severity, Vulnerability effect (type of vulnerability)

Participant N4**Date: 08 April 2021**

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	three > pug > typegraphql >= xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity and CVE
[Optional] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: three Level of severity: high Other note: CVE-2020-28496
D2: Dependency 2	Name of vulnerable package: typegraphql Level of severity: low Other note:
D3: Dependency 3	Name of vulnerable package: xmldom Level of severity: low Other note:
D4: Dependency 4	Name of vulnerable package: pug Level of severity: high Other note:
Test 1 Part 2 - npm audit (https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test1)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Pug > three > xmldom > typegraphql
Questions	What criteria do you use to prioritize these vulnerability updates?

Participant N4

Date: 08 April 2021

Answer	Vulnerability Types
Test 1 Part 3 - achilles Achilles (achilles-sp.azurewebsites.net)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	

Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Netmask > base64-url > minimalist > angular-expressions
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Impact, Severity, and CVE (newest)
[Optional] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package: minimalist Level of severity: low Other note: CVE-2020-7598
D2: Dependency 2	Name of vulnerable package: netmask Level of severity: high Other note:
D3: Dependency 3	Name of vulnerable package: angular-expressions Level of severity: low Other note:

Participant N4**Date: 08 April 2021**

D4: Dependency 4	Name of vulnerable package: base64-url Level of severity: high Other note: CVE-2021-29418
Test 2 Part 2 - npm audit (https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test2)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Netmask > angular-expressions > base64-url > minimalist
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Vulnerability Type, Severity, and Impact
Test 2 Part 3 - achilles Achilles (achilles-sp.azurewebsites.net)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	

Participant N5**Date: 08 April 2021**

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Three >= Pug > type-graphql > xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity
[Option] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: three Level of severity: High Other note:
D2: Dependency 2	Name of vulnerable package: Pug Level of severity: High Other note:
D3: Dependency 3	Name of vulnerable package: type-graphql Level of severity: Low Other note:
D4: Dependency 4	Name of vulnerable package: xmldom Level of severity: low Other note:
Test 1 Part 2 - npm audit (https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test1)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Three >= pug > type-graphql > xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?

Participant N5

Date: 08 April 2021

Answer	Severity
Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Netmask >= base64-url > angular-expressions >= minimalist
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity
[Option] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package: netmask Level of severity: High Other note:
D2: Dependency 2	Name of vulnerable package: base64-url Level of severity: High Other note:
D3: Dependency 3	Name of vulnerable package: angular-expressions Level of severity: Low Other note:
D4: Dependency 4	Name of vulnerable package: minimalist Level of severity: Low Other note:
Test 2 Part 2 - npm audit	
(https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test2)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Netmask >= base64-url > angular-expressions >= minimalist

Participant N5**Date: 08 April 2021**

Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity

Participant N6**Date: 08 April 2021**

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Three >= pug > xml-dom >= type-graphql
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity
[Option] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: three Level of severity: High Other note:
D2: Dependency 2	Name of vulnerable package: pug Level of severity: High Other note:
D3: Dependency 3	Name of vulnerable package: xml-dom Level of severity: Low Other note:
D4: Dependency 4	Name of vulnerable package: type-graphql Level of severity: Low Other note:
Test 1 Part 2 - npm audit (https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test1)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Three >= pug > xml-dom >= type-graphql
Questions	What criteria do you use to prioritize these vulnerability updates?

Participant N6

Date: 08 April 2021

Answer	Severity
--------	----------

Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Netmask >= base64-url > minimalist >= angular-expressions
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity
[Option] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package: netmask Level of severity: High Other note:
D2: Dependency 2	Name of vulnerable package: base64-url Level of severity: High Other note:
D3: Dependency 3	Name of vulnerable package: minimalist Level of severity: Low Other note:
D4: Dependency 4	Name of vulnerable package: angular-expression Level of severity: Low Other note:
Test 2 Part 2 - npm audit (https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test2)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Netmask > angular-expressions > base64-url > minimalist

Participant N6

Date: 08 April 2021

Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity, Dependency

Participant N7

Date: 08 April 2021

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Type-graphql > three.js > xmldom > pug
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	The ease of patching the packages.
[Option] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: Level of severity: Other note:
D2: Dependency 2	Name of vulnerable package: Level of severity: Other note:
D3: Dependency 3	Name of vulnerable package: Level of severity: Other note:
D4: Dependency 4	Name of vulnerable package: Level of severity: Other note:
Test 1 Part 2 - npm audit (https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test1)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Type-graphql > three.js > xmldom > pug
Questions	What criteria do you use to prioritize these vulnerability updates?

