Improving Vulnerability Assessment through Multiple Vulnerability Sources

GUSTAV SVENSSON
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Improving Vulnerability Assessment through Multiple Vulnerability Sources

Gustav Svensson
dat12gs1@lth.se

Department of Electrical and Information Technology
Lund University

Supervisor: Martin Hell

Examiner: Thomas Johansson

October 6, 2020

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

Finding vulnerabilities in open source code is getting more important with the increasing use of open source. The National Vulnerability Database (NVD) provides a database for public vulnerabilities, or CVEs (Common Vulnerabilities and Exposures), which is a standard for identifying vulnerabilities. NVD is the most common used source for vulnerabilities but there exists other vulnerability sources that often are for specific programming languages or package managers. The package manager Node Package Manager (NPM) has its own vulnerability database, or security advisory as you also can call it. Many of the vulnerabilities on the NPM security advisory overlap with the CVEs on NVD, but there are vulnerabilities that do not exist on NVD, and vice versa. In this thesis I will do a comparison of NVD and the NPM security advisory by looking at the vulnerabilities that overlap and see what information that differ, and also see how many vulnerabilities that only exist on one of the sources. The mapping of the vulnerabilities will be done by looking at their third-party references, and if they have common references they can be mapped to each other. It will also be investigated if vulnerabilities are published earlier on one of the sources. The goal is to find if it is best to use NVD in combination with the NPM security advisory.

Glossary and Abbreviations

NVD - National Vulnerability Database
NPM - Node Package Manager
CVE - Common Vulnerabilities and Exposures
CWE - Common Weakness Enumeration
CVSS - Common Vulnerability Scoring System
CPE - Common Platform Enumeration

Popular Science Summary

Vulnerabilities in open source code is among the top 10 of the most seen web application security risks. The usage of open source is increasing and therefore it is getting more important to find vulnerabilities. To find vulnerabilities in open source code one can use public sources like the National Vulnerability Database (NVD) and manually find vulnerabilities there and then update the affected open source components. There also exists automatic tools that can find vulnerabilities for you, for example the package manager Node Package Manager (NPM) has a terminal command for this, `npm audit`. This command gives a report of all the known vulnerabilities in your used NPM packages. NVD makes it possible to create similar tools by using their data feeds or API for fetching the vulnerabilities. NPM has their own source for vulnerabilities which is called the NPM security advisory. There are vulnerabilities that exist on both NVD and the NPM security advisory, but it is not always clear that they are the same vulnerability. It is also possible for vulnerabilities to only exist on one of the sources. The information available for the vulnerabilities that overlap can differ, and it might be a good idea to use both NVD and the NPM security advisory in combination with each other in order to find as much information as possible, and to find other vulnerabilities that only exist on one of the sources. Vulnerabilities on NVD and the NPM security advisory can be mapped to each other by looking at their third-party references, if they have a common reference they can be mapped to each other. An investigation of how many vulnerabilities from NVD and NPM could be mapped to each other was made. Then a comparison of the mapped vulnerabilities was done to see which important information that differ.

We fetched 691 vulnerabilities affecting `node.js` from NVD and 990 advisories from NPM and it was found that 50,5% of the CVEs could be mapped to 35,6% of the advisories. It was also found that there exists critical vulnerabilities on only one of the sources, 9,6% of the CVEs were critical and could not be mapped to an advisory. 6,7% of the advisories were critical and could not be mapped to a CVE. For the mapped advisories it was found that the rating is not always the same and there exists critical vulnerabilities where the vulnerability from the other source has a much lower rating. We also looked at the published dates and the result was that 81,3% of the mapped advisories was published earlier than its mapped CVE. The biggest published date difference found between mapped vulnerabilities was 6 years, where the vulnerability was published on NPM first.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Background

The usage of open source components in software is rising and many of them have security flaws that are undetected. According to OWASP (The Open Web Application Security Project) vulnerabilities in open source and third party components are among the top 10 of the most seen web application security risks [1]. Open source code can save a lot of money and time for companies, but they must do regular security checks, and patch or update the software, in order to avoid security vulnerabilities. Failing to do so will increase the risk of introducing vulnerabilities in commercial software. This is a big risk for several industries. 96% of commercial codebases use open source software, and over 60% of these contained vulnerabilities [2].

One famous security vulnerability is the Heartbleed bug, which affected the OpenSSL cryptography library. The bug was introduced in 2012 and it would remain there undiscovered for two years. In April 2014 the bug was discovered, and at that time it was believed that around 17% (around half a million) of the internet's secure web servers were vulnerable. The bug made it possible to obtain sensitive data, passwords or the server's private key for example. The first fixed version was released a few days after the discovery. But as mentioned before, companies need to keep track of the versions in their third-party open source components themselves, and as of June 2014, around 300 000 public web servers were still vulnerable to the Heartbleed bug [3]. On the 2nd of May 2014, 1291 sites out of the top 150 000 SSL/TLS enabled sites were vulnerable. On the 6th of April 2020, there were 46 websites out of these 150 000 that are still vulnerable to the Heartbleed bug [4]. Early detection and better monitoring of potential vulnerabilities is therefore very important for companies today, but can be very time consuming if done manually.

The National Vulnerability Database (NVD) provides a source for publicly known vulnerabilities, which enables the possibility to have automated monitoring of vulnerabilities [5]. Automated detection of vulnerabilities makes it easier for companies to discover vulnerabilities and prevent them from making any harm. But not all vulnerabilities are published on NVD and many of them are published on other sources before they reach NVD. These sources are for example GitHub repositories with security advisories or package managers' own security advisories.

For example the JavaScript package manager for node.js packages, Node Package Manager (NPM), has its own security advisory [6]. These sources could be used in order to find vulnerabilities earlier or to find any other information that is not available on NVD.

1.2 Project Goal

In this thesis it will be investigated if the NPM security advisory should be used over or in combination with NVD in order to find more information about vulnerabilities that affects NPM packages. There are vulnerabilities that exist on both sources, but it is not clear that they are the same vulnerability. A mapping of vulnerabilities on NVD and NPM will be done and it will be investigated if there is extra information on one of the sources which can motivate to use both sources in combination with each other. It will also be investigated if there are vulnerabilities that only exist on one of the sources. This will be done by fetching vulnerabilities from NPM by using an automatic method called web scraping. These vulnerabilities will then be mapped to vulnerabilities on NVD. Then a comparison of the vulnerabilities on the two sources will be done.

The main questions to be answered are:

- Should the NPM security advisory be used over or in combination with NVD in order to find more information about vulnerabilities affecting `node.js` packages?
- Are there vulnerabilities that do not exist on NVD, but exist on NPM, and vice versa?

1.3 Related Work

There is a lot of work about vulnerabilities in open-source dependencies and some of the related work to this thesis is presented below.

1.3.1 Timeline Investigation of NPM Vulnerabilities

In [7], the authors analyse vulnerabilities from the NPM security advisory, and investigates the timeline for these vulnerabilities.

Specifically, they find when vulnerabilities are introduced, discovered, fixed and published. They discuss reasons to when in time these events occur, for example that the majority of vulnerabilities are fixed before they are published. This is similar to the investigations we will do in this thesis but we will look at the published dates on vulnerabilities from NVD and the NPM security advisory and then investigate if they are published earlier on one of the sources.

1.3.2 Information Availability of Vulnerabilities

In [8], the authors also analyse the timeline of vulnerabilities. They analyse how available information about the timeline of a vulnerability is on publicly available

sources, and suggests improvements on this. Also, they suggest improvements concerning what information is available and that combining different vulnerability sources could be a good idea to have all information about a vulnerability more easily accessible.

1.4 Structure

Chapter 1 explains the goal of the thesis, presents the questions to be answered and other related work. In Chapter 2 relevant background knowledge for reading the thesis is given. Chapter 3 explains how the fetching and mapping of vulnerabilities from NVD and the NPM security advisory was done. Chapter 4 will present the result of the mapping and then show the result of the comparison of the two vulnerability sources and Chapter 5 will discuss the results. And finally, Chapter 6 concludes the thesis and suggests future research on the subject.

Chapter 2
Background

This chapter will explain relevant background to understand the content of this thesis.

2.1 Software Vulnerabilities and Dependencies

2.1.1 Software Vulnerability

A software vulnerability is defined as a weakness in publicly released software that can be exploited by a malicious actor [9]. Most software vulnerabilities are enumerated and identified through the Common Vulnerabilities and Exposures (CVE, see Section 2.1.3) identifiers, but not all of them. Some vulnerabilities can be found in security advisories from software vendors where they have their own identifier, but can be linked to a CVE.

2.1.2 Software Dependency

A software dependency, can also be called package, is a piece of software that can be imported to another software. The software is then dependent on the dependency in order for it to work properly.

In software projects, dependencies are usually managed by so called package managers. The package manager is responsible for installing all dependencies for a software project. Which versions of the dependencies that should be installed is usually specified [10].

Package managers are usually grouped by language, one example is the PHP package manager Composer [11]. Another example is the JavaScript package manager NPM. NPM has both public and paid-for private packages in its registry, which is called the NPM registry [12]. The public part of the registry is open source [13].

When a vulnerability is introduced in a software it is because a vulnerable dependency was imported. The vulnerabilities are usually identified by using CVEs and affects certain versions of dependencies. A way of identifying software dependencies is by the Common Platform Enumeration (CPE), see Section 2.1.8.

2.1.3 CVE

CVE is a list of publicly known vulnerabilities with CVE identifiers (CVEs) which is provided by the Mitre Corporation [14]. Vulnerabilities are given identifiers by a CVE Numbering Authority (CNA) and Mitre act as the primary CNA, but other organizations can act as their own CNA, e.g. Microsoft. As of April 6th 2020 there are 118 organizations spread over 21 countries that are authorized to assign CVE identifiers [15]. CVEs affect specific software dependencies and is only for software that has been publicly released, including commercial software. NVD has a list of CVEs which is synchronized with Mitre, but with additional information, see Section 2.1.4.

2.1.4 NVD

NVD is a database for software vulnerabilities provided by the National Institute of Standards and Technology (NIST), which is part of the United States Department of Commerce [16]. NVD is synchronized with Mitres CVE list, but compared to Mitre, NVD provides more in-depth information about CVEs. The information provided by NVD for every CVE is listed below. Figure 2.1 shows how it looks on NVD for `CVE-2020-8125`, excluding software configurations, see Figure 2.6 for this.

- The CVE identifier.
- The date the CVE was published on NVD.
- The date the CVE was last modified on NVD.
- A description about the vulnerability.
- A CWE identifier and its CWE name, this is a category which the vulnerability belongs to, see more Section 2.1.7.
- The affected software configurations, see Section 2.1.8 about CPEs.
- CVSS information (see Section 2.1.9):
  - The CVSS base score, a severity score for the vulnerability.
  - The CVSS impact subscore, a subscore used to calculate the base score.
  - The CVSS exploitability subscore, a subscore used to calculate the base score.
  - The base score metric values, different metrics used to calculate the scores above.
- Third-party references, external links where more information about the vulnerability is provided. Can be a reference to a security advisory, the NPM security advisory for example.

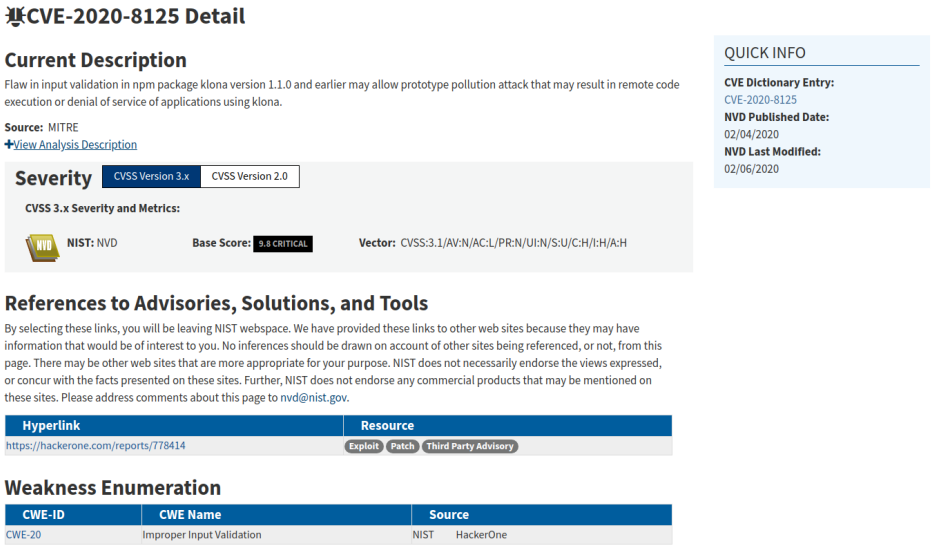

Figure 2.1: Screenshot of CVE-2020-8125 from NVD [17].

