COMBATING SOURCE POISONING AND NEXT-GENERATION SOFTWARE SUPPLY CHAIN ATTACKS USING EDUCATION, TOOLS, AND TECHNIQUES

By

THOMAS G. HASTINGS

B.S., Colorado Christian University, USA, 2013

M.E., University of Colorado Colorado Springs, USA, 2018

A dissertation submitted to the Graduate Faculty of the

University of Colorado Colorado Springs

in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2024

This dissertation for Doctor of Philosophy degree by

Thomas G. Hastings

has been approved for the

Department of Computer Science

by

Kristen Walcott, Chair

C. Edward Chow

Richard White

Joshua Alcorn

Joel Coffman

April 18, 2024

Date

Hastings, Thomas G. (Ph.D., Engineering)

Combating Source Poisoning and Next-Generation Software Supply Chain Attacks Using Education, Tools, and Techniques

Dissertation directed by Associate Professor Kristen Walcott

ABSTRACT

We are heading for a perfect storm, making open-source software poisoning and next-generation supply chain attacks easier to execute, which could have significant implications for organizations. The widespread adoption of open source, the ease of today's package managers, and the best practice of implementing continuous delivery for software projects provide an unprecedented opportunity for attack. It used to be patch on a Friday to prevent a breach on Monday. Now, it is patch on Friday and a breach on Monday. Open-source projects are being targeted at an alarming rate by malicious maintainers and actors. These exploits flow to downstream projects that ingest the compromised patch, and those projects are potentially running the malicious code. Software developers must stay aware of such risks, and organizations must equip themselves to understand or monitor the thousands or tens of thousands of dependencies in their software.

This thesis develops and evaluates techniques organizations can use to protect their software from next-generation supply chain attacks. The first set of techniques evaluates organizational procedures for a framework for continuously evaluating open-source components throughout the software life-cycle. Our evaluation takes a holistic approach

to evaluating open-source components starting at Day 0 with the decision to incorporate a component into a software project through Day 2 operations, which occurs after the software has been deployed to production and is in maintenance. We start with evaluating the known knowns of an open-source component beginning with the component's community and code base. Then, we move on to the known unknowns and the unknown unknowns and place techniques and procedures in place to mitigate the risks of the unknowns. Defined controls combine the techniques and procedures into a flexible and repeatable framework that any organization leveraging code-reuse and open-source components in their software development can apply.

The second set of techniques is foundational to an organization's workforce, focusing on education for software engineers. The curriculum gives students the background and knowledge they need to protect the organization. First, students learn to identify risks associated with open-source components. Second, students learn how malicious actors target and execute attacks on open-source components. Third, students learn tactics to mitigate the risks of next-generation supply chain attacks using our open-source platform that provides a standardized development environment for each student running on cloud infrastructure, providing an equitable experience. Our curriculum had a 84% efficiency rate when taught to undergraduate and graduate students at the University of Colorado Colorado Springs. Additionally, we developed a cloud-based platform that removes barriers to our curriculum and provides an inclusive platform, giving students access to computing resources independently of their local hardware, which they access through a web browser.

ACKNOWLEDGMENTS

I want to thank my dissertation committee for providing thoughtful feedback throughout this process and acknowledge the State of Colorado and UCCS for awarding me grant money to continue pursuing my education, totaling $13,000.

Dr. Kristen Walcott, thank you for accepting me as your graduate student, first as a master's student and then as a Ph.D. Your mentorship has meant a lot to me as I navigated the academic world, from publishing papers to presenting at conferences and teaching; you stood by me and helped me grow. Thank you for the numerous opportunities you gave me to excel at UCCS and for helping me find my passion for teaching and working with students.

Lastly, thank you to my employers and managers for showing me grace when I needed the mental bandwidth to work on my coursework and research. Scott Cowher at Parsons Corp, Cullen Frye at Amazon Web Services, and Dr. Earl Brewster at the United States Air Force Academy, thank you!

DEDICATION

To my wonderful wife, Kati, and our daughters, Adilyn, Harper, Laurel, Dorothy, and Eloise, for putting up with me working late at night and weekends and being irritable at times as I worked through my classes and this dissertation. Kati, thank you for encouraging me and not letting me quit as much as I wanted to. This dissertation was only possible because of you. I dedicate this work to my Lord and Savior, Jesus Christ.

TABLE OF CONTENTS

CHAPTER

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURE

CHAPTER I

INTRODUCTION

We are heading towards a perfect storm for insecure and malicious software to enter production software stacks. The rise of open-source component utilization, the lack of project vetting techniques, and the overwhelming sense to deliver value faster have left us vulnerable to attack. Open-source software is utilized in 99% of software applications today [2]. Unfortunately, many software engineers rely on limited defensive techniques when vetting software projects, such as looking at recent activity within a project before incorporating the project into their software baseline [3]. This method offers very little protection for software projects.

Currently, there is no good way to manage the weak links of open-source packages [4]. Best practices are unwittingly putting organizations at risk. One reason is that open-source components allow developers to incorporate new features seamlessly and effortlessly with only minor modifications. This code reuse enables software engineers to deliver value faster to their customer base. This practice is encouraged within organizations and is referred to by the National Institute of Science and Technology

(NIST) in their publication for Secure Software Development [5] as a best practice. Unsurprisingly, modern programming languages have capitalized on the efficiencies of code reuse from open-source components and made it easier than ever by providing package managers.

Many package managers use semantic versioning, allowing developers to automatically pull the latest major, minor, or patch versions of components each time they run a build. This convenience is excellent for ensuring the software project is always up to date with all the latest dependency releases, but what happens when the newest release is malicious? As more open-source projects adopt and implement continuous integration (CI) and continuous delivery (CD) pipelines in their builds, it will be easier than ever for an adversary to poison a project and release the malicious code into the wild under the auspiciousness of a simple patch.

There are over 5,000 open-source security advisories on GitHub today [6]. The zero trust model assumes that every project will have some security findings even without targeted supply chain attacks. Therefore, we need better vetting and monitoring of open source components throughout the component's life cycle. We can no longer trust open-source components; developers must understand the risks, and organizations must vet components before adopting them in their projects.