Participant N7

Date: 08 April 2021

Answer	Ease of patching. Update packages with no breaking changes first.
Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Netmask > angular-expression >= minimalist > base64-url
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Ease of fixing and level of vulnerability (Order by ease of fixing, then the level of vulnerability)
[Option] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package: Level of severity: Other note:
D2: Dependency 2	Name of vulnerable package: Level of severity: Other note:
D3: Dependency 3	Name of vulnerable package: Level of severity: Other note:
D4: Dependency 4	Name of vulnerable package: Level of severity: Other note:
Test 2 Part 2 - npm audit (https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test2)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign

Participant N7**Date: 08 April 2021**

Answer	Netmask > angular-expressions > karma-mocha > uid-safe
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Patch the packages with no breaking changes first (order by level of severity).

Participant N8**Date: 09 April 2021**

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Pug > three > xmldom > typegraphql
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Severity level + details of vulnerability risk
[Option] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: Pug Level of severity: high Other note:
D2: Dependency 2	Name of vulnerable package: three Level of severity: high Other note:
D3: Dependency 3	Name of vulnerable package: xmldom Level of severity: low Other note:
D4: Dependency 4	Name of vulnerable package: typegraphql Level of severity: low Other note:
Test 1 Part 2 - npm audit (https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test1)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Three > pug > typegraphql >= xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?

Participant N8

Date: 09 April 2021

Answer	Level of vulnerability + expected error
Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	D1 > D4 >= D3 > D2
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Details of vulnerability
[Option] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package: netmask Level of severity: high severity Other note: bypass access ctrl
D2: Dependency 2	Name of vulnerable package: base64-url Level of severity: high Other note: allocate uninit buffer
D3: Dependency 3	Name of vulnerable package: angular - expression Level of severity: low Other note: bypass but using complex payload
D4: Dependency 4	Name of vulnerable package: minimist Level of severity: low Other note: atker modify prototype
Test 2 Part 2 - npm audit (https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test2)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Netmask > Base64-url > minimalist >= angular expression

Participant N8

Date: 09 April 2021

Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	Level and vulnerability that can cause.

Participant N9**Date: 09 April 2021**

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	D1 three >= D4 pug > D2 type_graphql >= D3 xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	By the level of risk of severity
[Option] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: three Level of severity: High Other note:
D2: Dependency 2	Name of vulnerable package: type_graphql Level of severity: Low Other note:
D3: Dependency 3	Name of vulnerable package: xmldom Level of severity: Low Other note:
D4: Dependency 4	Name of vulnerable package: pug Level of severity: High Other note:
Test 1 Part 2 - npm audit (https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test1)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	D2 three >= D3 pug > D1 type-graphql >= D4 xmldom
Questions	What criteria do you use to prioritize these vulnerability updates?

Participant N9

Date: 09 April 2021

Answer	By the level of risk of severity
Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	D2 netmask >= D4 base64-url > D1 minimalist >= D3 angular-expressions
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	By the risk level of severity
[Option] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package: minimalist Level of severity: Low Other note:
D2: Dependency 2	Name of vulnerable package: netmask Level of severity: High Other note:
D3: Dependency 3	Name of vulnerable package: angular-expressions Level of severity: High Other note:
D4: Dependency 4	Name of vulnerable package: base64-url Level of severity: Low Other note:
Test 2 Part 2 - npm audit (https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test2)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	D1 base64-url >= D4 netmask > D2 minimalist >= D3 angular-expressions

Participant N9**Date: 09 April 2021**

Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	By the risk level of severity

Participant N10

Date: 19 April 2021

Test 1 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Pug >= three > type-graphql => xlmdom
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	From severity level
[Option] Test 1 Note Section	
D1: Dependency 1	Name of vulnerable package: Level of severity: Other note:
D2: Dependency 2	Name of vulnerable package: Level of severity: Other note:
D3: Dependency 3	Name of vulnerable package: Level of severity: Other note:
D4: Dependency 4	Name of vulnerable package: Level of severity: Other note:
Test 1 Part 2 - npm audit	
(https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test1)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Pug >= three > type-graphql >= xlmdom
Questions	What criteria do you use to prioritize these vulnerability updates?