2.1.5 The NPM Security Advisory

There exist other sources for vulnerabilities, such as security advisories from either the vendor or developers. A security advisory is like a vulnerability but reported by for example the package manager corporation. As mentioned before, NPM has its own security advisory, which is a list of vulnerabilities affecting JavaScript dependencies in the NPM registry. The data that exists for each advisory is listed below. Figure 2.2 and 2.3 shows how an advisory looks like on the NPM security advisory.

- The title of the vulnerability.
- The date the advisory was published.
- The date the advisory was reported to NPM.
- An overview (description) of the advisory.
- The name of the NPM dependency that is affected.
- A remediation, e.g. what version the affected dependency has to be updated to in order for it to be secure.
- Which versions of the dependency that are affected and unaffected.
- Third-party references, such as Github issues where the vulnerability was discovered or a reference to a CVE.
- The severity of the vulnerability (low, moderate, high, critical).
- A status, if it has been patched or not.

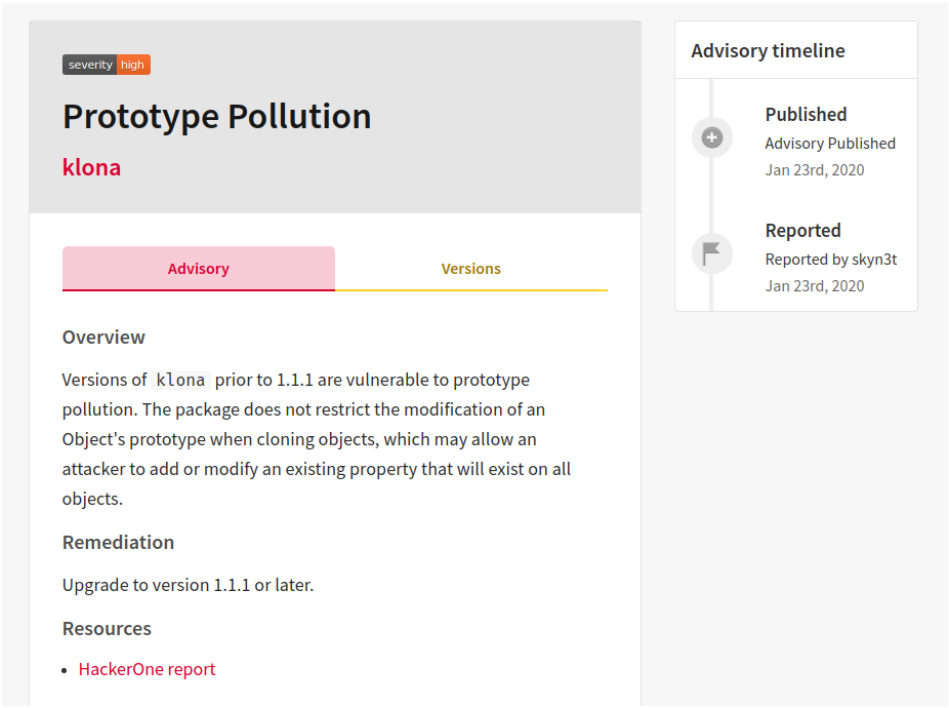

Figure 2.2: Screenshot of a security advisory on NPM [18].

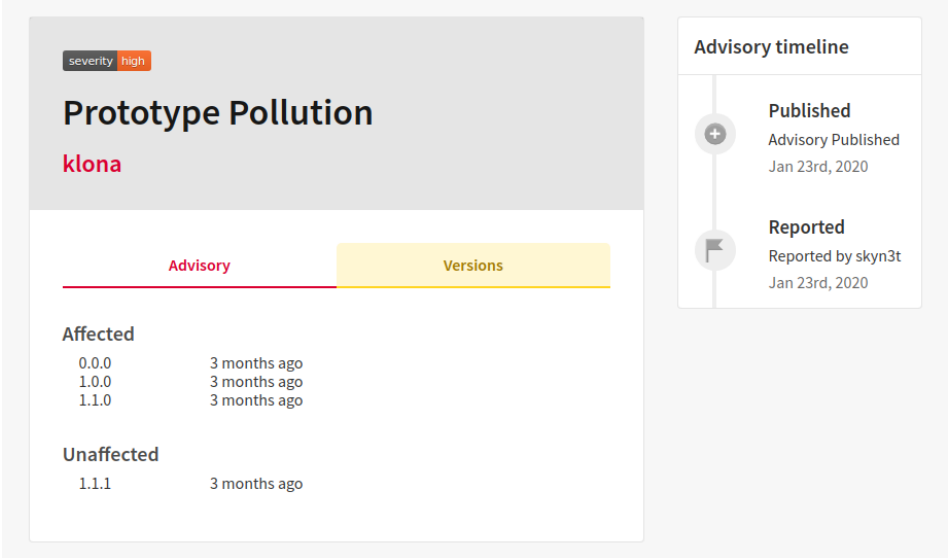

Figure 2.3: Screenshot of the affected/unaffected versions from a security advisory on NPM [19].

All advisories in this thesis will be fetched from the NPM security advisory with a method called web scraping [20].

2.1.6 Debricked

Debricked is a company which provides a tool which companies and developers can use for detecting vulnerabilities in their imported code automatically. Debricked's main source for vulnerabilities is NVD [21].

Debricked fetches all the data that is listed in Section 2.1.4 and saves it in their own vulnerability database. The CVEs that are fetched in this thesis will be from Debricked.

2.1.7 CWE

The Common Weakness Enumeration (CWE) is a system for categorising vulnerabilities [22]. A weakness is an error in software, which might lead to a vulnerability. As with CVE, CWE is also a list, of different weaknesses, provided by the Mitre Corporation. Every CWE has an identifier and a description that describes the type of the vulnerability. The top 1 most dangerous software weakness from 2019 is `CWE-119` [23]. Figure 2.4 shows the identifier, title and description of this CWE.

```
CWE: CWE-119

Title: Improper Restriction of Operations within the
Bounds of a Memory Buffer

Description: The software performs operations on a
memory buffer, but it can read from or write to a
memory location that is outside of the intended
boundary of the buffer.
```

Figure 2.4: CWE-119 with title and description [24].

2.1.8 CPE

The Common Platform Enumeration (CPE) is a standardized method of describing and identifying IT products, software dependencies for example [25]. They can be identified by using their CPE name, which uses a format called Well-Formed Names (WFN) [26]. The WFNs can be presented as a string, in Figure 2.5 there is an example of the software dependency `klona` as a CPE.

```
cpe:2.3:a:klona_project:klona:*:*:*:*:node.js:*:*
```

Figure 2.5: Example of a CPE name in its string format.

The first part of the string, "cpe:2.3" describes which version of the CPE specification that is used, the rest of the CPE string consists of the following attributes:

- part, this can have the following values:
  - a, when the CPE concerns an application, this will be the case for software dependencies.
  - o, when the CPE concerns an operating system.
  - h, when the CPE concerns hardware devices.
- vendor, this describes or identifies the person or organization that created the product.
- product, this describes or identifies the product.
- version, this is an alphanumeric string which describes the release version of the product.
- update, this is an alphanumeric string which describes the particular update (patch) of the version. For example 1.23a.
- edition, this is deprecated in version 2.3 of the CPE specification. Has the value ANY, which is "*" in the string format.
- language, this describes the language that is used in the user interface of the product.
- sw_edition, this describes if the product is tailored to a particular market or class of end users.
- target_sw, this describes the software computing environment that the product operates within, node.js for example.
- target_hw, this describes the instruction set architecture on which the product operates on, for example x86.
- other, this describes any other identifying information which is vendor or product specific, and does not fit in any other attribute.

These are in the same order as in the example CPE in Figure 2.5, which has the part a, the vendor klona_project, the product klona and the target software node.js.

NVD provides a CPE dictionary, which contains all known CPE names. A CPE name can be mapped to a CVE if the CPE is vulnerable. For example the CPE in Figure 2.5 is affected by CVE-2020-8125. Figure 2.6 shows the affected software configurations list for CVE-2020-8125, this is the same vulnerability as in Figure 2.1.

Figure 2.6: Affected software configurations for CVE-2020-8125.

A software configuration is something used by NVD on their vulnerability details page, which is a combination of software that are affected by the vulnerability [27]. For a vulnerability, there is a list of affected software configurations and NVD uses CPE strings to represent the software. The affected versions of the CPEs are also specified, this can either be a specific version or a version range. For `CVE-2020-8125` the affected version range is `<= 1.1.0`, version `1.1.0` is the last affected version. An affected CPE can have both start version and end version. The different version range assignments are listed below.

- <= up to, including the specified version
- < up to, excluding the specified version
- >= from, including the specified version
- > from, excluding the specified version

2.1.9 CVSS

The Common Vulnerability Scoring System (CVSS) is a method for assigning severity to vulnerabilities (CVEs) [28]. The score ranges from 0 to 10 with 10 being the most severe. The scores are grouped in to different ratings in order to have a text representation of them, see Table 2.1. The score is calculated based on several metrics such as how you exploit it or how easy it is to exploit the vulnerability. In version 3 of CVSS the metrics are divided into three groups, the *exploitability metrics*, *scope metric* and *impact metrics*.

Table 2.1: Table of CVSS score ratings.

Rating	Base score
None	0
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10.0

Exploitability Metrics

These metrics describe how easy it is to exploit the vulnerability.

- Attack Vector (AV), this describes how a vulnerability can be exploited. This value, and the base score, will be larger the more remote an attacker can be, both physically and logically. It can have the following values:
  - Network (N), the vulnerability is "remotely exploitable". The attacker does not have to be on the same network, which means that the possible set of attackers include the entire internet. For example a denial of service (DoS) attack is a type of network attack.
  - Adjacent (A), the attacker must be on the same physical network as the vulnerable component to exploit the vulnerability.
  - Local (L), the vulnerability can not be exploited over a network. The attacker must exploit it either locally (e.g. by keyboard) or through user interaction, for example by social engineering, tricking the user into doing something malicious.
  - Physical (P), the attacker must physically touch or manipulate the vulnerable component in order to exploit the vulnerability.
- Attack Complexity (AC), this describes how easy or difficult it is to exploit the vulnerability. It describes conditions that is beyond the attacker's control which must exist in order to exploit the vulnerability, collecting information about the target is one example. The base score will be greater the less complex the attack is. This metric can have the following values:
  - Low (L), this means there are no special conditions for exploiting the vulnerability.
  - High (H), a successful exploit depends on conditions beyond the attacker's control. The attacker must be prepared in order to successfully exploit the vulnerability.
- Privileges Required (PR), this describes the level of privileges an attacker must have before executing a successful attack. If no privileges are required the base score will be greatest. This metric can have the following values:
  - None (N), the attacker is unauthorized before the attack. Which means the attacker does not need access to any settings or files in the vulnerable system in order to exploit the vulnerability.
  - Low (L), the attacker has low authorization. Requires access to files and settings that are non-sensitive.
  - High (H), the attacker has high authorization, e.g. administrative access of the vulnerable system.
- User Interaction (UI), this metric describes if any interaction from a user that is not the attacker is required in order to exploit the vulnerability. If no user interaction is required the base score will be higher. It can have the following values:
  - None (N), the vulnerability can be exploited without any interaction from any user.

  - Required (R), it is required that an user take some action before the vulnerability can be exploited, for example an exploit may only be possible during the installation of an application by a system administrator.