1.1 Motivation

In an era where software development is increasingly dynamic and interconnected, the reliance on open-source components and complex supply chains has be-

come ubiquitous. While driving innovation and efficiency, this integration also introduces many vulnerabilities and challenges, necessitating a reevaluation of security and verification practices. This dissertation explores critical aspects of software security, focusing on continuously verifying open-source components and understanding and combating next-generation supply chain attacks.

Open-source software has revolutionized how organizations develop, deploy, and maintain software. Its pervasive use across industries has led to improved collaboration, faster time to market, and significant cost savings. However, the benefits come with substantial risks, often overlooked until a security breach occurs. The first part of this dissertation, "Continuous Verification of Open Source Components Throughout the Life-Cycle," delves into the mechanisms and strategies necessary to assess, manage, and mitigate the risks associated with open-source software.

Building on the discussion of open-source component verification, the dissertation expands into the broader and increasingly sophisticated realm of software supply chain attacks. The chapter "Understanding and Combating Next-Generation Supply Chain Attacks" recognizes that adversaries are continually evolving their tactics, leveraging the complexity and trust inherent in modern software supply chains and that software engineers need to understand the risks. This section highlights the emerging trends and techniques employed by attackers and thoroughly examines defensive strategies that organizations can adopt. We argue for a proactive, intelligence-driven approach to supply chain security that anticipates and neutralizes threats before they materialize through secure software engineering education.

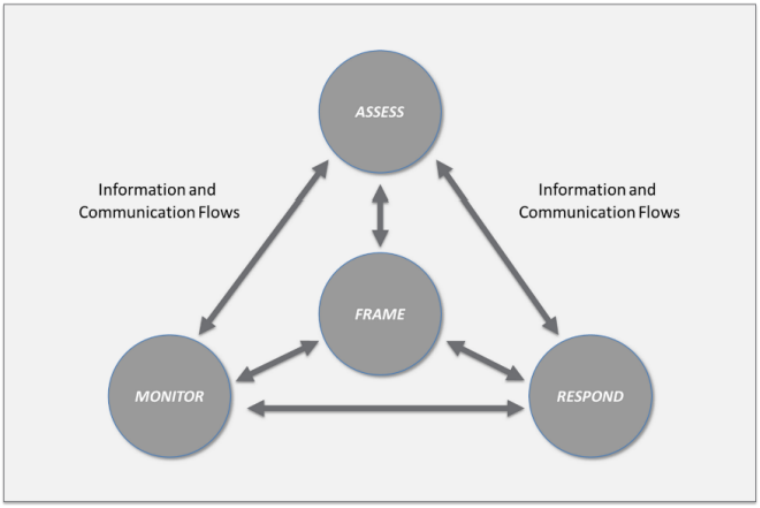


Figure 1.1: NIST 800-161 Risk Management Process

This dissertation aims to contribute to the field of software security by providing a detailed examination of current challenges and forward-looking strategies in open-source verification and supply chain protection. It seeks to bridge the gap between theoretical research and practical, actionable solutions, offering critical insights and recommendations for practitioners, researchers, and educators alike.

1.2 Problem Background

We took a multidisciplinary approach to solve our research goals by leveraging lessons learned and research from the fields of cyber security and cloud platform engineering in addition to software engineering for a truly holistic solution leveraging DevSecOps using the National Science and Technology Institute's (NIST) Risk Management Process as the lens through which we view the problem.

In May of 2022, the National Science and Technology Institute (NIST) released a special publication titled Cybersecurity Supply Chain Risk Management Practices

for Systems and Organizations (NIST SP 800-161 Rev.1) [7]. This publication outlines steps that organizations need to take if they are going to deliver software to the United States Government. This document was created to help answer the President's Executive Order (EO) 14028 on Improving the Nation's Cybersecurity [8].

NIST describes a Risk Management Process (RMP) in the 800-161 document as we see in Figure 1.1. The RMP is comprised of the following steps [7]:

1. Frame risk. Establish the context for risk-based decisions and the current state of the enterprise's information and communications technology and services and the associated supply chain.

2. Assess risk. Review and interpret criticality, threat, vulnerability, likelihood, 15 impact, and related information.

3. Respond to risk. Select, tailor, and implement mitigation controls based on risk assessment findings.

4. Monitor risk. Monitor risk exposure and the effectiveness of mitigating risk on an ongoing basis, including tracking changes to an information system or supply chain using effective enterprise communications and a feedback loop for continuous improvement.

We developed a novel risk framework and curriculum based on the RMP to address NIST's Configuration Management Control 10 (CM-10). Configuration Management Control 10 (CM-10) directly applies to open-source software components in software projects as described below.

Control Enhancement(s): When considering software, the enterprise should understand and review the open source community's typical procedures regarding provenance, configuration management, sources, binaries, reusable frameworks, reusable libraries' availability for testing and use, and any other information that may impact levels of exposure to cybersecurity risks throughout the supply chain. Numerous open source solutions are currently in use by enterprises, including in integrated development environments (IDEs) and web servers. The enterprise should [7]:

1. Track the use of OSS and associated documentation,
2. Ensure that the use of OSS adheres to the licensing term and that these terms are acceptable to the enterprise,
3. Document and monitor the distribution of software as it relates to the licensing agreement to control copying and distribution, and
4. Evaluate and periodically audit the OSS's supply chain as provided by the open source developer (e.g., information regarding provenance, configuration management, use of reusable libraries, etc.). This evaluation can be done through obtaining existing and often public documents, as well as using experience based on software update and download processes in which the enterprise may have participated.

1.2.1 Development

Software supply chain attacks targeting open-source components have increased 430% [9]. These exploits continue to grow in frequency and magnitude. Over the previous two years, this topic has become a priority among organizational leaders

and researchers. As a result, the software community has identified weak links in packages [4], created standards to highlight best practices for component vetting [10], automated vulnerability look-ups for dependencies [11], and widened the aperture for reporting vulnerabilities and malicious packages [12].

GitHub, the owners of NPM, have double-downed on their commitment to the NPM registry due to the increasing attacks on NPM packages. GitHub now requires 2FA for NPM package maintainers [13]. This is a step in the right direction as researchers have identified that hacking 20 high-profile developer accounts could compromise half of the NPM ecosystem [14]. However, more is needed, as researchers have identified six core weak links in the NPM ecosystem. One of which is that maintainers are using expired domains for their email accounts [4].