Participant N10

Date: 19 April 2021

Answer	From level of severity
Test 2 Part 1 - Dependabot Report	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Netmask > angular-expressions > minimalist > base64-url
Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	From chance of attacker
[Option] Test 2 Note Section	
D1: Dependency 1	Name of vulnerable package:minimalist Level of severity: low Other note:
D2: Dependency 2	Name of vulnerable package:netmask Level of severity: high Other note:
D3: Dependency 3	Name of vulnerable package: angular-expressions Level of severity: low Other note:
D4: Dependency 4	Name of vulnerable package: base64-url Level of severity: high Other note:
Test 2 Part 2 - npm audit	
(https://npm_audit_achilles-achilles-baak.cdr.co/?folder=/home/baak-npm-audit/test2)	
Question	If you need to update the vulnerable dependencies to fix the security issues, please order which vulnerabilities you will update first. For example, D1 [Name] > D2 [Name] > D3 [Name] > D4 [Name] would mean you will fix the dependency 1 first, then fix dependency 2, and dependency 3, and dependency 4 respectively. In case the order is interchangeable, please use >= sign
Answer	Netmask>= base64-url > angular-expressions >= minimalist

Participant N10

Date: 19 April 2021

Questions	What criteria do you use to prioritize these vulnerability updates?
Answer	From severity

APPENDIX F
ACHILLES SECURITY VULNERABILITY REPORT FROM
GITHUB PROJECTS



Achilles: Vulnerability Report

April 18th 2021, 10:13 pm

KlintonICT
baak-packagejson-test
From:cpnmjs/package.json



Summary

Total vulnerabilities: 11

Dependency	Type	Updating	Severity
sequelize	Direct	< 4.44.4 → 4.44.4	moderate
sequelize	Direct	< 4.44.3 → 4.44.3	high
sequelize	Direct	< 4.12.0 → 4.12.0	high
treetkill	Direct	≥ 0.0.0 → none	high
debug	Indirect	< 2.6.9 → 2.6.9	low
debug	Indirect	< 2.6.9 → 2.6.9	low
debug	Indirect	< 2.6.9 → 2.6.9	low
debug	Indirect	< 2.6.9 → 2.6.9	low
ejs	Indirect	< 2.5.3 → 2.5.5	high
ejs	Indirect	< 2.5.5 → 2.5.5	high
ejs	Indirect	< 2.5.5 → 2.5.5	moderate

Total of vulnerable direct dependency: 4

Total of vulnerable indirect dependency: 7

Vulnerabilities

Potentially Vulnerable:	sequelize
Severity:	moderate
Current Usage Version:	^3.23.4
Vulnerable Version:	< 4.44.4
Patch Version:	4.44.4
Vulnerability Chaining:	

 Vulnerabilities and Advisory link: GHSA-fw4p-36j9-rrj3	
<p>Dependency to be updated: sequelize Update sequelize to latest version: ^3.23.4 → 6.6.2</p> <hr/> <p>Potentially Vulnerable: sequelize Severity: high Current Usage Version: ^3.23.4 Vulnerable Version: < 4.44.3 Patch Version: 4.44.3</p>	<p>Vulnerability Chaining:</p> 
<p>Vulnerabilities and Advisory link: GHSA-m9lw-237r-gvfv CVE-2019-10752 </p> <hr/> <p>Dependency to be updated: sequelize Update sequelize to latest version: ^3.23.4 → 6.6.2</p>	
<p>Potentially Vulnerable: sequelize Severity: high Current Usage Version: ^3.23.4 Vulnerable Version: < 4.12.0 Patch Version: 4.12.0</p>	<p>Vulnerability Chaining:</p> 
<p>Vulnerabilities and Advisory link: GHSA-wfp9-vr4j-f49j </p> <hr/> <p>Dependency to be updated: sequelize Update sequelize to latest version: ^3.23.4 → 6.6.2</p> <hr/> <p>Potentially Vulnerable: treekill Severity: high Current Usage Version: ^1.0.0 Vulnerable Version: >= 0.0.0 Patch Version: Currently, no patch version</p>	