Scope Metric

Scope (S), this metric group describes if a vulnerability in one vulnerable component affects other components that is not in the same security scope. For example the resources an application has access to, e.g. files and other resources, are under one security scope. The application is called a security authority, and other examples of security authorities are operating systems or firmwares. If the impact of a vulnerability affects components outside of the security scope, a scope change occurs. If a scope change occurs the base score will be higher. It can have the following values:

- Unchanged (U), the vulnerability only affects resources managed by the same security authority.
- Changed (C), the vulnerability can affect resources that is beyond the vulnerable component's security scope.

Impact Metrics

These metrics describes the severity of the consequences of a successful exploit.

- Confidentiality (C), this metric measures the impact to the confidentiality of the resources managed by the vulnerable component. The base score will be higher if the confidentiality value is high. The possible values is listed below:
  - High (H), total loss of confidentiality. All resources within the impacted component are disclosed to the attacker.
  - Low (L), some loss of confidentiality. The attacker gets access to some restricted information, but can not control over what is obtained.
  - None (N), there is no loss of confidentiality.
- Integrity (I), this metric measures the impact to integrity of the vulnerable component, in other words, if the attacker is able to modify data. The base score will be higher if the integrity value is high. The possible values is listed below:
  - High (H), total loss of integrity. The attacker can modify all files protected by the vulnerable component.
  - Low (L), modification of data is possible, but the attacker can not control the consequences of modification.
  - None (N), there is no loss of integrity.

- Availability (A), this metric refers to the loss of availability of the vulnerable component itself. For example an attack that consumes network bandwidth has some impact on the availability of the vulnerable component. The base score will be higher if the availability value is high. The possible values is listed below

  - High (H), total loss of availability. The attacker is able to fully deny access to resources in the vulnerable component.
  - Low (L), resources are partially available. The attacker can not completely deny access.
  - None (N), there is no loss of availability.

The metrics can be presented by a vector string used to determine the base score. An example of a vector string is shown in Figure 2.7. Every metric in the vector string is represented by an abbreviation of the metric name and its value, separated by a colon. For example "AV:N", which is "Attack Vector: Network".

```
CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:L/I:L/A:N
```

Figure 2.7: Example of a CVSS vector string.

The vector string has the following metric values:

- Attack Vector (AV): Network (N)
- Attack Complexity (AC): Low (L)
- Privileges Required (PR): High (H)
- User Interaction (UI): None (N)
- Scope (S): Unchanged (U)
- Confidentiality (C): Low (L)
- Integrity (I): Low (L)
- Availability (A): None (N)

The scores can be calculated by using NVDs CVSS score calculator [29]. The metrics above results in the following scores:

- CVSS Base Score: 3.8
- Impact Subscore: 2.5
- Exploitability Subscore: 1.2

2.1.10 Severity

CVEs have CVSS scores and advisories have something similar which is called severity. The severity is based on the impact and exploitability of the vulnerability in its most common use case [30]. The severity can have four different values, and each of them have a recommended action. See Table 2.2. No more information on what the severity is based on is proved by NPM and we could not find any other information on what metrics the severity is based on.

Table 2.2: Table of severity ratings.

Severity	Recommended action
Low	Address at your discretion
Moderate	Address as time allows
High	Address as quickly as possible
Critical	Address immediately

Chapter 3
Methodology

In this chapter Section 3.1 will explain how the fetching of the CVEs and advisories was done. Then in Section 3.2 we will go through how the mapping between CVEs and advisories was done, and which datasets were created.

3.1 Scraping Vulnerabilities

3.1.1 NPM Security Advisory

In order to scrape the NPM security advisory we had to go through it manually and see which data was available and how the vulnerabilities could be automatically fetched. After deciding on which data that was going to be scraped, the actual scraper could be implemented. The scraper was written in Python and made continuous requests to the NPM advisory site and scraped the data. All of the advisories scraped from the NPM security advisory were scraped before the 18th of May 2020. There will not be any advisories with a publication date after this date. There are over 1 million dependencies in the NPM registry [31] and there are in total 1344 advisories in the NPM security advisory with a published date before the 18th of May 2020, affecting 1149 unique dependencies.

Some vulnerable dependencies have the title "Malicious Package", these will not be analysed, because they do not affect real dependencies. They only have names that are very similar to the real dependency, but contains malicious code. This is called typosquatting [32]. All advisories were saved in a local database, see the structure in Figure 3.4.

NVD

For all CVEs from NVD, we wanted to find those CVEs that affects `node.js` dependencies and/or had a reference to the NPM security advisory. The Debricked database was queried for all CVEs that had a reference to the NPM security advisory or had `node.js` as target software in the CPE string. This corresponds to CVEs that affects CPEs that has a CPE string on the formats shown in Figure 3.1. All CVEs were saved in a local database, see the structure in Figure 3.4.

Since all the advisories were scraped before the 18th of May 2020, all CVEs fetched from NVD also have a publication date before this date.

```
cpe:2.3:*:*:*:*:*:*:*:*:node.js:*:*
cpe:2.3:*:*:*:*:*:*:*:*:nodejs:*:*
```

Figure 3.1: CPEs with node.js as target software.

Examples Scraped Vulnerabilities

In Figure 3.2 there is an example of all the data scraped for a vulnerability from the NPM security advisory. The vulnerability has the identifier 1463 from NPM but in this thesis all advisories will be prefixed with "npm-" before the identifier, for example npm-1463.

```
Advisory: npm-1463

Title: Prototype Pollution

Package: klona

Affected versions: 1.0.0, 1.1.0

Unaffected versions: 1.1.1

Published date: 2020-01-23

Remediation: Upgrade to version 1.1.1 or later.

Severity: High

Description: Versions of klona prior to 1.1.1 are
vulnerable to prototype pollution. The package does
not restrict the modification of an Object's prototype
when cloning objects, which may allow an attacker to add
or modify an existing property that will exist on all objects.

Third-party references: https://hackerone.com/reports/778414
```

Figure 3.2: Vulnerability npm-1463, this is the same vulnerability as in Figure 2.2.

In Figure 3.3 there is a example of all the data that was fetched for a CVE from the Debricked database. Note, this is the same vulnerability as the advisory in Figure 3.2 but with data from NVD. Also note that they both have the same third-party reference, but no direct reference to each other.

```
CVE: CVE-2020-8125

Package: klona_project/klona

Published date: 2020-02-04

Updated at: 2020-02-06

CVSS3 score: 98

Vector string: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CWE: CWE-20

CWE category: Improper Input Validation

Description: Flaw in input validation in NPM package
klona version 1.1.0 and earlier may allow prototype pollution
attack that may result in remote code execution or denial of
service of applications using klona.

Affected software configurations:
cpe:2.3:a:klona_project:klona:*:*:*:*:*:node.js:*:* (<= 1.1.0)

Third-party references: https://hackerone.com/reports/778414
```

Figure 3.3: Vulnerability CVE-2020-8125, this is the same vulnerability as in Figure 2.1.

3.2 Mapping Vulnerabilities between NVD and NPM

In order to find which advisories might exist on NVD and vice versa, we wanted to find those advisories with references to either NVD or Mitre's CVE list, and CVEs with references to the NPM security advisory. Also, all advisories and CVEs which had no direct references to each other, can be mapped if they have any common reference or a so called indirect reference. In order to create a mapping between advisories and CVEs the following was done:

- All advisories and CVEs from NVD with any common reference were mapped to each other, by doing a join on the references in the simplified MySQL database we created. The common reference is an exact match. See Section 3.2.1.

- For those advisories which had no common references with CVEs, it is still possible to map it to a CVE by looking at the third-party references. There might exist references to CVEs in the third-party references and we can call this an indirect reference to a CVE. See Section 3.2.2

3.2.1 Mapping by Common Reference

Figure 3.4 shows a simplified database structure that the scraped advisories and CVEs from NVD was saved in. In order to map advisories and CVEs on common reference the join was done on the `npm_reference` and `nvd_reference` field. See Figure 3.5 for the complete SQL code.

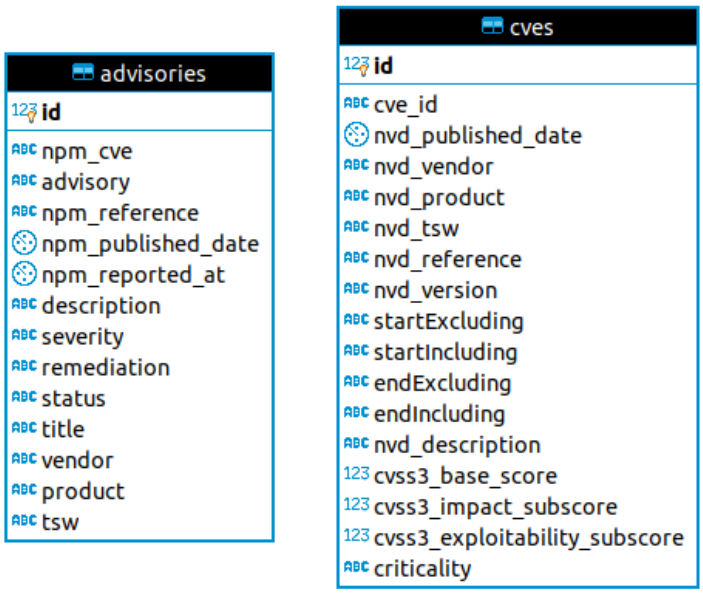

Figure 3.4: Simplified database structure of the fetched advisories and CVEs.

```
SELECT
advisory.npm_cve,
cve.cve_id,
advisory.npm_reference,
cve.nvd_reference
FROM cves.advisories advisory
INNER JOIN cves.cves AS cve
ON advisory.npm_reference = cve.nvd_reference;
```

Figure 3.5: Database query used for joining advisories and CVEs on common reference.

NVD had 244 references to a domain called `nodesecurity.io`. References with this domain redirects to `npmjs.com`. This is because the nodesecurity platform, which was a catalogue of JavaScript vulnerabilites, was aquired by npmjs in 2018 [33]. The vulnerabilities now exist on `npmjs.com` instead, with the same identifier (e.g. `https://nodesecurity.io/advisories/525` corresponds to `https://npmjs.com/advisories/525`, where 525 is the identifier for the advisory). All `nodesecurity.io` domains were replaced with `npmjs.com` domains and this is reflected in Table 4.2.

Some references included "`www`" in the reference or "`https://www`" and some did not. In order to get as many matches as possible the URLs were stripped so that they only contained the domain name and the path. For example `https://www.npmjs.com/advisories/10` was stripped to `npmjs.com/advisories/10`.

Figure 3.2 shows an example of how a NPM advisory with a HackerOne reference can look like and Figure 3.3 shows the corresponding CVE. As one can see they have the same third-party reference to HackerOne. It looks the same for all other advisories and CVEs that are mapped by common reference.

3.2.2 Mapping by Indirect Reference

When looking through some of the references from `github.com`, `hackerone.com` and `snyk.io` manually, it was found there might exist references to CVEs in them. If there was no common reference, there might exist a reference to a CVE by looking at the NPM advisory's third-party references. The mapping can then be done by scraping that third-party reference automatically or manually. Figure 3.6 shows how indirect references work. The CVE and advisory have no common references but the advisory has a reference that references a CVE, an indirect reference.

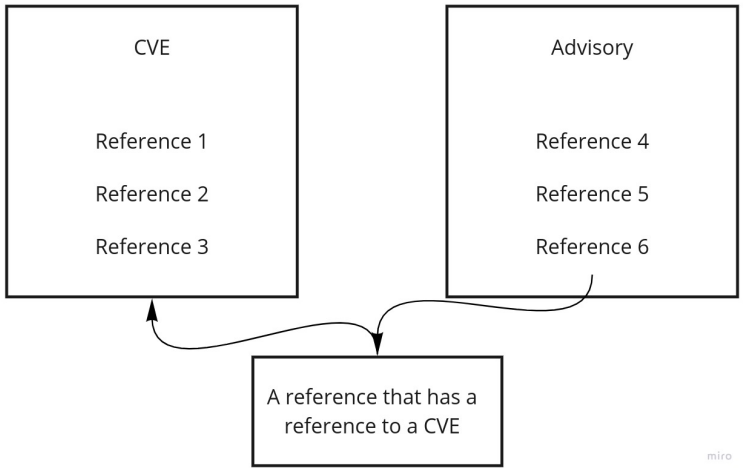

Figure 3.6: Visualisation of how indirect references work.