So, how do we overcome weak links in our supply chain? The Open Source Security Foundation has been doing much research into the topic of software supply chain protection as well. One of the projects we use extensively in our study is their Security Scorecards for Open-Source Projects. Although we did not use all the metrics, the scorecards provided a foundation for our use case, and then we added a couple. According to the researchers from Google, “The goal of Scorecards is to auto-generate a ’security score’ for open source projects to help users decide the trust, risk, and security posture for their use case. This data can also be used to augment any decision making in an automated fashion when new open source dependencies are introduced inside projects or at organizations” [10].

Understanding the package ecosystem and the community support around a package is essential, but a more holistic view is required to understand the risks before

incorporating open-source packages. The MITRE Corporation has been a faithful steward of maintaining two critical databases used for static code analysis. The first database is the Common Vulnerabilities and Exposures (CVE) database. This database contains a list of known package vulnerabilities [15]. The Common Weakness Enumeration (CWE) database is the second database they steward. This database contains "weakness types for software and hardware and is used as a baseline for weakness identification, mitigation, and prevention" [16].

We leverage the CVEs and the CWEs in our methods to identify known vulnerabilities in the open-source component and its dependencies. Then, we use the CWEs to check for known code signatures that allow the package to be compromised if measures are not implemented to prevent malicious attacks.

1.2.2 Security

The software community is not the only community handling an unprecedented rise in cyber attacks or supply chain exploits. The corporate information technology (IT) security communities have been handling and defending malicious attacks for decades. Information technology groups use a couple of stand-out models to protect their organizations from attacks. Many IT organizations understand what connects to their networks, plan to manage those assets, practice zero trust, and implement defense-in-depth.

NIST describes Zero Trust (ZT) as "the term for an evolving set of cybersecurity paradigms that move defenses from static, network-based perimeters to focus on users, assets, and resources" [17]. NIST defines defense-in-depth as, "information se-

curity strategy integrating people, technology, and operations capabilities to establish variable barriers across multiple layers and dimensions of the organization" [18]. Our methods implement elements from zero trust using defense-in-depth concepts by using automated tooling and policy throughout the life cycle of an open-source component.

1.2.3 Operations

There is a new way to think about the operations life-cycle, and some have called it the software development life-cycle in the cloud age [19]. The premise is that organizations can manage their IT operations using Day 0, Day 1, and Day 2 nomenclatures. We leverage Day 0, Day 1, and Day 2 in our evaluation. Day 0 is the design phase, where an organization considers how or if it will incorporate a new open-source component into a software project. Once the decision had been made to use the open-source component, we move into Day implementation. Day 1 is what goes into actually incorporating the component. This may include adding the component to the package or vendorizing it to make it available to the organization [20]. Day 2 operations occur after the package is included and running in production when maintenance and monitoring become priorities.

1.3 Contributions and Merits

There are three contributions from this dissertation. The first contribution is the risk reduction framework for organizations. The work offers a comprehensive framework for organizations to vet open-source projects effectively. This framework aims

to help organizations reduce their exposure to the inherent risks in using open-source software, guiding them to scrutinize and select open-source components wisely.

Our results indicate that our framework is effective at identifying key indicators for risk in packages and for managing risk after packages have been used in software projects. Additionally, we learned that communities play a key role in identifying malicious actions of others in open-source packages. We published a paper titled, ”Continuous Verification of Open Source Components in a World of Weak Links” at the 2022 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW) [21].

The second contribution is a curriculum for educating software engineers and engineering managers; recognizing the critical need for awareness and education in handling open-source components, the dissertation provides a specialized curriculum. This curriculum is designed to equip software engineers and engineering managers with the necessary knowledge and skills to identify and manage risks associated with open-source software.

Our evaluation of the curriculum shows students are better equipped to make informed decisions regarding open-source components and that they are aware of the risks and how to mitigate the risks of open-source components. The student’s average increased 32% and we demonstrated that our curriculum had a 84% gain in student scores.

The final contribution is an open-source Kubernetes-based development platform for secure software education and research to support the practical application of secure software engineering practices. The novel platform is intended for use in education and

research, providing a robust environment for learning, experimentation, and advancement in secure software engineering.

The platform has been used to teach 7 courses at the University of Colorado Colorado Springs and the United States Air Force Academy across undergraduate and graduate software engineering and advanced software engineering courses. Our environment has been downloaded over 8,400 times on Docker and the open-source project on GitHub has been forked by 3 developers [22] [23].

Collectively, these contributions aim to enhance the security and reliability of software engineering practices, particularly in the context of open-source software usage. The contributions address the theoretical underpinnings and practical applications needed to combat the challenges posed by next-generation software supply chain attacks and other security threats in the software development landscape.

CHAPTER II

LITERATURE REVIEW

For this chapter we focus on three main areas, two that are tangential to our thesis and one that is the core of our research. Figure 2.1 highlights the three main areas that we discuss. The first area is on open source adoption and research papers geared towards highlighting the increasing adoption of open source projects in software applications. The second area we look at is package management and how increasingly common it is for developers to utilize package managers to pull in dependencies. The third area, and the core of our thesis, is used to tie it all together and that is highlighting the risks of open source packages. We highlight the related research that discusses how to mitigate some of the risks associated with indiscriminately bringing in open-source packages using our VITAL approach. Lastly, we discuss current tools and best-practices and how these approaches differ from our risk reduction framework and curriculum.