Vulnerability Chaining:



Vulnerabilities and Advisory link:

[GHSA-533p-g2hq-qr26](#)

Dependency to be updated:

treekill

Update **treekill** to latest version:

$\wedge 1.0.0 \rightarrow 1.0.0$

Potentially Vulnerable:

debug

Severity:

low

Current Usage Version:

$\sim 2.2.0$

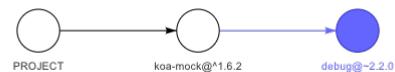
Vulnerable Version:

$< 2.6.9$

Patch Version:

2.6.9

Vulnerability Chaining:



Vulnerabilities and Advisory link:

[GHSA-gxpi-cx7g-858c](#)

[CVE-2017-16137](#)

CWEs:

[CWE-400 : Uncontrolled Resource Consumption](#)

Dependency to be updated:

koa-mock

Update **koa-mock** to latest version:

$\wedge 1.6.2 \rightarrow 2.0.0$

Potentially Vulnerable:

debug

Severity:

low

Current Usage Version:

$\sim 0.8.0$

Vulnerable Version:

$< 2.6.9$

Patch Version:

2.6.9

Vulnerability Chaining:



Vulnerabilities and Advisory link:

[GHSA-gxpi-cx7g-858c](#)

[CVE-2017-16137](#)

CWEs:

[CWE-400 : Uncontrolled Resource Consumption](#)

Dependency to be updated:

changes-stream

Update **changes-stream** to latest version:

$\wedge 1.1.0 \rightarrow 2.2.0$

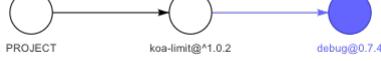
Potentially Vulnerable:

debug

Severity:

low

Current Usage Version:

0.7.4	Vulnerable Version:
< 2.6.9	Patch Version:
2.6.9	Vulnerability Chaining:
	Vulnerabilities and Advisory link:
	
GHSA-gxpi-cx7g-858c	CWEs:
CVE-2017-16137	
CWE-400 : Uncontrolled Resource Consumption	Dependency to be updated:
koa-limit	Update koa-limit to latest version:
^1.0.2 → 1.0.2	
<hr/>	
Potentially Vulnerable:	debug
Severity:	low
Current Usage Version:	^0.7.4
Vulnerable Version:	< 2.6.9
Patch Version:	2.6.9
	
Vulnerabilities and Advisory link:	GHSA-gxpi-cx7g-858c
	CVE-2017-16137
CWEs:	CWE-400 : Uncontrolled Resource Consumption
Dependency to be updated:	koa-middlewares
Update koa-middlewares to latest version:	^2.1.0 → 6.0.0
<hr/>	
Potentially Vulnerable:	ejs
Severity:	high
Current Usage Version:	^1.0.0
Vulnerable Version:	< 2.5.3
Patch Version:	2.5.5
	
Vulnerabilities and Advisory link:	GHSA-3w5v-p54c-f74x
	CVE-2017-1000228
Dependency to be updated:	koa-middlewares

Update koa-middlewares to latest version:	$\wedge 2.1.0 \rightarrow 6.0.0$
Potentially Vulnerable:	ejs
Severity:	high
Current Usage Version:	$\wedge 1.0.0$
Vulnerable Version:	< 2.5.5
Patch Version:	2.5.5
Vulnerability Chaining:	<pre> graph LR koaMiddlewares((koajs@^2.1.0)) --> koaejs((koajs-ejs@~1.1.3)) koaejs --> ejss((ejss@^1.0.0)) PROJECT --> koaMiddlewares </pre>
Vulnerabilities and Advisory link:	GHSA-6x77-ppqf-j6mw CVE-2017-1000189
Dependency to be updated:	koajs
Update koajs to latest version:	$\wedge 2.1.0 \rightarrow 6.0.0$
Potentially Vulnerable:	ejs
Severity:	moderate
Current Usage Version:	$\wedge 1.0.0$
Vulnerable Version:	< 2.5.5
Patch Version:	2.5.5
Vulnerability Chaining:	<pre> graph LR koaMiddlewares((koajs@^2.1.0)) --> koaejs((koajs-ejs@~1.1.3)) koaejs --> ejss((ejss@^1.0.0)) PROJECT --> koaMiddlewares </pre>
Vulnerabilities and Advisory link:	GHSA-hwcf-pp87-7x6p CVE-2017-1000188
Dependency to be updated:	koajs
Update koajs to latest version:	$\wedge 2.1.0 \rightarrow 6.0.0$