In order to create a mapping between advisories and CVEs through indirect references the following was done:

- For those advisories which had no common reference with a CVE on NVD but has a reference to a GitHub security advisory, there might be a reference to a CVE. This mapping was done manually.
- For those advisories which had no common reference with a CVE on NVD but has a reference to HackerOne, there might be a reference to a CVE. This mapping was done manually.
- For those advisories which had no common reference with a CVE on NVD but has a reference to Snyk there might be a reference to a CVE on Snyk. This mapping was done manually.
- For those advisories which had no common reference with a CVE on NVD but has a reference to `cve.mitre.org` we can be certain the advisory has a CVE. This mapping was done manually.

We chose to do the indirect mapping for these four type of references because they were the most common ones and had the highest chance of referencing a CVE.

HackerOne References

HackerOne is a site which has a so called bug bounty program where individuals can get recognition by reporting security vulnerabilities [34]. If there exists a CVE reference on HackerOne, the mapping to an advisory can be done. This was done manually because of problems with scraping HackerOne, due to JavaScript being blocked. Figure 3.7 shows how a HackerOne reference with a reference to a CVE looks like. Also, the date the vulnerability was reported to HackerOne tells us around when the vulnerability was discovered.

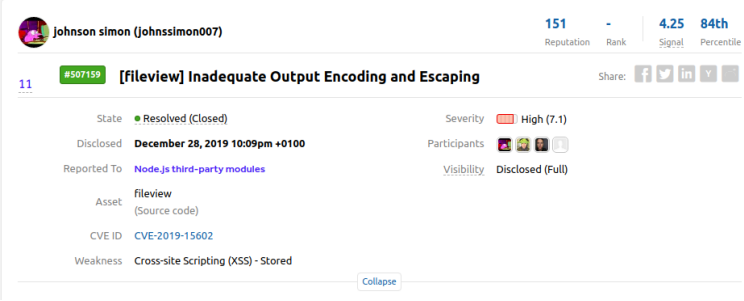

Figure 3.7: An example of a HackerOne advisory with a reference to CVE-2019-15602.

GitHub References

The references to GitHub could be a GitHub issue, pull request or a security advisory. A GitHub security advisory is a vulnerability that has been reported in a repository on GitHub, and which might have a CVE mapped to it. When the

vulnerability has been fixed it can be published and is then saved in the GitHub advisory database. GitHub is a CNA and is therefore authorised to assign CVE numbers to its security advisories [35]. Figure 3.8 shows an example of how a GitHub security advisory can look like with a reference to a CVE. GitHub issues can tell when a vulnerability was discovered by looking at when the issue was created, but might also contain references to CVEs. A GitHub pull request or commit might also contain a reference to a CVE.

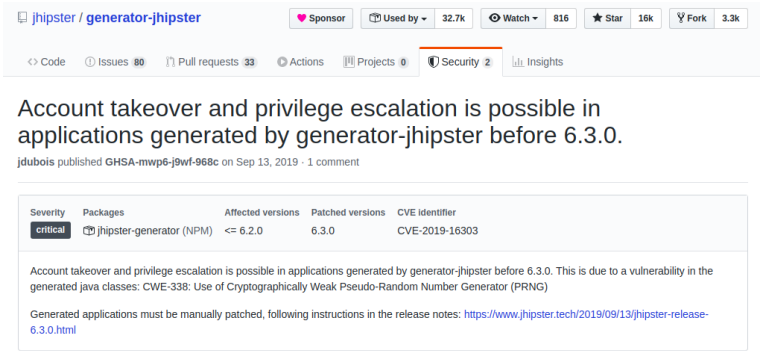

Figure 3.8: An example of a GitHub security advisory with a reference to CVE-2019-16303.

Snyk References

Snyk provides a vulnerability database where NPM vulnerabilities can be found [36]. Some security advisories from NPM have common Snyk references with CVEs from NVD. Figure 3.9 shows an example of this.

3.2.3 Created Datasets

When the mapping was done, three main datasets was created, see below. The datasets are visualised in Figure 3.10. All advisories and CVEs have a published date before 2020-05-18 and no advisory affects any malicious package.

- Dataset A. This dataset consists of all advisories from the NPM security advisory.
- Dataset B. This dataset consists of all CVEs that had `node.js` as target software or had a reference to the NPM security advisory.
- Dataset AB. This dataset consists of all advisories and CVEs that could be mapped to each other either by common reference or indirect reference.

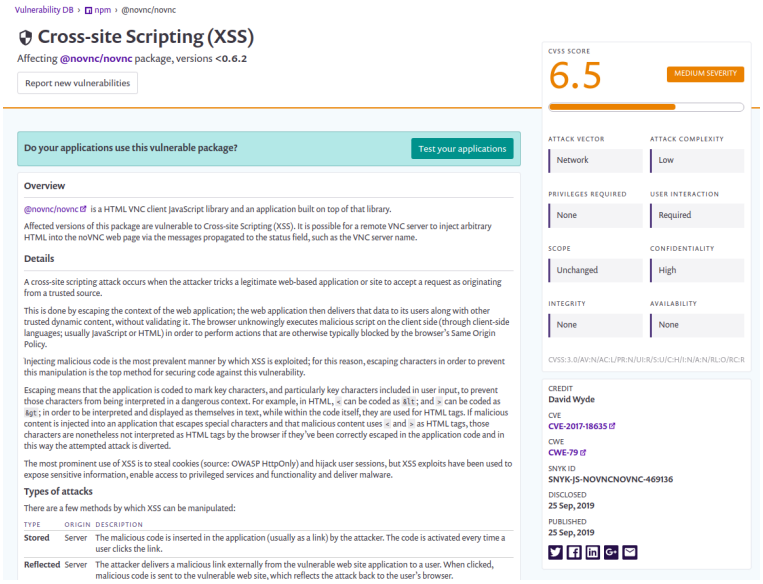

Figure 3.9: An example of a Snyk vulnerabilty with a reference to CVE-2017-18635.

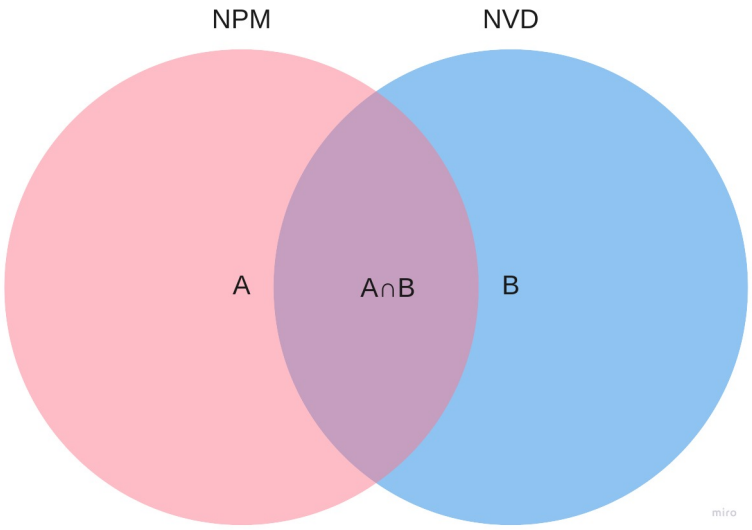

Figure 3.10: Visualisation of the datasets created.

Chapter 4
Result

This chapter will show the result of the mapping and the comparison of NVD and the NPM security advisory. See Section 4.1 and 4.2.

4.1 Mapping Vulnerabilities between NVD and NPM

In Section 4.1.1 the number of scraped and mapped CVEs and advisories is shown. Section 4.1.2 show the result of the mapping by common reference and Section 4.1.3 show the result of the mapping by indirect reference. Section 4.1.4 summarises the result.

4.1.1 Number of Advisories and CVEs

There are over 1 million dependencies in the NPM registry and there are in total 1344 advisories in the NPM security advisory with a published date before the 18th of May 2020, affecting 1149 unique dependencies. After filtering out all the advisories that affected a Malicious Package, 990 NPM advisories remained, affecting 802 unique `node.js` dependencies.

There are in total 144382 CVEs from NVD in the Debricked database, that were published before the 18th of May 2020. The number of CVEs that had `node.js` as target software or had a reference to the NPM security advisory was 691 (only 0.4% of all CVEs), affecting 622 unique `node.js` dependencies. The result of the scraping and mapping is visualised in Figure 4.1 and summarised in Table 4.1. As shown in Table 4.1, 38% of the scraped advisories could be mapped to a CVE and 54% of the fetched CVEs could be mapped to an advisory.

Table 4.1: Number of advisories and CVEs fetched with number of unique affected dependencies and the number of mapped vulnerabilities.

Dataset	Vulnerabilities	Unique affected dependencies	Mapped
All advisories (A)	990	802	352 (35,6%)
All CVEs affecting node.js (B)	691	622	349 (50,5%)

Some CVEs can belong to several advisories and vice versa, which is why the number of mapped CVEs is lower. One example is CVE-2018-3717 [37], which

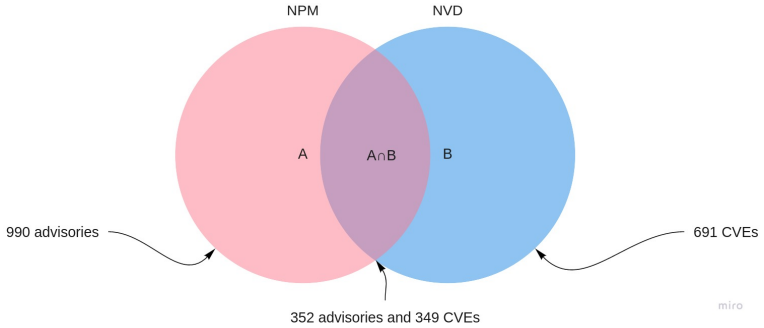

Figure 4.1: Venn diagram with number of advisories and CVEs in the three datasets.

belongs to both advisory npm-584 [39] and npm-595 [40]. The CVE has references to each of the advisories' HackerOne references.

4.1.2 Mapping by Common Reference

Out of the 990 advisories and 691 CVEs there were 347 advisories (35,1%) that could be mapped to 344 CVEs (49,8%) by common reference.

Table 4.2 shows how many advisories and CVEs that were mapped to each other by common reference grouped on the domain they were mapped on. Advisories and CVEs can be mapped on multiple references, which is the reason for the sum of the "Mapped" columns being greater than the actual number of mapped CVEs and advisories.

Table 4.2: Table of number of advisories and CVEs that were mapped to each other grouped on the domain they were mapped on.

Domain	Advisories	Mapped	CVEs	Mapped
npmjs.com	990	239 (24%)	267	239 (89,5 %)
github.com	422	49 (11,6%)	154	49 (31,8%)
hackerone.com	144	79 (54,9%)	121	77 (63,6%)
snyk.io	71	15 (21,1%)	75	14 (18,7%)
gist.github.com	6	1 (16,7%)	3	1 (33,3%)
blog.npmjs.org	3	3 (100%)	3	3 (100%)
auth0.com	2	2 (100%)	2	2 (100%)
sailsjs.org	1	1 (100%)	1	1 (100%)
timmclean.net	1	1 (100%)	1	1 (100%)
blog.intothesymmetry.com	1	1 (100%)	1	1 (100%)
blog.truesec.com	1	1 (100%)	1	1 (100%)
bugs.debian.org	1	1 (100%)	2	1 (50%)
cigital.com	1	1 (100%)	1	(100%)

The result shows that it is most common for advisories and CVEs to be mapped on common references to `npmjs.com`, `github.com`, `hackerone.com` and `snyk.io`.