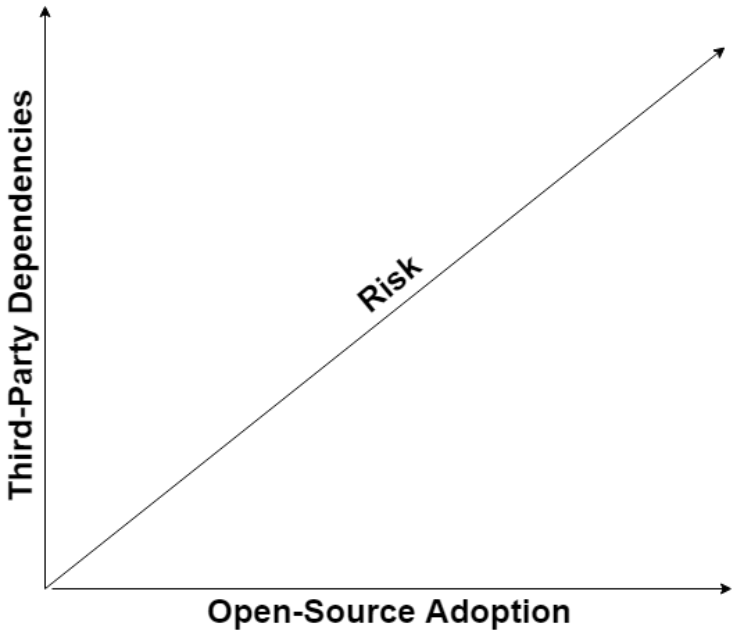


Figure 2.1: Third-Party Dependencies vs Open-Source Adoption and Risk

2.1 Open Source Adoption

Open source software has made a major impact to the state of software engineering. Code reuse is at an all-time high and software engineers are writing less and less code. According to a study from QSM, “On average, today’s developers deliver about 40% as much new and modified code per project as they did 40 years ago” [24].

Audris Mockus from the Avaya Labs Research in New Jersey wrote a paper about the large-scale code reuse in open source applications. Audris explains the benefits of code-reuse and how it has the potential to save developer time and software quality. His research explores how much of the code in open source software comes from code-reuse among other projects. He identified three patterns of reuse across 38.7 thousand unique projects and 10.7 million distinct file name paths. He noted that developers

re-use language translations for user messages. The next pattern he identified were instances of code to install modules for Perl. The last pattern that was widely reused included C language functions related to internationalization [25].

We were very surprised by Mockus's research. His research was conducted in 2007 and found code reuse widely implemented in internationalization and Perl module installations. We wounder how different his research would be today, 11 years later. We suspect there would be additional patterns identified involving JavaScript. We believe this because according to GitHub, there are more repositories on GitHub written in JavaScript than any other languages. JavaScript is also the top programming language by contributors [26]. This is one of the reasons why we evaluated NPM open-source packages in our case studies.

Haefliger et al. conducted similar research to Mockus in the area of code reuse in open source software but took a wider approach. The authors looked at the problem differently and highlighted some of the areas where software engineers take advantage of open source software. The authors also highlight some of the problems with using open source software from a commercial setting. The authors defined three broad forms of knowledge and code reuse among the software samples they analyzed. They defined a knowledge form for algorithms and methods, one for single lines of code, and the last one for components. Based on these forms of knowledge they came to the conclusion that developers reused code for three reasons. Developers want to integrate functionality quickly, developers preferred to write certain parts of the code over others, and developers could mitigate their development costs through code reuse [27].

2.2 Package Managers

Package managers are extremely popular among new and old software ecosystems. In 1993 to help manage the Linux kernel the first package managers began to appear [28]. Package managers help software engineers manage dependencies at build time. Modern day package managers pull in dependencies from multiple sources around the internet and include binaries or source. Many packages are provided by open source communities and are managed and contributed to by developers around the globe. Often software engineers only focus on what functionality the package provides.

In addition to the functionality, software engineers need to consider a few different factors when bringing in packages or source code from the internet. Software engineers need to look at the open source community that manages the packages and ask themselves a couple of key questions.

2.3 Addressing Component Risk Using VITAL

Including packages from third-parties can bring a certain amount of risk into any software project. We came up with the VITAL acronym to summarize our approach to addressing the risk. V stands for Value. We look at the value a package brings and we weigh it with our costs. I stands for Impact. We evaluate the impact the package provides and we examine if the impact can be generated in-house. T stands for Trends. When we decide to use a third-party package we need to research and inspect the community that supports the package to ensure that it is healthy. A stands for Administration. In

administration we look at what is involved with maintaining the package once we have included it. L stands for Licensing. We look at research that has gone into automating the discovery process of open source licensing in repositories.

2.3.1 Package Value

The first question that should be answered is does the value of the feature(s) outweigh the risk? Is the package one that we need, that will save us time, and provide functionality that we otherwise would not be able to get without expending resources that we may not have?

Bogart et al. from Carnegie Mellon, wrote a paper titled, “When it breaks, it breaks; How ecosystem developers reason about the stability of dependencies”. The authors focus in on centralized software management that is giving way to socio-technical ecosystems. An ecosystem, according to the authors, is as encompassing “multiple units of software, distributed over multiple systems, manged by multiples people, and organizations”. The authors surveyed package maintainers from two different ecosystems; NPM and R. They discovered that the package maintainers had a hard time of articulating why packages were stable other than by describing some of them as “classic”, “core”, or “stable”. The maintainers explained how difficult it was to stay aware of potentially import changes to upstram dependencies and that some of the maintainers tried to follow though mailing lists [29]. They all said it was too much information to digest. Over all the authors have identified a weak spot for practitioners and researchers to fill the void.

2.3.2 Package Impact

The second question the software engineer needs to consider is if an in-house solution could provide the impact the package provides. Is the feature trivial? Could the in-house team develop the same features or functionality? The software engineer really needs to do a quick cost-benefit-analysis and identify if the cost it would take to develop the functionality in house greater than the risk of including the package.

Abdalkareem et al, published a paper titled, “Why do Developers use Trivial Packages? An Empirical Case Study on npm”. In the paper the authors mine more than 230,000 NPM packages and 38,000 JavaScript applications. They found that trivial packages (packages that implement simple and trivial tasks) are common and make up 16.8 percent of the studied packages. The authors use the left-pad incident as a perfect example of a trivial package that broke many web applications. The authors also made an interesting find during a survey they conducted and made it a point to share that developers would rather use a trivial package than use code from Q and A websites such as StackOverflow or Reddit [30].

The trivial package paper highlights some of the dangers of using trivial packages in software applications. The authors bring up many interesting points and do a great survey of the current state of NPM. Some of the other concerns we looked at below which deal more with what impact dependencies will have brought into a system and how licensing of these packages could impact the project.