Achilles: Vulnerability Report

April 18th 2021, 10:19 pm

KlintonICT
baak-packagejson-test
From:npx/package.json



Summary

Total vulnerabilities: 1

Dependency	Type	Updating	Severity
yargs-parser	Indirect	<code>>= 6.0.0, < 13.1.2</code> → <code>13.1.2</code>	<code>low</code>

Total of vulnerable direct dependency: 0

Total of vulnerable indirect dependency: 1

Vulnerability

Potentially Vulnerable: yargs-parser

Severity: `low`

Current Usage Version: `^9.0.2`

Vulnerable Version: `>= 6.0.0, < 13.1.2`

Patch Version: `13.1.2`

Vulnerability Chaining:



Vulnerabilities and Advisory link: [GHSA-p9pc-299p-vxgp](#)

CWEs: [CWE-471: Modification of Assumed-Immutable Data \(MAID\)](#)

Dependency to be updated: yargs

Update **yargs** to latest version: `^11.1.0 → 16.2.0`



Achilles: Vulnerability Report

April 18th 2021, 10:08 pm

KlintonICT
baak-packagejson-test
From:sinopia/package.json



Summary

Total vulnerabilities: 12

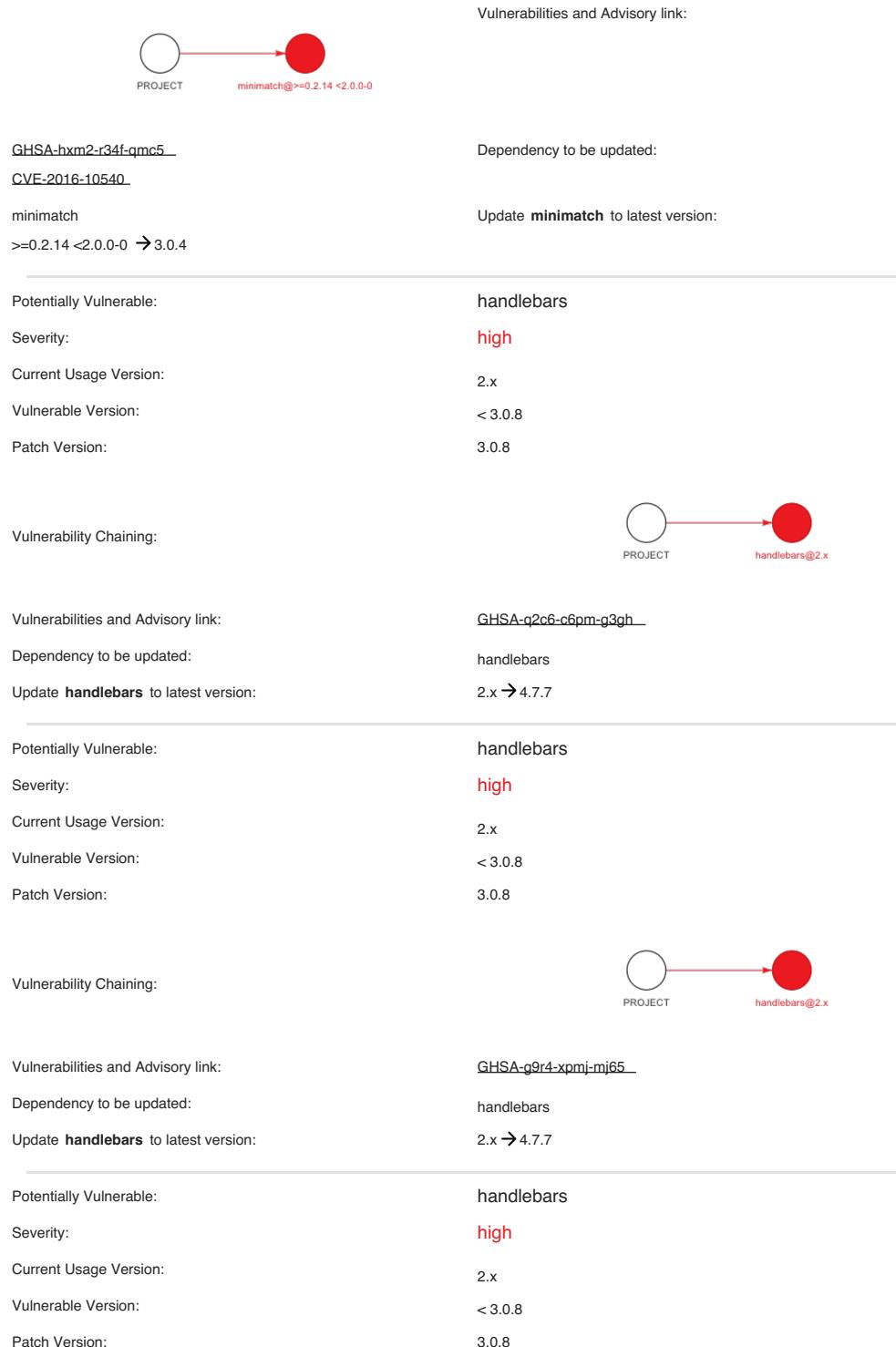
Dependency	Type	Updating	Severity
minimatch	Direct	<3.0.2 → 3.0.2	high
handlebars	Direct	<3.0.8 → 3.0.8	high
handlebars	Direct	<3.0.8 → 3.0.8	high
handlebars	Direct	<3.0.8 → 3.0.8	high
handlebars	Direct	<4.3.0 → 4.3.0	high
handlebars	Direct	<3.0.7 → 3.0.7	critical
handlebars	Direct	<4.0.0 → 4.0.0	high
handlebars	Direct	<4.0.0 → 4.0.0	moderate
highlight.js	Direct	<9.18.2 → 9.18.2	low
uglify-js	Indirect	<2.4.24 → 2.4.24	high
uglify-js	Indirect	<2.4.24 → 2.4.24	low
uglify-js	Indirect	<2.6.0 → 2.6.0	low