As one can see 239 out of 267 (89,5%) CVEs with references to the domain `npmjs.com` could be mapped to an advisory on NPM. The other 28 references that leads to `npmjs.com` is to another page that is not `npmjs.com/advisories` or is a faulty url to `npmjs.com/advisories`. For example there are 8 references

to `npmjs.com/package`, which is the URL for NPM packages, not an advisory. CVE-2020-7636 [41] is one example, which has a reference to `npmjs.com/package/adb-driver`. An example with a faulty url is CVE-2015-1370 [42], which has a reference to `npmjs.com/advisories/marked_vbscript_injection` that leads to a 404 not found error.

One interesting thing to note is that there were only 49 (11.6%) of the advisories with GitHub references that could be mapped to a CVE. Table 4.3 shows the number of advisories and CVEs that were mapped on any reference, grouped by the domains they have in their references. The table shows that there are now more advisories with GitHub references that were mapped to a CVE. There were 94 advisories (22,3%) with GitHub references that were mapped on any common reference (compared to 49 that were mapped on a GitHub reference). This means that there were 45 advisories that had references to GitHub, but were mapped on another reference than GitHub.

Table 4.3: Table of number of advisories and CVEs that could be mapped to each other on any common reference, grouped by the domains in the vulnerabilities' references.

Domain	Advisories	Mapped on any reference	CVEs	Mapped on any reference
npmjs.com	990	347 (35%)	267	244 (91,4 %)
github.com	422	94 (22,3%)	154	89 (57,8%)
hackerone.com	144	81 (56,3%)	121	77 (63,6%)
snyk.io	71	17 (23,9%)	75	17 (22,7%)
gist.github.com	6	1 (16,7%)	3	3 (100%)
blog.npmjs.org	3	3 (100%)	3	3 (100%)
cve.mitre.org	6	2 (33,3%)	0	0 (0%)
auth0.com	2	2 (100%)	2	2 (100%)
sailsjs.org	1	1 (100%)	1	1 (100%)
timmclean.net	1	1 (100%)	1	1 (100%)
blog.intothesymmetry.com	1	1 (100%)	1	1 (100%)
blog.truesec.com	1	1 (100%)	1	1 (100%)
bugs.debian.org	1	1 (100%)	2	1 (50%)
cigital.com	1	1 (100%)	1	(100%)1
eslint.org	1	1 (0%)	0	0 (0%)
owasp.org	1	1 (0%)	0	0 (0%)
en.wikipedia.org	1	1 (0%)	0	0 (0%)
backbonejs.org	1	1 (0%)	0	0 (0%)
openwall.com	0	0 (0%)	8	8 (0%)
securityfocus.com	0	0 (0%)	8	8 (0%)
access.redhat.com	0	0 (0%)	6	6 (0%)
oracle.com	0	0 (0%)	4	4 (0%)
lists.opensuse.org	0	0 (0%)	4	4 (0%)
lists.fedoraproject.org	0	0 (0%)	4	4 (0%)
security.netapp.com	0	0 (0%)	3	3 (0%)
bugzilla.redhat.com	0	0 (0%)	2	2 (0%)
lists.apache.org	0	0 (0%)	2	2 (0%)
tenable.com	0	0 (0%)	2	2 (0%)
support.f5.com	0	0 (0%)	1	1 (0%)
raw.githubusercontent.com	0	0 (0%)	1	1 (0%)
packetstormsecurity.com	0	0 (0%)	1	1 (0%)
usn.ubuntu.com	0	0 (0%)	1	1 (0%)
mega.nz	0	0 (0%)	1	1 (0%)
jvn.jp	0	0 (0%)	1	1 (0%)
ibm.com	0	0 (0%)	1	1 (0%)
exchange.xforce.ibmcloud.com	0	0 (0%)	1	1 (0%)
bugs.chromium.org	0	0 (0%)	1	1 (0%)
app.snyk.io	0	0 (0%)	1	1 (0%)

One other interesting thing to note is that there are 6 advisories with references to `cve.mitre.org` and 2 of those were mapped on any common reference. Also there are domains in the references where either the advisories have none of the CVEs' references and vice versa. These domains are the ones with either 0 in the "Advisories" or "CVEs" column. This result shows that there are extra references in either advisories or CVEs that they necessarily are not mapped on.

Mapped GitHub References

Table 4.4 shows number of CVEs and advisories that were mapped on any common GitHub reference, grouped on the possible GitHub paths.

Table 4.4: Table of number of mapped CVEs and advisories that were mapped on a reference from GitHub.

Path	Advisories	Mapped	CVEs	Mapped
github.com/*/*/issues	133	24 (18%)	39	24 (61,5%)
github.com/*/*/commit	70	12 (17%)	71	12 (16,9%)
github.com/*/*/pull	69	15 (7,2%)	30	15 (50%)
github.com/nodejs/security-wg/*/vuln	14	0 (0%)	0	0 (0%)
github.com/*/advisories/*	28	4 (14,3%)	13	4 (30,1%)
github.com/*/blob/*	104	1 (0,9%)	26	1 (3,8%)
github.com/*/releases/*	9	0 (0%)	6	0 (0%)
other GitHub domains	37	0 (0%)	6	0 (0%)

The result shows that it is most common for advisories and CVEs to be mapped on GitHub issues. It is also more common for advisories to have references to GitHub issues. 18% of the advisories with references to GitHub issues could be mapped to a CVE, and 61,5% of the CVEs could be mapped to an advisory.

The references to `github.com/*/*/commit` and `github.com/*/*/pull` are to GitHub pull requests and commits. They were the second and third most common types of GitHub references to be mapped on.

The `github.com/nodejs/security-wg/*/vuln` paths are to a repository with NPM vulnerabilities [43], we can call this the NPM vulnerability repository from now on. There were 14 advisories with references to this domain, but not any CVE had this type of reference and therefore no advisory was mapped on this type of reference.

The references to `github.com/*/advisories/*` are to GitHubs own security advisory and the `github.com/*/blob/*` references links to a specific file in a repository. There were 4 out of the 28 advisories with references to GitHub advisories that could be mapped to CVEs. There were 104 advisories with references to specific files (`github.com/*/blob/*`) but only one of them could be mapped to a CVE.

Finally, the `github.com/*/releases/*` links to specific releases of a repository and the other GitHub domains are to other paths than the ones mentioned. For example, `github.com/snyk/zip-slip-vulnerability` is one of the domains included in "other GitHub domains". There were no CVEs or advisories that could be mapped on either releases or "other GitHub domains".

4.1.3 Mapping by Indirect Reference

For all the advisories that were not mapped we can try to map them indirectly through their third-party references. Table 4.5 shows the total number of advisories that have references to `github.com`, `hackerone.com`, `snyk.io`, `gist.github.com`, `cve.mitre.org` (these are the same numbers as the "Advisories" column in Table 4.2) with how many that have already been mapped by common reference. For those that have not been mapped, we tried to map them indirectly, how many this was is also shown in the table. We chose to do it for these domains because they are the most common ones, and the highest chance of finding indirect references to CVEs. There were actually more CVEs that could be mapped indirectly than what we have in the final result. This is because some CVEs missed `node.js` as target software and since our dataset of CVEs only have CVEs with `node.js` as target software, we will only have indirectly mapped CVEs with `node.js` as target software in the final result. Table 4.6 shows which advisories and CVEs we have in the final result that were mapped indirectly to each other.

Table 4.5: Number of advisories mapped by common reference and indirectly.

Domain	Advisories	Common reference	Indirectly (`node.js` only)	Indirectly (all CVEs)
github.com	422	49 (11,6%)	0 (0%)	12 (2,8%)
gist.github.com	6	1 (16,7%)	1 (16,7%)	1 (16,7%)
hackerone.com	144	79 (54,9%)	4 (2,8%)	6 (4,1%)
snyk.io	71	15 (21,1%)	0 (0%)	29 (40,8%)
cve.mitre.org	6	0 (0%)	0 (0%)	4 (66,7%)

Table 4.6: CVEs and advisories that could be mapped indirectly through a reference.

CVE	Advisory	Indirect Reference
CVE-2019-11358	npm-796	`https://hackerone.com/reports/454365`
CVE-2018-3755	npm-671	`https://hackerone.com/reports/328210`
CVE-2018-3743	npm-669	`https://hackerone.com/reports/320693`
CVE-2018-3757	npm-670	`https://hackerone.com/reports/340208`
CVE-2019-10769	npm-1221	`gist.github.com/JLLeitschuh/609bb2efaff22ed84fe182cf574c023a`

There were 5 advisories that could be mapped to 5 CVEs with `node.js` as target software. If we would have included CVEs without `node.js` as target software, there would have been 42 extra advisories and CVEs that were mapped by common reference or indirectly.

4.1.4 Summary

Table 4.7 shows a summary of how many advisories and CVEs that were mapped, both by common reference and indirect reference. Since we only have CVEs with `node.js` as target software, we only mapped 5 advisories and CVEs indirectly. If we would have extended the dataset there would have been more advisories and CVEs mapped both by common reference and indirectly.

Table 4.7: Summary of the mapping.

Dataset	Vulnerabilities	Total mapped	Common reference	Indirect reference	Not mapped
Advisories	990	352 (35,6%)	347	5	638 (64,4%)
CVEs	691	349 (50,5%)	344	5	342 (49,5%)

The table shows that 35,6% of the scraped advisories could be mapped to 50,5% of the fetched CVEs. This means there are 638 advisories (64,4%) that does not exist on NVD, and there are 342 CVEs (49,5%) that does not exist on the NPM security advisory.

4.2 Comparison of NVD and the NPM Security Advisory

This section will make a comparison between CVEs from NVD and advisories from the NPM security advisory. There are some specific things we want to look at when we compare NVD and the NPM security advisory, which we decided are important for deciding if they should be used in combination with each other. These are the following:

- Things that always differ and differences in general between a mapped CVE and an advisory. There exists information that always differ between CVEs and advisories, CVEs having CWEs is one example. This is shown in Section 4.2.1.
- CVSS and severity. Can we find critical vulnerabilities on one of the sources that are not mapped to each other? Are there mapped vulnerabilities but with different CVSS and severity ratings? This is shown in Section 4.2.2.
- The published dates. Are there mapped vulnerabilities that are published earlier on one of the sources? Finding vulnerabilities as early as possible is important for developers. And are there any mapped vulnerabilities that were published earlier on one of the sources and have rating “Critical”? This is shown in Section 4.2.3.

4.2.1 General Differences

In this section we will show some examples of CVEs and advisories that differ. Some things that always differ between a mapped CVE and an advisory are the following:

- Advisories have a title which summarises what the vulnerability is about.
- The description slightly differs, sometimes there is more information on NVD and vice versa.
- Advisories might have information about when the vulnerability was reported (can be used as a proxy for the discovered date) but CVEs have no information about this.
- CVEs have CVSS scores and advisories have severity.

- CVEs have information about CVSS metrics, advisories have nothing similar to this.
- CVEs have information about which CWEs the CVE is categorised as.
- CVEs have affected software configurations and advisories simply tell which package is affected. For CVEs the package name is included in the software configuration string.
- Advisories show exactly which versions are affected and unaffected. CVEs only show a version range for the affected versions.
- Advisories have a status which tells if the vulnerability has been patched or not.
- Advisories suggests a remediation.
- Different number of references. It is common for advisories and CVEs to have different number of references.

Mapped on a Single Reference

Figure 4.2 and 4.3 shows CVE-2018-16472 [44] and advisory npm-739 [45], which differ in some ways. They were mapped on a common reference to HackerOne (`https://hackerone.com/reports/390847`).

```
CVE: CVE-2018-16472

Published date: 2018-11-06

Updated at: 2019-10-09

CVSS3 score: 7.5 (High)

Vector string: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

CWEs: CWE-20 (NIST), CWE-400 (HackerOne)

CWE categories: Improper Input Validation,
Uncontrolled Resource Consumption

Description: A prototype pollution
attack in cached-path-relative
versions <=1.0.1 allows an
attacker to inject properties on
Object.prototype which are then inherited by all
the JS objects through the prototype chain causing a
DoS attack.

Affected software configurations:
cpe:2.3:a:cached-path-relative_project:cached-path-relative:*:*:*:*:*:node.js:*:* (<= 1.0.1)

Third-party references:
https://hackerone.com/reports/390847
```

Figure 4.2: Example of a CVE which differ to its mapped advisory.