Wang and Capretz discuss a model to identify the impact dependency changes have on systems. They use ripple-effect analysis by calculating service dependency,

cohesion, and impact effect within and among services. They focus on web applications and make recommendations that would allow a developer to apply changes directly to the web service definition [31]. Although web applications as a service and web services are not the same thing as packaged dependencies, these services represent a new form of dependency that many applications are using. This paper focuses on the evolution of the dependency where the dependencies are no longer included in the software package and are managed by a third party.

In addition to identifying how dependencies will impact the system software engineers need to be aware of open source licensing. We discuss licensing below and we wanted to highlight one paper in this section on the subject. Colazo and Fang published a paper in 2009 on the impact of license choice on open source software development activity. This research not only relates to licensing but it also reinforces what we believe about tracking software ecosystems and the importance of identifying active communities when selecting dependencies. The authors demonstrate, using a study, that a social movement among developers working on open source software account of developers' voluntary participation in OSS projects and seem to predict the relationships between license choice and OSS development. In this particular study the authors identified that projects with a copyleft license attract more developers [32].

2.3.3 Package Trends

The third question software engineers need to ask is if there is a way to track the health of the package. Are the communities active? By active we mean that code is still being committed to the projects and that the most recent commits are relatively recent.

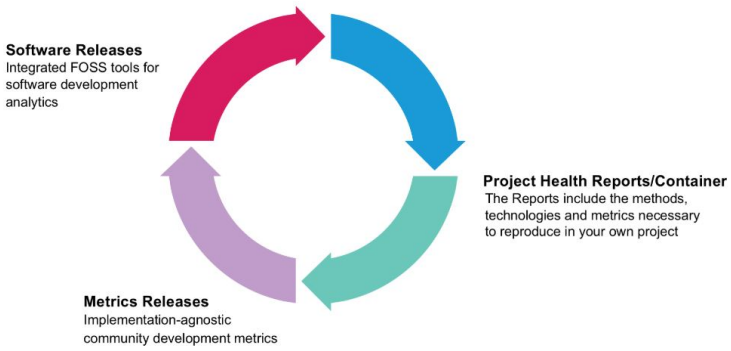


Figure 2.2: CHAOSS Goals

How does the community respond to vulnerabilities once they have been identified? What is the mean time to repair?

These are important questions and we have come across some excellent communities and tools to help provide answers. The first project we want to highlight is CHAOSS. CHAOSS is a, “Linux Foundation project focused on creating analytics and metrics to help define community health” [33]. Figure 1 shows the CHAOSS cycle for FOSS software. The project works to bring software applications to the open source community that help provide critical information about open source communities.

One of the projects from CHAOSS is called Prospector which was open sourced by RedHat in 2017. Prospector is a tool for automated collection and continuous tracking of a wide range of metrics of open source projects that are useful in evaluating the health and trends of the project [34]. Prospector scores open source projects on a number of different criteria. The first rating that it computes is infrastructure. The infrastructure measurement is based on a checklist of criteria such as existing project website, mailing list, contributor licensing statement or agreement, bug tracking system, and source code repository. The second rating is on community activity. Metrics such

as age of most recent release, commit activity, bug report activity, and blog posts are measured to come up with an activity score. The third measurement is one of code quality. This measurement is taken from bugs per KLOC and ratio of bugs opened to bugs closed. The last measurement is publicity. Metrics such as downloads and download frequency are collected along with number of blog posts per month and the interval at which blogs are published.

2.3.4 Package Administration

The fourth question is one of administration. What is the risk if the open source project is pulled from the internet? Is the team up to incorporating new versions of the package? Who will track the changes? How will vulnerabilities be detected and patched? A a few priorities to keep in mind with software as it pertains to software are confidentiality, integrity, and availability. At what point will a package erode one of those three priorities?

Khoo and Robey wrote a paper for the European Journal of Information Systems. In the paper they investigate the decision process to upgrade packaged software. The authors identify key areas that organizations should be aware of when including third-party packages that will need eventual upgrades. The first section they highlight is business needs. Business needs typically require that software is available and working. The second section the authors talk about is corporate and IT policies to mitigate software risks such as what policies need be in place for developers prior to leveraging open-source components. The next section is one on resource availability. Who is going to ensure the policies in place are followed? Who is going to patch the package? Who

is going to do regression testing on the new version of the package? The last section of the author's model talks about IT needs. What are the needs of operations? Who is going to handle new system requirements when package updates require new OS packages? Overall, the authors do a good job of highlighting, from a high level, what needs to be in place for carrying out upgrades [35]

Khoo and Robey started off at the high level. McCamant and Ernst begin to focus in on solving problems caused by package upgrades. McCamant and Ernst present a "technique to assess whether replacing a component of a software system by a purportedly compatible component may change the behavior of the system" [36]. The authors want to mitigate the risk of upgrading a dependency which will break the software. They approach the problem by comparing the observed behavior of the old package to the observed behavior of a the new package and permit the upgrade only if the behaviors are compatible. They use a test suite to compare the operational abstractions of the upgrade with the old package as exercised by the application. The test suite is typically provided by the vendor on release of an update. To showcase their work they used the selection sort application. They compared one version with another and identified that the new version, through testing, used swap differently in the new version than in the old. This is interesting research, especially if we could apply it in an automated way, so that we could automate testing upgrades before incorporating the upgrades into the software.

McCamant and Ernst worked to identify what upgrades to include in a system as to not break the current system. Nguyen and Tran have provided research into identifying vulnerable software packages by using dependency graphs. When software

engineers include third-party packages in their software projects they don't just include the package. They include the package's dependencies as well. This raises some serious concerns. How would the software engineer know that the package they just included has vulnerable dependencies? Nguyen and Tran attempt to solve this issue. They come up with a vulnerability prediction model where they used classification, are you vulnerable, and a ranking, how likely you are vulnerable. They go on to discuss precision and recall and they explain how a good prediction model should obtain high precision and recall. They also explained how it seems that the relationship between recall and precision appear to be a zero-sum game. If recall is good then precision probably is not. If precision is good then recall probably is not.

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

The authors used static code analysis to loop over a packages dependency graph and then compare the dependency to known vulnerability databases such as Bugzilla and the National Vulnerability Database. As the dependencies are compared to known vulnerabilities a score would come back [37].

We do not know if the research provided by Nguyen and Tran was used in industry but there are providers that will traverse dependency trees from source code

repositories and notify software engineers when there are known vulnerabilities in packages included in their projects. One of these services is called Snyk.