Total of vulnerable direct dependency: 9

Total of vulnerable indirect dependency: 3

Vulnerabilities

Potentially Vulnerable:	minimatch
Severity:	high
Current Usage Version:	>=0.2.14 <2.0.0-0
Vulnerable Version:	< 3.0.2
Patch Version:	3.0.2
Vulnerability Chaining:	



Vulnerability Chaining:



Vulnerabilities and Advisory link:

[GHSA-2cf5-4w76-r9qv](#)

Dependency to be updated:

handlebars

Update **handlebars** to latest version:

2.x → 4.7.7

Potentially Vulnerable:

handlebars

Severity:

high

Current Usage Version:

2.x

Vulnerable Version:

< 4.3.0

Patch Version:

4.3.0

Vulnerability Chaining:



Vulnerabilities and Advisory link:

[GHSA-w457-6q6x-cgp9](#)[CVE-2019-19919](#)

Dependency to be updated:

handlebars

Update **handlebars** to latest version:

2.x → 4.7.7

Potentially Vulnerable:

handlebars

Severity:

critical

Current Usage Version:

2.x

Vulnerable Version:

< 3.0.7

Patch Version:

3.0.7

Vulnerability Chaining:



Vulnerabilities and Advisory link:

[GHSA-q42p-pg8m-cqh6](#)

Dependency to be updated:

handlebars

Update **handlebars** to latest version:

2.x → 4.7.7

Potentially Vulnerable:

handlebars

Severity:

high

Current Usage Version:

2.x

Vulnerable Version:

< 4.0.0

Patch Version:

4.0.0

Vulnerability Chaining:



Vulnerabilities and Advisory link:

[GHSA-9prh-257w-9277](#)

[CVE-2015-8861](#)

Dependency to be updated:

handlebars

Update **handlebars** to latest version:

2.x → 4.7.7

Potentially Vulnerable:

handlebars

Severity:

moderate

Current Usage Version:

2.x

Vulnerable Version:

< 4.0.0

Patch Version:

4.0.0

Vulnerability Chaining:



Vulnerabilities and Advisory link:

[GHSA-fmr4-7g9q-7hc7](#)

[CVE-2015-8861](#)

Dependency to be updated:

handlebars

Update **handlebars** to latest version:

2.x → 4.7.7

Potentially Vulnerable:

highlight.js

Severity:

low

Current Usage Version:

8.x

Vulnerable Version:

< 9.18.2

Patch Version:

9.18.2

Vulnerability Chaining:



Vulnerabilities and Advisory link:

[GHSA-vfrc-777c-w9mx](#)

[CVE-2020-26237](#)

CWEs:

[CWE-471: Modification of Assumed-Immutable Data \(MAID\)](#)

Dependency to be updated:

highlight.js

Update **highlight.js** to latest version:

8.x → 10.7.2

Potentially Vulnerable:

uglify-js

Severity:

high

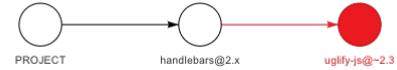
Current Usage Version:

~2.3

Vulnerable Version: < 2.4.24

Patch Version: 2.4.24

Vulnerability Chaining:



Vulnerabilities and Advisory link:

[GHSA-g6f4-j6c2-w3p3](#)

[CVE-2015-8857](#)

Dependency to be updated:

handlebars

Update **handlebars** to latest version:

2.x → 4.7.7

Potentially Vulnerable:

uglify-js

Severity:

low

Current Usage Version:

~2.3

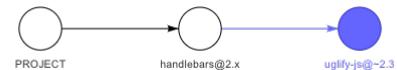
Vulnerable Version:

< 2.4.24

Patch Version:

2.4.24

Vulnerability Chaining:



Vulnerabilities and Advisory link:

[GHSA-34r7-q49f-h37c](#)

[CVE-2015-8857](#)

Dependency to be updated:

handlebars

Update **handlebars** to latest version:

2.x → 4.7.7

Potentially Vulnerable:

uglify-js

Severity:

low

Current Usage Version:

~2.3

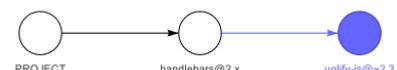
Vulnerable Version:

< 2.6.0

Patch Version:

2.6.0

Vulnerability Chaining:



Vulnerabilities and Advisory link:

[GHSA-c9f4-xj24-8jqx](#)

[CVE-2015-8858](#)

Dependency to be updated:

handlebars

Update **handlebars** to latest version:

2.x → 4.7.7



Achilles: Vulnerability Report

April 18th 2021, 10:18 pm

KlintonICT
baak-packagejson-test
From:window-build-tools/package.json



Summary

Total vulnerabilities: 1

Dependency	Type	Updating	Severity
mem	Indirect	<4.0.0 → 4.0.0	low

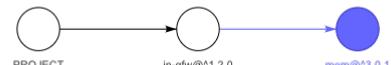
Total of vulnerable direct dependency: 0

Total of vulnerable indirect dependency: 1

Vulnerability

Potentially Vulnerable:	mem
Severity:	low
Current Usage Version:	^3.0.1
Vulnerable Version:	< 4.0.0
Patch Version:	4.0.0

Vulnerability Chaining:



Vulnerabilities and Advisory link:

[GHSA-4xcv-9jx-qfj3](#)

Dependency to be updated:

in-gfw

Update **in-gfw** to latest version:

^1.2.0 → 1.2.0

REFERENCES

- [1] Alyssa Miller SZ., “The State of Open Source Security 2020k”; June 2020 [cited 9 November 2020], [Online]. Available: <https://snyk.io/open-source-security/>.
- [2] Todorov B., Kula R., Ishio T., Inoue K., “SoL Mantra: Visualizing Update Opportunities Based on Library Coexistence”; 09 2017. p. 129–133.
- [3] Liran Tal SM., “npm passes the 1 millionth package milestone! What can we learn?”; June 2019 [cited 1 November 2020], [Online]. Available: <https://snyk.io/blog/npm-passes-the-1-millionth-package-milestone-what-can-we-learn/>.
- [4] Kula RG., German DM., Ouni A., Ishio T., Inoue K., “Do Developers Update Their Library Dependencies?”, Empirical Software Engineering. Feb. 2018;23(1): 384–417, [Online]. Available: <https://doi.org/10.1007/s10664-017-9521-5>.
- [5] East T., “2020 GitHub Universe Micro-Mentoring Application”; October 2020 [cited 1 November 2020], [Online]. Available: <https://github.blog/2020-10-27-2020-github-universe-micro-mentoring-application/>.
- [6] Synopsys, “Heartbleed Bug”; June 2020 [cited 1 November 2020], [Online]. Available: <http://heartbleed.com/>.
- [7] Bennett JT., “Shellshock in the Wild”; September 2014 [cited 1 November 2020], [Online]. Available: <https://www.fireeye.com/blog/threat-research/2014/09/shellshock-in-the-wild.html>.
- [8] Chinhanet B., Kula RG., McIntosh S., Ishio T., Ihara A., Matsumoto K.. “Lags in the Release, Adoption, and Propagation of npm Vulnerability Fixes”; 2020.
- [9] Yano Y., Kula R., Kula T., Ishio K., Inoue K., “VerXCombo: An Interactive Data Visualization of Popular Library Version Combinations”; 05 2015. .

- [10] Fatih Erikli AK. Burak Arikan, “NPM Dependency Network”; [cited 9 November 2020], [Online]. Available: <https://graphcommons.com/graphs/a7ec343d-2a0c-47bb-9658-bb8315e8a096?auto=trueshow=analysis-cluster>.
- [11] Security R., “Vulnerable Dependencies”; [cited 1 November 2020], [Online]. Available: <https://ropesec.com/articles/vulnerable-dependencies/>.
- [12] W3School, “JSON - Introduction”; [cited 1 November 2020], [Online]. Available: https://www.w3schools.com/js/js_json_intro.asp.
- [13] MongoDB, “What Is MongoDB?”; [cited 1 November 2020], [Online]. Available: <https://www.mongodb.com/what-is-mongodb>.
- [14] Bostock M., “D3 Data-Driven Documents”; [cited 1 November 2020], [Online]. Available: <https://d3js.org/>.
- [15] Bostock M., “Directional Force Layout Diagram”; [cited 9 November 2020], [Online]. Available: <https://gist.github.com/d3noob/5141278>.
- [16] Docs G., “About GitHub Security Advisories”; [cited 14 May 2021], [Online]. Available: <https://docs.github.com/en/code-security/security-advisories/about-github-security-advisories>.
- [17] Thompson B., “Applying machine intelligence to GitHub security alerts”; [cited 24 May 2021], [Online]. Available: <https://github.blog/2018-10-09-applying-machine-intelligence-to-security-alerts/leveraging-the-community>.
- [18] Ko A., LaToza T., Burnett M., “A practical guide to controlled experiments of software engineering tools with human participants”, *Empirical Software Engineering*. 02 2013;20.
- [19] Gopstein YDZY. Iannacone, Cappos, “Understanding Misunderstandings in Source Code”, *European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 09 2017;.

- [20] Zeke Sikelianos FA. Ionică Bizău, “nice-registry/all-the-package-names”; [cited 19 April 2021], [Online]. Available: <https://github.com/nice-registry/all-the-package-names>.
- [21] Docs G., “Searching code”; [cited 19 April 2021], [Online]. Available: <https://docs.github.com/en/github/searching-for-information-on-github/searching-code>.

BIOGRAPHIES

NAME	Miss. Vipawan Jarukitpipat
DATE OF BIRTH	11 January 1999
PLACE OF BIRTH	Bangkok, Thailand
INSTITUTIONS ATTENDED	Mahidol University International Demonstration School, 2016: High School Diploma Mahidol University, 2021: Bachelor of Science (ICT)
NAME	Miss. Wachirayana Wanprasert
DATE OF BIRTH	12 November 1998
PLACE OF BIRTH	Loei, Thailand
INSTITUTIONS ATTENDED	Loeipittayakom School, 2016: High School Diploma Mahidol University, 2021: Bachelor of Science (ICT)
NAME	Mr. Clinton Chhun
DATE OF BIRTH	28 February 1999
PLACE OF BIRTH	Phnom Penh, Cambodia
INSTITUTIONS ATTENDED	Bak Touk High School, 2016: High School Diploma Mahidol University, 2021: Bachelor of Science (ICT)