The advisory has “Prototype pollution” as the title, and also mentions it in the description. As mentioned before the CVE has no title but mentions prototype pollution in its description. The CVE also has a more detailed description than the advisory.

The CVE was published earlier (2018-11-06) than the advisory (2018-11-28). Also, the advisory has information about “Updated at” and “Created at”, which is

```
Advisory: npm-739

Title: Prototype Pollution

Package: cached-path-relative

Affected versions: 1.0.0, 1.0.1

Unaffected versions: 1.0.2

Published date: 2018-11-28

Created at: 2018-11-29

Updated at: 2018-11-29

Description: Version of cached-path-relative
before 1.0.2 are vulnerable to prototype pollution.

Remediation: Update to version 1.0.2 or later.

Severity: High

Status: Patched

Third-party references:
https://hackerone.com/reports/390847,
https://github.com/ashaffer/cached-path-relative/issues/3,
https://github.com/nodejs/security-wg/blob/master/vuln/NPM/480.json
```

Figure 4.3: Example of an advisory which differ to its mapped CVE.

later than the published date. This is because the published date is fetched from the "Date of advisory" column in `https://www.npmjs.com/advisories`. In this case that date is 2018-11-28. For this advisory the reported date is missing.

The CVE has a CVSS3 score of 7.5 ("High") and the advisory has severity "High". There are cases where these ratings are different, more about this in Section 4.2.2. And as mentioned before the CVE also has a vector string, which tells us which CVSS metrics the CVE has. The advisory does not have this. This CVE has the following metrics:

- Attack Vector (AV): Network (N)
- Attack Complexity (AC): Low (L)
- Privileges Required (PR): None (N)
- User Interaction (UI): None (N)
- Scope (S): Unchanged (U)
- Confidentiality (C): None (N)
- Integrity (I): None (N)
- Availability (A): High (H)

The CVE belongs to two CWEs (`CWE-20 (NIST)` and `CWE-400 (HackerOne)`). The CVE was categorised as `CWE-20 (Improper Input Validation)` by NIST and `CWE-400 (Uncontrolled Resource Consumption)` by HackerOne. As mentioned before, the advisory has no CWEs.

The CVE's affected software configuration tells us that the dependency with vendor and product `cached-path-relative` with the target software `node.js` is

affected by CVE-2018-16472 up until and including version `1.0.1`. The advisory simply have the package name (`cached-path-relative`), but tells us exactly which versions that are affected (`1.0.0, 1.0.1`) and unaffected (`1.0.2`). The advisory informs about the status of the vulnerability and a suggested remediation. In this case the vulnerability has been patched and the suggested remediation is to update to a version higher than `1.0.2`. The CVE does not have this type of information.

All CVEs and advisories were mapped on its common references, but either the CVE or the advisory often has extra references that we could not map on. In the example above the advisory has a reference to a GitHub issue and to a vulnerability in the NPM vulnerability repository. The CVE and the advisory was mapped on a common reference to HackerOne and the only reference the CVE has is that one. This indicates that there can be extra information found if NVD is used in combination with the NPM security advisory.

Mapped on Multiple References

Figure 4.4 and 4.5 shows CVE-2018-1002204 [46] and advisory npm-994 [47] which were mapped to each other on several references. They were mapped on a reference to Snyk and a GitHub pull request.

```
CVE: CVE-2018-1002204

Published date: 2018-07-25

CVSS3 score: 5.5 (Medium)

Vector string: CVSS:3.1/AV:L/AC:L/PR:N/UI:R/S:U/C:N/I:H/A:N

CWEs: CWE-22 (NIST, Snyk)

CWE categories: Improper Limitation of a Pathname
to a Restricted Directory ('Path Traversal')

Description: adm-zip NPM library before 0.4.9 is vulnerable to
directory traversal, allowing attackers to write to arbitrary files
via a ../ (dot dot slash) in a Zip archive entry
that is mishandled during extraction. This vulnerability is
also known as 'Zip-Slip'.

Affected software configurations:
cpe:2.3:a:adm-zip_project:adm-zip:*:*:*:*:*:node.js:*:* (< 0.4.9)

Third-party references:
github.com/cthackers/adm-zip/
commit/62f64004fefb894c523a7143e8a88ebe6c84df25,
github.com/cthackers/adm-zip/pull/212,
github.com/snyk/zip-slip-vulnerability,
snyk.io/research/zip-slip-vulnerability,
snyk.io/vuln/NPM:adm-zip:20180415,
securityfocus.com/bid/107001
```

Figure 4.4: Example of a CVE which has been mapped to an advisory on the NPM security advisory by several common references.

The advisory has the title "Arbitrary File Write" and arbitrary file write is mentioned in both the CVE and advisory description.

The CVE was published earlier (2018-07-25) than the advisory (2019-07-11), but the advisory has a "Reported at" date (2018-08-11), which is 17 days after the CVE published date. The reported date in this case is therefore not a good proxy

```
Advisory: npm-994

Title: Arbitrary File Write

Package: adm-zip

Affected versions: 0.1.1, 0.1.2, 0.1.3, 0.1.4,
0.1.5, 0.1.6, 0.1.7, 0.1.8, 0.1.9, 0.2.0,
0.2.1, 0.4.3, 0.4.4, 0.4.5, 0.4.6,
0.4.7, 0.4.8

Unaffected versions: 0.4.9, 0.4.10, 0.4.11,
0.4.13, 0.4.14, 0.4.16

Published date: 2019-07-11

Reported at: 2018-08-11

Description: Versions of adm-zip before 0.4.9
are vulnerable to arbitrary file write when
used to extract a specifically crafted archive
that contains path traversal
filenames (../../file.txt for example).

Remediation: Update to version 0.4.9 or later.

Severity: High

Status: Patched

Third-party references:
github.com/cthackers/adm-zip/pull/212,
snyk.io/research/zip-slip-vulnerability,
hackerone.com/reports/362118
```

Figure 4.5: Example of an advisory which has been mapped to a CVE by several common references.

for when the vulnerability was discovered.

The CVE has a CVSS3 score of 5.5 (“Medium”) and the advisory has severity “High”. This is an example of where a mapped CVE and advisory have different ratings. As with the previously compared CVE and advisory the CVE has a vector string, which tells us what CVSS metrics the CVE has. This CVE has the following metrics:

- Attack Vector (AV): Low (L)
- Attack Complexity (AC): Low (L)
- Privileges Required (PR): None (N)
- User Interaction (UI): Required (R)
- Scope (S): Unchanged (U)
- Confidentiality (C): None (N)
- Integrity (I): High (H)
- Availability (A): None (N)

The CVE has one CWE, `CWE-22` (`Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')`), which both NIST and Snyk assigned to the CVE. The advisory has no CWEs.

The CVE’s affected software configuration tells us that the dependency with vendor `adm-zip_project` and product `adm-zip` with the target software `node.js`

is affected by CVE-2018-1002204 up until but not including version `0.4.9`. The advisory tells us which package that is affected (`adm-zip`) and which versions that are affected and unaffected. The advisory informs us that the vulnerability has been patched and the suggested remediation is to update to version 0.4.9 or later.

In this example it is the CVE that has more references than the advisory. The CVE has extra references to a repository owned by Snyk which informs about the vulnerability, a GitHub commit which fixes the vulnerability, a Snyk reference to the vulnerability and a SecurityFocus reference to the vulnerability. SecurityFocus also provides a vulnerability database as NVD and NPM does [48]. The NPM advisory has fewer references but has one reference which the CVE does not have, a HackerOne reference.

4.2.2 CVSS and Severity

Figure 4.6 show the distribution of CVSS and severity rating in the venn diagram. The result show that most of the advisories and CVEs have rating "High", 57,3% of all advisories and 60,8% of all CVEs. For the mapped advisories and CVEs 68,2% of the advisories and 60,5% of the CVEs have rating "High". The result also show that there is a higher proportion of CVEs with rating "Critical", 17,8% of all CVEs and 9,1% of all advisories. For the mapped CVEs 16,3% have rating "Critical" and for the mapped advisories 6,8% have rating "Critical".

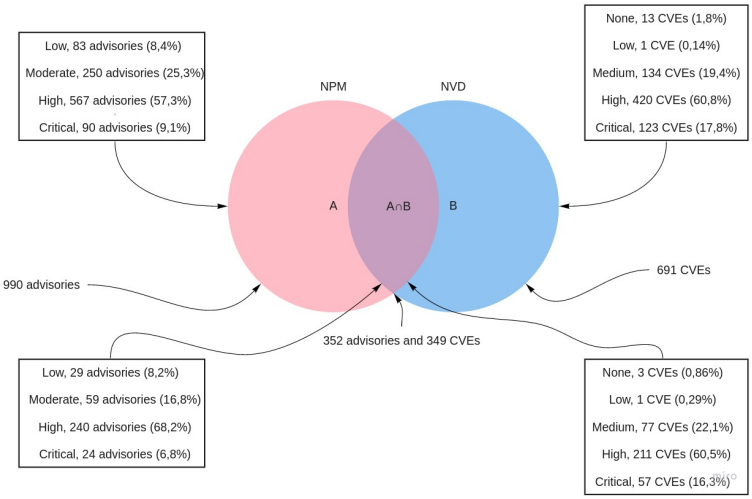

Figure 4.6: Venn diagram with CVSS distribution.

Figure 4.7 shows the distribution of CVSS ratings and severity for the vulnerabilities that could not be mapped. There are 66 critical advisories (6,7% of all advisories) that could not be mapped to a CVE, which is quite worrisome since NVD is often used as a main source for vulnerabilities. There is a lot of critical vulnerabilities that might not be found because of this. On NVD there was 66 CVEs (9,6% of all fetched CVEs) that were critical and could not be mapped to an advisory.

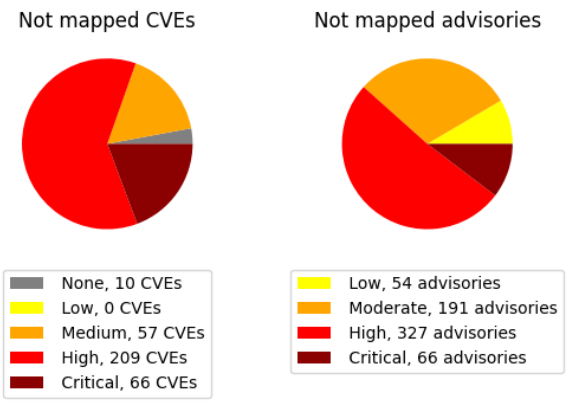

Figure 4.7: Graph over distribution of CVSS rating and severity for all CVEs and advisories that were not mapped to each other by common reference.

Since there are vulnerabilities on both NVD and NPM with critical severity that could not be mapped to each other this is an indication to use both sources.

Critical Vulnerability Existing on NPM only

One example of a critical vulnerability that exists on NPM only is npm-1487 [49], see Figure 4.8. The advisory was published 2020-02-28 and still has the status "Not patched" (2020-09-16). The vulnerability affects all versions of the package `react-oauth-flow` and the remediation is therefore to use an alternative module. As the description says, the package `react-oauth-flow` fail to properly implement the OAuth protocol and stores secrets in front-end code. The third-party reference leads to a GitHub issue in the `ethereum/web3.js`, which is a JavaScript API for the cryptocurrency platform Ethereum [50]. In the issue they discuss a solution for avoiding saving the private key used for loading the crypto wallet in the local storage in the front-end.

Critical Vulnerability Existing on NVD only

Figure 4.9 shows an example of a critical vulnerability that exists on NVD only, CVE-2020-8147 [51]. The vulnerability was published on 2020-04-03 and as the description says the NPM package `utils-extend_project/utils-extend` may allow prototype pollution attack which can result in remote code execution or denial of service. If we look in the NPM registry we can see which versions that exist for this package, and there it is found that the latest version is `1.0.8` [52].