Snyk ties in to popular source code repositories such as GitHub and BitBucket. The application maps the full application dependency tree, finds vulnerabilities in all of the open source dependencies, continuously tests for newly disclosed vulnerabilities, and tests dependencies against a vulnerability database. When a vulnerability is detected Snyk has the ability to generate a pull request against the project to update the dependency that has the vulnerability to a version that has patched the issue. It also has the ability to send alerts via email and chat to notify developers when there is a security vulnerability [38].

Sabetta and Bezzi from SAP Security Research wrote a paper on an approach that uses machine learning to analyze source code repositories and identify commits that are likely to fix a vulnerability. With Snyk the company leverages preexisting databases to compare dependencies against to highlight vulnerabilities. Sabetta and Bezzi propose taking it a step closer to the source code by using static code analysis on commit messages. The authors gather commit messages and classify the different messages into two categories, log and patch. They used machine learning algorithms and natural language processing methods to identify and group the different commit messages. The patch messages are important and from these messages the authors are able to come up with a certain level of precision that the package was patched to fix the bug [39]. We find this approach interesting and bundled with some of the other research in this survey paper might be able to further the state of the art for bug detection.

NPM has made headlines lately for security issues. Back in May, it removed a package called GetCookies from the NPM repository. Snyk and the research done by Sabetta and Bezzi most likely would not have caught this particular security issue. A project contributor had, on purpose, tried to hide a backdoor in the package. The NPM management was alerted by the project's community that there was a security issue. The impact, initially, seemed minimal. No one had downloaded the package since the backdoor was added. However, GetCookies, had been included in a popular package called Mailparser that had 66,000 downloads weekly [40]. Luckily, the security issue was identified early and remediation happened quickly. This is why analyzing communities for trends such as community involvement is so important as we discussed earlier.

Decan et al. presented a paper titled, "On the impact of security vulnerabilities in the npm package dependency network" at the Mining Software Repositories conference in May of 2018 in Gothenburg, Sweden. In the paper the authors present an empirical study of nearly 400 security repos over a 6-year period in the NPM dependency network. The authors highlight that Snyk released an analysis of 430,000 websites and of those no less than 77 percent of those analyzed ran at least one front-end library with a known security vulnerability [41]. The authors aim to answer a series of questions regarding package vulnerabilities. The first question is how many packages are known to be affected by vulnerabilities. The second question is how long do packages remain vulnerable. The third question is when are vulnerabilities discovered. The fourth question is when are vulnerabilities fixed. The last question is when are vulnerabilities fixed in dependent packages. Overall, the authors did an excellent job of identifying impor-

tant questions and highlighted an area that should be of concern for software engineers as they use third-party dependencies.

As if all of this is not enough to worry about, what will the team do when a package is removed from the source repository. This happened back in March of 2016. A developer, Azer Koçulu, pulled a package called left-pad [42] along with 249 other modules from NPM. NPM is a popular package manager for JavaScript projects. When he pulled left-pad along with the other 249 modules it broke some of the most visited website on the internet including Facebook and Airbnb [43].

There are a couple of tools that help with this exact issue in industry. The first one that we want to talk about is Artifactory. Artifactory provides a feature called remote repositories. Once a remote repository is configured, software engineers can pull artifacts from the source repository, through Artifactory, and then to the software project. By pulling in dependencies through Artifactory, Artifactory is able to cache the package [44]. This prevents the software project from breaking if the package is removed from the source repository.

2.3.5 Package Licensing

The fifth, and final question, question that businesses need to be addressed is one of licensing. Licensing affects the intellectual property rights of software. Licenses can impact organizations' rights to sell and distribute software through copyleft, and each licenses brings different restrictions as to what an organization can do with the software after it is developed. Some licenses allow organizations to do whatever they want with the code or package. This includes modifying, distributing, and selling the

software. Other packages allow modifying and distributing but not selling. Some do not allow modifying or distributing. The risk to an organization if a license is not vetted or identified is that it could render, depending on the license, intellectual property rights void.

Mustonen does an excellent job of summarizing what copyleft is and why developers should care. According to Mustonen copy left is, "To copyleft a program, the programmer, besides copyrighting the program to himself, also signs a General Public License (GPL) granting everyone the right to use, modify and distribute the program on the condition that the licensee also grants similar rights over the modifications he has made" [45].

The GPL license is an interesting license and as German et al discovered below it is one of the most popular licenses for open source projects. One of the risks associated with using the GPL license is that it caries the risks of copyleft. If an organization wants to maintain intellectual property rights it might be best not to include a project with GPL. We talk more about license identification and methods for automating such identifications.

Alspaugh et al. worked on a project to automate the detection of software licenses by using a scheme that describes the software licenses in a formal and less ambiguous form than natural language. The authors start with a tuple of actor, operation, action, object for expressing a right or obligation. They are then able to scan a software repository and, using static code analysis, follow a project's dependency tree to identify any licenses included in the dependencies. Using the approach they are able to identify and calculate the copyright rights and obligations for the systems' configuration. Currently

the license traceability analysis tool works for such licenses as the GPL and Mozilla Public Licence (MPL) licenses [46].

German et al. worked on a similar problem as Alspaugh et al. German et al. attacked the problem differently using sentence-matching methods to automate detecting licenses in source code files. Instead of using tuples these authors used, essentially, regular expressions to detect licenses by-inclusion and by reference. For example the authors look at the license statement and then from there are able to identify if the license is embedded in the file. The authors describe the licensing statement in four sections. The first section is a list of copyright owners. The second section is a list of authors. The third section are the licenses that cover the file. The fourth section is made up of warranty and liability statements. If a license is not embedded in the license file with the statement then the authors have to look closer at the licensing file to see where the licenses can be found. The authors work through an empirical study and show that they are able to identify licenses with GPLv2 being the most popular license in the software they analyzed [47].

There are other licenses that are better to use from a organizational standpoint if selling and maintaing intellectual property rights software are the goals. These licenses are typically BSD, MIT, and Apache to name a few. These licenses usually allow for organizations to sell, distribute, and maintain rights to the software.

The research highlighted in this section is important because it saves organizational rights and developer time when exploring packages to include in the software project. Automating the discovery of licenses might not seem very interesting but it has a major impact.