This means that all versions of `utils-extend_project/utils-extend` is affected by the vulnerability. It is hard to know that the vulnerability has not been patched when NVD does not inform about the status of the vulnerability or a suggested remediation, as the NPM security advisory does.

```
Advisory: npm-1487

Title: Improper Authorization

Package: react-oauth-flow

Affected versions: 1.0.0, 1.0.1, 1.0.2, 1.1.0, 1.1.1, 1.1.2, 1.2.0

Unaffected versions: None

Published date: 2020-02-28

Reported at: 2020-02-28

Description: All versions of react-oauth-flow fail to properly implement
the OAuth protocol. The package stores secrets in the front-end code.
Instead of using a public OAuth client, it uses a confidential client on the browser.
This may allow attackers to compromise server credentials.

Remediation: No fix is currently available. Consider using an alternative
module until a fix is made available.

Severity: Critical

Status: Not patched

Third-party references:
https://github.com/ethereum/web3.js/issues/2739,
```

Figure 4.8: Example of an advisory with critical severity which has not been mapped to a CVE.

```
CVE: CVE-2020-8147

Package: utils-extend_project/utils-extend

Published date: 2020-04-03

CVSS3 score: 9.8 (Critical)

Vector string: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CWEs: CWE-20 (NIST), CWE-471 (HackerOne)

CWE categories: Improper Limitation of a Pathname
to a Restricted Directory ('Path Traversal')

Description: Flaw in input validation in NPM package utils-extend
version 1.0.8 and earlier may allow prototype pollution attack
that may result in remote code execution or denial of
service of applications using utils-extend.

Affected software configurations:
cpe:2.3:a:utils-extend_project:utils-extend:*:*:*:*:*:node.js:*:* (<= 1.0.8)

Third-party references:
https://hackerone.com/reports/801522
```

Figure 4.9: Example of a CVE with critical severity which has not been mapped to an advisory.

Different Ratings between Mapped Vulnerabilities

Since we could not find information about what the severity of advisories is based on (if it is the same as CVSS) we can not be completely sure that a critical vulnerability on NVD should be critical on NPM, it might be so that on NPM the rating will be "High". But it seems reasonable that for a CVE with rating "Critical", the corresponding advisory should have at least rating "High". Therefore we will try to find mapped vulnerabilities where the CVE has rating "Critical" and the advisory has a lower rating than "High", and where the advisory has rating "Critical" and the CVE has a lower rating than "High".

There are 43 mapped CVEs (12,3% of the mapped CVEs) and advisories (12,2% of the mapped advisories) where the CVE has rating "Critical" and the advisory has a lower rating. Out of these there are 20 (5,7% of the mapped advisories) advisories that have a lower rating than "High" when the CVE has rating "Critical". See Table 4.8.

Table 4.8: Advisories that have a lower rating than "High" when the mapped CVE has rating "Critical".

CVE	Rating	Advisory	Rating
CVE-2018-16492	Critical	npm-996	Moderate
CVE-2018-3767	Critical	npm-970	Moderate
CVE-2019-10061	Critical	npm-789	Low
CVE-2018-16491	Critical	npm-781	Moderate
CVE-2018-16489	Critical	npm-780	Moderate
CVE-2018-16486	Critical	npm-778	Moderate
CVE-2019-5413	Critical	npm-736	Moderate
CVE-2018-16460	Critical	npm-728	Moderate
CVE-2018-16461	Critical	npm-719	Moderate
CVE-2018-3752	Critical	npm-717	Low
CVE-2018-3753	Critical	npm-716	Low
CVE-2018-3751	Critical	npm-715	Low
CVE-2018-3786	Critical	npm-694	Low
CVE-2018-3745	Critical	npm-646	Moderate
CVE-2018-3750	Critical	npm-612	Low
CVE-2018-3749	Critical	npm-611	Low
CVE-2017-16127	Critical	npm-482	Moderate
CVE-2016-10554	Critical	npm-113	Moderate
CVE-2015-9244	Critical	npm-66	Moderate
CVE-2015-8857	Critical	npm-39	Low

For example the CVE-2019-10061 in the table above has rating "Critical" [53] but the mapped advisory has rating "Low" [54]. The CVE has a direct reference to `https://www.npmjs.com/advisories/789` and that is the reference they were mapped on. The vulnerability is about command injection where the `opencv` package does not validate user input and that allows attackers to execute arbitrary commands.

There are 12 mapped CVEs (3,43% of the mapped CVEs) and advisories (3,4%

of the mapped advisories) where the advisory has rating "Critical" and the CVE has a lower rating. Out of these there are 8 CVEs (2,3% of the mapped CVEs) that have a lower rating than "High" when the advisory has rating "Critical". See Table 4.9.

Table 4.9: CVEs that have a lower rating than "High" when the mapped advisory has rating "Critical".

Advisory	Rating	CVE	Rating
npm-523	Critical	CVE-2014-9682	None
npm-26	Critical	CVE-2014-10067	Medium
npm-144	Critical	CVE-2016-10548	Medium
npm-17	Critical	CVE-2016-10555	Medium
npm-87	Critical	CVE-2016-10555	Medium
npm-570	Critical	CVE-2018-3726	Medium
npm-741	Critical	CVE-2018-16474	Medium
npm-1143	Critical	CVE-2019-5480	Medium

For example advisory `npm-1143` in the table above has rating "Critical" [55] but the mapped CVE (`CVE-2019-5480`) has rating "Medium" [56]. They were mapped on a common reference to HackerOne (`https://hackerone.com/reports/570035`). The advisory's description says that the dependency `statichttpserver` is vulnerable to Path Traversal, because it fails to sanitize URLs which allows attackers to access server files outside of the served folder using relative paths. The CVE's description simply says that it is possible to list files in arbitrary folders. All versions of the package is affected, and on NVD it says all versions up until and including version (`0.9.7`) is affected. Again, it is unclear on NVD if all versions are affected and one have to check themselves on the NPM registry which versions are available for `statichttpserver`.

4.2.3 Published dates

Earlier Published Date on NPM

There are 286 mapped advisories (81,3% of the mapped advisories) that have an earlier published date than its mapped CVE. This is a strong indication of using the NPM security advisory together with NVD.

One example of a critical advisory that was published much earlier than the CVE (over 3 years difference) is advisory npm-27 [57], which was mapped to CVE-2014-3741 [58] on a common reference to a GitHub commit. See Figure 4.10 and 4.11.

The CVE has published date 2017-10-23 and the advisory has 2014-03-06. Since this is a critical vulnerability on both sources it would have been important to publish it earlier on NVD in order for developers to discover it as early as possible. They were mapped on a common reference to a GitHub commit (`https://github.com/tojocky/node-printer/commit/e001e38738c17219a1d9dd8c31f7d82b9c0013c7`) and the vulnerability is about command injection in

```
CVE: CVE-2014-3741

Published date: 2017-10-23

CVSS3 score: 9.8 (Critical)

Vector string: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CWEs: CWE-77 (NIST)

CWE categories: Improper Neutralization of
Special Elements used in a Command ('Command Injection')

Description: The printDirect function in lib/printer.js
in the node-printer module 0.0.1 and earlier for Node.js
allows remote attackers to execute arbitrary
commands via unspecified characters in the lpr command.

Affected software configurations:
cpe:2.3:a:node-printer_project:node-printer:*:*:*:*:*:node.js:*:* (<= 0.0.1)

Third-party references:
openwall.com/lists/oss-security/2014/05/13/1
openwall.com/lists/oss-security/2014/05/15/2
github.com/tojocky/node-printer/commit/e001e38738c17219a1d9dd8c31f7d82b9c0013c7
npmjs.com/advisories/printer_potential_command_injection
```

Figure 4.10: Example of a CVE which has been mapped to an advisory on the NPM security advisory by common reference, where the advisory has an earlier published date.

```
Advisory: npm-27

Title: Potential Command Injection

Package: printer

Affected versions: 0.0.1

Unaffected versions: 0.0.2, 0.0.3, 0.0.4, 0.0.5, 0.0.6,
0.1.0, 0.1.1, 0.2.0, 0.2.1, 0.2.2, 0.4.0

Published date: 2014-03-05

Reported at: 2015-10-16

Description: Versions 0.0.1 and earlier of printer are
affected by a command injection vulnerability resulting
from a failure to sanitize command arguments properly in
the printDirect() function.

Remediation: Update to version 0.0.2 or later.

Severity: Critical

Status: Patched

Third-party references:
https://github.com/tojocky/node-printer/commit/e001e38738c17219a1d9dd8c31f7d82b9c0013c7
```

Figure 4.11: Example of an advisory which has been mapped to a CVE on NVD by common reference, where the advisory has an earlier published date.

the `printer` dependency, affecting version `0.0.1`. The dependency fails to sanitize command arguments properly.

Earlier Published Date on NVD

There are 60 mapped CVEs (17,2% of the mapped CVEs) that have an earlier published date than its mapped advisory.

One example of a CVE that was published earlier than the advisory and the advisory has rating "Critical" is CVE-2014-9682 [59] which was mapped to npm-523 [60] on a GitHub issue and a GitHub commit. See Figure 4.12 and 4.13. The CVE was published (2015-02-27) 2 years earlier than the advisory (2017-09-07) and this was the biggest difference in published dates we could find where the CVE was published earlier than the advisory. This means that advisories in general are published earlier than CVEs since there were more advisories that was published earlier and the biggest date difference we could find when an advisory was published earlier than the CVE was 6 years. This was for CVE-2013-4691 [61] and advisory npm-3 [62]. The CVE was published on 2019-12-27 and the advisory on 2013-06-30.

```
CVE: CVE-2014-9682

Published date: 2015-02-27

CVSS3 score: None

Vector string: None

CWEs: CWE-77 (NIST)

CWE categories: Improper Neutralization of
Special Elements used in a Command ('Command Injection')

Description: The dns-sync module before 0.1.1 for
node.js allows context-dependent attackers to execute
arbitrary commands via shell metacharacters in the first
argument to the resolve API function.

Affected software configurations:
cpe:2.3:a:dns-sync_project:dns-sync:*:*:*:*:*:node.js:*:* (<= 0.1.0)

Third-party references:
openwall.com/lists/oss-security/2014/11/11/6
github.com/skoranga/node-dns-sync/commit/d9abaae384b198db1095735ad9c1c73d7b890a0d
github.com/skoranga/node-dns-sync/issues/1
```

Figure 4.12: Example of a CVE which has been mapped to an advisory on the NPM security advisory by common reference, where the CVE has an earlier published date.

The CVE in Figure 4.12 and advisory in Figure 4.13 is also about command injection in the `dns-sync` dependency, affecting version `0.1.0`. The advisory has rating "Critical" and the CVE has rating "None". Older CVEs sometimes miss CVSS3 scores. Although there are unaffected versions the advisory says that the remediation should be to use an alternative dns resolver.

```
Advisory: npm-523

Title: Command Injection

Package: dns-sync

Affected versions: 0.1.0

Unaffected versions: 0.1.1, 0.1.2, 0.1.3, 0.2.0, 0.2.1

Published date: 2017-09-07

Reported at: 2017-09-05

Description: Affected versions of dns-sync have an
arbitrary command execution vulnerability in the
resolve() method.

Remediation:
Use an alternative dns resolver
Do not allow untrusted input into dns-sync.resolve()

Severity: Critical

Status: Patched

Third-party references:
github.com/skoranga/node-dns-sync/issues/1
github.com/skoranga/node-dns-sync/commit/d9abaae384b198db1095735ad9c1c73d7b890a0d
```

Figure 4.13: Example of an advisory which has been mapped to a CVE on NVD by common reference, where the CVE has an earlier published date.

Chapter 5
Discussion

The results show that it is probably best to use NVD in combination with the NPM security advisory in order to get as much information about vulnerabilities as possible, but also in order to find vulnerabilities that does not exist on one of the sources. The mapping of the vulnerabilities between NPM and NVD showed that there are many vulnerabilities, 638 advisories (64,4% of all scraped advisories) and 342 CVEs (49,5% of all fetched CVEs), that only exists on one of the sources. This was after doing the mapping on common references and on indirect references from `github.com`, `hackerone.com`, `snyk.io` and `cve.mitre.org`.