2.4 Current Tactics, Techniques, and Procedures

In our thesis, we developed a framework for evaluating and auditing open-source software supply chains for risk. The Open Source Security Foundation (OpenSSF) released its first guide for evaluating open-source software in 2021 [48]. Their guide aligns with our VITAL framework and most of our day 0 controls discussed in Chapter III. However, their guide does not address day 1 or day 2 protections like our framework, and we have additional vetting criteria for our day 0 controls that their guide does not address.

In 2021, Google released a high-level framework called "Know, Prevent, Fix: A framework for shifting the discussion around vulnerabilities in open source." They discuss topics similar to those in the OpenSSF guide. However, they released a new application called Scorecards that automates the evaluation of open-source software communities [49]. The Scorecard software was eventually turned over to OpenSSF [50]. When identifying community risks inside our packages, we leverage a subset of the Scorecards in our framework for our day 0 controls. However, the scorecards do not account for our controls, such as 'dependents,' that we evaluate in our framework. Additionally, the Scorecards do not address day 1 or day 2 controls, which our framework evaluates, giving our approach additional defense in depth protections.

As we developed our novel framework, we learned that many software engineers were unaware of the many risks open-source software brings to an organization. There is a need to increase education as many software practitioners are becoming the weakest link in supply chain attacks [51]. We can create a framework that holistically evaluates

open-source components. However, software engineers need to understand the context of why we need to evaluate components so the framework is useful.

We developed a curriculum to fulfill step 1 of the Risk Management Process (RMP), establishing the context for risk-based decisions. First, we needed to identify the risks of open-source components, and then we needed to explain how to defend against the risks.

The Linux Foundation sponsors a Community Health Analytics Open Source Software (CHAOSS) project that creates metrics, metrics models, and software to better understand open source community health on a global scale [33]. The CHAOSS risk working group has done an excellent job breaking down open-source software risks into six core focus areas: Business Risk, Code Quality, Licensing, Security, Transparency, and Dependency Risk [52]. These focus areas were our starting point when we began building the curriculum that we talk about in Chapter V helping us frame the problem for our framework. These focus areas are logically separate, which makes it easy for students to follow and fill in a critical knowledge gap that many software engineers have regarding open-source software risks [51].

There are many researchers looking into secure software development education. These approaches look at fundamental concepts and principles of security such as critical security risks based on OWASP top 10, secure software requirements, design, implementation, and software sustainment [53] [54] [55]. These methodologies do not teach or examine the risk of open-source components.

Our curriculum is novel because it goes further than explaining the risks that the CHAOSS Risk Working Group has identified. We expand on it using the research

from Chapter III, where we show students how to mitigate the risks for an applied and well-rounded approach. Our curriculum and framework provide the students with the tools they need to understand and meet the intent of NIST 800-161 as outlined in NIST's Risk Management Process.

CHAPTER III

CONTINUOUS VERIFICATION OF OPEN SOURCE COMPONENTS THROUGHOUT THE LIFE-CYCLE

In this chapter, we develop and present innovative methods for continuous verification throughout a component life-cycle to provide insight for understanding and monitoring the risk of incorporating open-source components. We take a holistic approach to analyzing components for risk of underlying vulnerabilities. We define six controls organizations can use to protect themselves from malicious supply chain attacks. Specifically, we utilize the controls to create a repeatable method to understand the risk of incorporating an open-source component throughout the component's life cycle.

We first describe attack vectors malicious actors leverage when targeting open-source components. We then describe the six controls we developed to protect against those attack vectors. Lastly, we apply the controls to an organizational setting to understand how the controls might perform in a real-world situation.

This chapter provides empirical evidence that the controls selected protect organizations from open-source supply chain attacks. In our study, we identified five open-source projects that malicious actors compromised. We analyze the projects using our controls and conduct a tabletop exercise to understand the effectiveness of our controls in protecting an organization from compromised packages. We found packages across multiple ecosystems that had been poisoned, ecosystem such as NPM, Ruby Gems, and Python. For each package, we considered how often the package was downloaded, how long it took the community to respond to the incident, ensured that the package had legitimate use before becoming malicious, and the use case for the package.

We divided the six controls into three categories using the phases for Day 0, Day 1, and Day 2 aligned with the the known knowns, the known unknowns, and the unknowns unknowns. These three categories provides us the opportunity check for known issues with a package, identify potential issues, and protect against issues organizations have not identified.

At the end of each case study was compare our framework against OpenSSF's Scorecards. The comparison looks at the average score of our framework and the Scorecards to determine the likely hood that a package will respond to malicious activity. The average score is translated to a rating of high risk, medium risk and low risk. High risk is an average score less than or equal to 5, medium risk is a score greater than 5 and less than 7.5, low risk is a score greater than or equal to 7.5 and less than or equal to 10. The risk is a direct correlation to how long it will take a project to respond to malicious activity, Time to Repair (TTR). High risk is measured in weeks or never, medium risk is measured in days, and low risk is measured in hours. We chose the threshold of hours,