It was also found that there are many critical vulnerabilities that only exist on one of the sources, which even further motivates to use both NVD and the NPM security advisory. 6,7% of all advisories were critical and did only exist on the NPM security advisory and 9,5% of all CVEs were critical and did only exist on NVD. In general there were more critical vulnerabilities on NVD than on NPM, which is good since NVD is often used as the main source for vulnerabilities, 17,8% on NVD vs 9,1% on NPM. But since we can not be completely sure that NPM's severity is based on the same metrics as CVSS, it might be so that some critical vulnerabilities on NVD have a lower rating on NPM.

For the vulnerabilities that were mapped to each other, we found there were many differences between them as well. The CVSS rating and severity rating might differ between CVEs and advisories, and this can make developers think that a vulnerability that has a lower rating on one the sources is not very critical if they only use one source.

There might be more references on one of the sources, which lets us find more information about a vulnerability if the two sources are used in combination. The descriptions almost always differ and one of the sources might give a more thorough description of the vulnerability. There is also information that always only exist on one of the sources. NVD has information about which CWE categories the vulnerability belongs to and NPM has information about the status of the vulnerability and the remediation for it. The CWE categories gives developers more information about what type of vulnerability it is, which NPM does not have. NPM having information about status and remediation makes it much easier for developers to know what action to take if they are affected by a vulnerability. NVD have the affected software configurations which tells us what version ranges are affected, but it is not clear if the vulnerability has been patched or not or what remediation that should be done.

The NPM security advisory has information about when the vulnerability was reported to NPM, which sometimes gives a good proxy for when the vulnerability was discovered. There are cases of when the reported date is missing or when it is about the same as the published date. In those cases the reported date might not be a good proxy for the discovered date. It was also found that the vulnerabilities might be published earlier on one of the sources. Most of the mapped vulnerabilities were published on the NPM security advisory first (81,3% of the mapped advisories). This allows developers to discover vulnerabilities earlier in its life cycle, and makes it possible to fix them as soon as possible.

Chapter 6
Conclusion and Future Research

In this thesis we have scraped vulnerabilities from the NPM security advisory and fetched CVEs affecting `node.js` from NVD using Debricked's vulnerability database. The advisories and CVEs were mapped to each other on their common references and indirect references. We scraped 990 advisories from the NPM security advisory which had a published date before 2020-05-18 and fetched 691 CVEs from Debricked with a published date before 2020-05-18. We managed to map 352 advisories (35,6 % of all scraped advisories) to 349 CVEs (50,5% of all fetched CVEs). This means that 64,4% of the advisories and 49,5% of the CVEs could not be mapped.

A comparison was then made between NPM and NVD as vulnerability sources. Some general differences between mapped advisories and CVEs was found. For example that CVEs have information about which CWEs the vulnerability belongs to but NPM has more clear information about the status of a vulnerability and the recommended remediation. It was also found that the mapped vulnerabilities might have different number of references which lets developers find more information about a vulnerability if the two vulnerability sources are used in combination with each other.

The CVSS ratings for CVEs and severity ratings for vulnerabilities can also differ. But, it is not certain that CVSS and NPM's severity is based on exactly the same metrics and therefore the ratings are not always the same. But we found that there was 5,7% of the mapped advisories with a lower rating than "High" when the corresponding CVE has rating "Critical". We also found that 2,3% of the mapped CVEs have a lower rating than "High" when the corresponding advisory has rating "Critical". In general there were more critical vulnerabilities on NVD than on NPM, 17,8% of the CVEs and 9,1% of the advisories were critical.

It was also found that the published dates differ between mapped advisories and CVEs, 81,3% of the mapped advisories were published earlier than the CVEs. The biggest date difference found between an advisory and a CVE was 6 years. This was for advisory npm-3 and CVE-2013-4691. The advisory was published on 2013-06-30 and the CVE on 2019-12-27. The biggest date difference found when a CVE was published earlier than an advisory was 2 years. This was for CVE-2014-9682 and npm-523.

To conclude this thesis the verdict will be that its best to use NVD in combination with NPM in order to get as much information as possible about vulnerabilities and to find vulnerabilities that only exist on one of the sources.

There could have been more mapped vulnerabilities if we did not restrict ourselves to CVEs with `node.js` as target software. But the result is still good enough in order to do a comparison of the two vulnerability sources. In the future one can try to map all existing CVEs on NVD to the advisories on the NPM security advisory. There also exists other similar sources as the NPM security advisory but for other package managers/programming languages. The PHP Security Advisories Database on GitHub is one example [63] and the Ruby Advisory Database [64] is another example. The same work done in this thesis could be done for those sources. One could also try to fetch vulnerabilities from all these different sources and map them to each other. And then create a combined vulnerability database with vulnerabilities that contain the available information from all these different sources.

We could have done the indirect mapping on all references that were not mapped on common reference, but was only done for the 4 mentioned domains above (`github.com`, `hackerone.com`, `snyk.io` and `cve.mitre.org`) since they were the most common. Also, the indirect mapping was done manually by simply searching for "CVE" in the third-party reference. This could have been done automatically by using text analysis/natural language processing in order to decide if the reference contains information about a CVE or not. This method can also be used in order to create the combined vulnerability database mentioned above. For example one can do natural language processing on GitHub issues and create vulnerabilities from the result.

Bibliography

[1] OWASP, *OWASP Top 10 web application security risks*, https://www.owasp.org/index.php/Top_10-2017_A9-Using_Components_with_Known_Vulnerabilities

[2] Synopsys Cybersecurity Research Center, *60 percent of enterprise codebases contain open-source vulnerabilities*, https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/rep-ossra-19.pdf

[3] Wikipedia, *Heartbleed*, https://en.wikipedia.org/wiki/Heartbleed

[4] SSL Labs, *SSL Pulse*, https://www.ssllabs.com/ssl-pulse/

[5] National Institute of Standards and Technology, *National Vulnerability Database*, https://nvd.nist.gov

[6] NPM, *NPM security advisory*, https://www.npmjs.com/advisories

[7] Alexandre Decan, Tom Mens and Eleni Constantinou. *On the Impact of Security Vulnerabilities in the NPM Package Dependency Network*. In 2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR), Gothenburg, Sweden, 2018

[8] Steven M. Muegge, S. M. Monzur Murshed. *Time to Discover and Fix Software Vulnerabilities in Open Source Software Projects: Notes on Measurement and Data Availability*. In 2018 Portland International Conference on Management of Engineering and Technology (PICMET), Honolulu, HI, USA, 2018

[9] Wikipedia, *Software vulnerability*, https://en.wikipedia.org/wiki/Vulnerability_(computing)

[10] Wikipedia, *Package manager*, https://en.wikipedia.org/wiki/Package_manager

[11] Wikipedia, *List of software package management systems*, https://en.wikipedia.org/wiki/List_of_software_package_management_systems#Application-level_package_managers

[12] Wikipedia, *npm (software)*, https://en.wikipedia.org/wiki/Npm_(software)

[13] NPM, *About the public npm registry*, https://docs.npmjs.com/about-the-public-npm-registry)

[14] Mitre, *Common Vulnerabilities and Exposures*, https://cve.mitre.org/

[15] Mitre, *CVE Numbering Authorities*, https://cve.mitre.org/cve/cna.html

[16] Wikipedia, *National Institute of Standards and Technology*, https://en.wikipedia.org/wiki/National_Institute_of_Standards_and_Technology

[17] National Vulnerability Database, CVE-2020-8125, https://nvd.nist.gov/vuln/detail/CVE-2020-8125

[18] NPM, npm-1463, https://www.npmjs.com/advisories/1463

[19] NPM, npm-1463, https://www.npmjs.com/advisories/1463/versions

[20] Wikipedia, *Web scraping*, https://en.wikipedia.org/wiki/Web_scraping

[21] Debricked, *Debricked vulnerability sources*, https://debricked.com/vulnerability-data/

[22] Mitre, *Common Weakness Enumeration*, https://cwe.mitre.org/

[23] Mitre, *Weaknesses in the 2019 CWE Top 25 Most Dangerous Software Errors*, https://cwe.mitre.org/data/definitions/1200.html

[24] Mitre, CWE-119, https://cwe.mitre.org/data/definitions/119.html

[25] NIST, *Common Platform Enumeration*, https://csrc.nist.gov/projects/security-content-automation-protocol/specifications/cpe/

[26] NIST, *Common Platform Enumeration* https://csrc.nist.gov/publications/detail/nistir/7695/final

[27] NIST, *Known Affected Software Configurations* https://nvd.nist.gov/vuln/Vulnerability-Detail-Pages

[28] FIRST, *Common Vulnerability Scoring System*, https://www.first.org/cvss/specification-document

[29] NVD, *CVSS score calculator*, https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator

[30] NPM, *About audit reports (severity)*, https://docs.npmjs.com/about-audit-reports#severity

[31] NPM, *Number of NPM packages*, https://www.npmjs.com/

[32] Wikipedia, *Typosquatting*, https://en.wikipedia.org/wiki/Typosquatting

[33] Medium, *NPM Acquires Ĺift Security and the Node Security Platform*, https://medium.com/npm-inc/npm-acquires-lift-security-258e257ef639

[34] Wikipedia, *HackerOne*, https://en.wikipedia.org/wiki/HackerOne

[35] GitHub, *GitHub security advisory*, https://help.github.com/en/github/managing-security-vulnerabilities/about-github-security-advisories#about-github-security-advisories

[36] Snyk, *Snyk Vulnerability DB*, https://snyk.io/vuln

[37] NVD, CVE-2018-3717, https://nvd.nist.gov/vuln/detail/CVE-2018-3717

[38] NPM, *Reporting a vulnerability in a NPM package*, https://docs.npmjs.com/reporting-a-vulnerability-in-an-npm-package

[39] NPM, npm-584, https://www.npmjs.com/advisories/584

[40] NPM, npm-595, https://www.npmjs.com/advisories/595

[41] NVD, CVE-2020-7636, https://nvd.nist.gov/vuln/detail/CVE-2020-7636

[42] NVD, CVE-2015-1370, https://nvd.nist.gov/vuln/detail/CVE-2015-1370

[43] GitHub, *Ecosystem Security Working Group*, https://github.com/nodejs/security-wg

[44] NVD, CVE-2018-16472, https://nvd.nist.gov/vuln/detail/CVE-2018-16472

[45] NPM, npm-739, https://www.npmjs.com/advisories/739

[46] NVD, CVE-2018-1002204, https://nvd.nist.gov/vuln/detail/CVE-2018-1002204

[47] NPM, npm-994, https://www.npmjs.com/advisories/994

[48] SecurityFocus, *About Security Focus*, https://www.securityfocus.com/about

[49] NPM, npm-1487, https://www.npmjs.com/advisories/1487

[50] GitHub, web3.js - Ethereum JavaScript API, https://github.com/ethereum/web3.js

[51] NVD, CVE-2020-8147, https://nvd.nist.gov/vuln/detail/CVE-2020-8147

[52] NPM, utils-extend, https://www.npmjs.com/package/utils-extend

[53] NIST, CVE-2019-10061, https://nvd.nist.gov/vuln/detail/CVE-2019-10061

[54] NPM, npm-789, https://www.npmjs.com/advisories/789

[55] NPM, npm-1143, https://www.npmjs.com/advisories/1143

[56] NIST, CVE-2019-5480, https://nvd.nist.gov/vuln/detail/CVE-2019-5480

[57] NPM, npm-27, https://www.npmjs.com/advisories/27

[58] NVD, CVE-2014-3741, https://nvd.nist.gov/vuln/detail/CVE-2014-3741

[59] NVD, CVE-2014-9682, https://nvd.nist.gov/vuln/detail/CVE-2014-9682

[60] NPM, npm-523, https://www.npmjs.com/advisories/523

[61] NVD, CVE-2013-4691, https://nvd.nist.gov/vuln/detail/CVE-2013-4691

[62] NPM, npm-3, https://www.npmjs.com/advisories/3

[63] GitHub, *PHP Security Advisories Database*, https://github.com/FriendsOfPHP/security-advisories

[64] GitHub, *Ruby Advisory Database*, https://github.com/rubysec/ruby-advisory-db

Printed by Tryckeriet i E-huset, Lund 2020

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2020-792
http://www.eit.lth.se