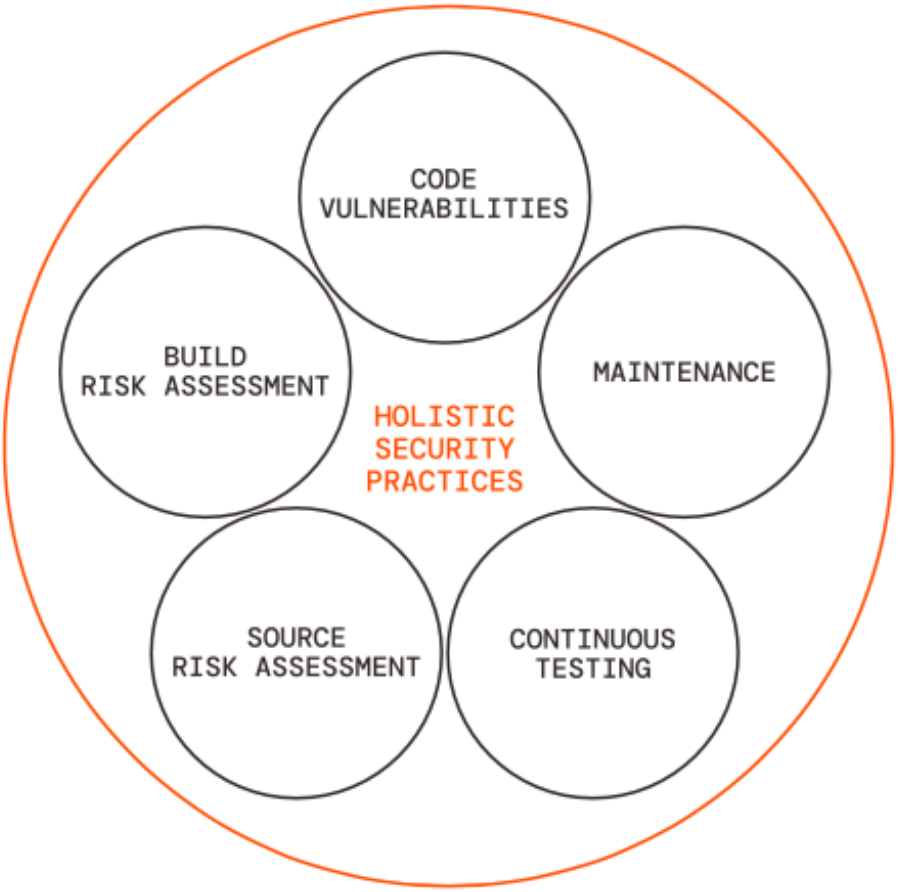


Figure 3.1: Scorecards Holistic Approach

days, and weeks for our risk based on our findings during our experiments. We noted that some packages, 60% of our case studies, are fixed quickly in a matter of hours. In 40% of the case studies, the communities fixed the packages in a couple of days but less than a week. We realize that not all packages have the same urgency to repair in a couple of hours or days, so we made a catch for those with a threshold more significant than a week.

3.1 Day 0: The Known Knowns

We can learn much information about the current state of a project by looking at the package's dependencies, source code, and community. We have identified four controls which are used to understand the current state of a package and the package's community and can inform the decision of whether or not to use the package. These four controls will be used throughout the package's life cycle, beginning at Day 0.

C1: checks the package for known vulnerabilities in a package's dependencies and in the package itself. This is accomplished by leveraging a query against the CVE database using a tool like Depdendabot or Snyk. We also evaluate the package for dependents. The more dependents a package has means there are more communities invested in the package staying secure. Figure 3.1 shows the Scorecards methodology for holistic security practices that we check in our Day 0 controls [56].

C2: checks the source code for known weaknesses in the code base using static code analysis, which leverages CWE information.

C3: looks at the package's community to understand the makeup of the project's maintainers. This control examines the number of companies maintaining a project and recent activity within the last 90 days. Multiple companies supporting a project and recent activity are signs of a healthy project.

C4: looks at the hygiene of the package. This control looks for dangerous workflows, signed commits, signed packages, and branch protection.

The Day 0 controls are automated using a combination of a subset of the OpenSSF Scorecards and our implementation for identifying and weighing dependents. We lever-

```
#!/bin/bash
docker run -e GITHUB_AUTH_TOKEN=$1
 ↪  gcr.io/openssf/scorecard:stable --show-details
 ↪ --repo=https://$2 --checks Branch-Protection,
 ↪  Code-Review, Contributors, Maintained,
 ↪  Dangerous-Workflow, SAST, Vulnerabilities,
 ↪  Signed-Releases | grep Aggregate > tmp.txt
sed -i 's/Aggregate score: \([0-9]*\(\.[0-9]*\)\?\)
 ↪  \/ 10/\1/g' tmp.txt
/usr/bin/python3 ./dependents.py $2
```

Figure 3.2: Bash Script to Run Scorecards and Dependents Metrics

age the branch protection, code review, contributors, maintained, dangerous workflow, Static Application Security Testing (SAST), vulnerabilities, and signed releases heuristics from the Scorecards project [56]. We picked our subset based on our research requirements and the research conducted by Zahan et al. Their research identified the scorecards' four most crucial security heuristics: maintained, code review, branch protection, and security policy for understanding vulnerability counts [57].

Our dependents' heuristic is integrated and weighted with the Scorecard's aggregate score from their eight metrics. The dependents metric has yet to be available in the GitHub API as of version 4, so we go directly to the GitHub website and scrape the data from the user interface. We automate this process in two parts. First, we call a bash script, Figure 3.2, that executes the Scorecards and saves the resulting aggregate score in a text file. Then, we execute a Python script, Figure 3.3, to get the dependents data, weigh it, add it to the aggregate score, and return a risk level.

```
import requests
import sys
from bs4 import BeautifulSoup
import math
score = 0
if len(sys.argv) > 1:
  argument = sys.argv[1]
  parts = argument.split('/', 1)
  url = 'https://'+ parts[0] + '/' + parts[1] +
   ↪  '/network/dependents'
response = requests.get(url)
if response.status_code == 200:
    soup = BeautifulSoup(response.text,
     ↪  'html.parser')
    a_href = '/' + parts[1] +
     ↪  '/network/dependents?dependent_type=REPO'
    a_element = soup.find('a', href = a_href)
    if a_element:
      number_of_dependents = a_element.text.strip()
      number_of_dependents =
       ↪  number_of_dependents.split('\n', 1)
      number_of_dependents =
       ↪  number_of_dependents[0].replace(",", "")
      number_of_dependents =
       ↪  int(number_of_dependents)
      if (number_of_dependents > 275):
        score = 10
      else:
          score = number_of_dependents / 2.75
          score = math.ceil(score) / 10
    else:
        print("Error: Anchor element not found.")
else:
    print(f"Failed to fetch the page, status code:
     ↪  {response.status_code}")
file_path = 'tmp.txt'
with open(file_path, 'r') as file:
    lines = file.readlines()
scorecard_scores = lines[0].strip()
scorecard_scores = float(scorecard_scores)
total = scorecard_scores * 8
all_together = ((score * 10) + total) / 9
all_together = round(all_together, 2)
if all_together >= 7.5:
    print("Low Risk")
elif all_together >= 5:
    print("Medium Risk")
else:
    print("High Risk")
```

Figure 3.3: Python Code to Scrape the GitHub UI

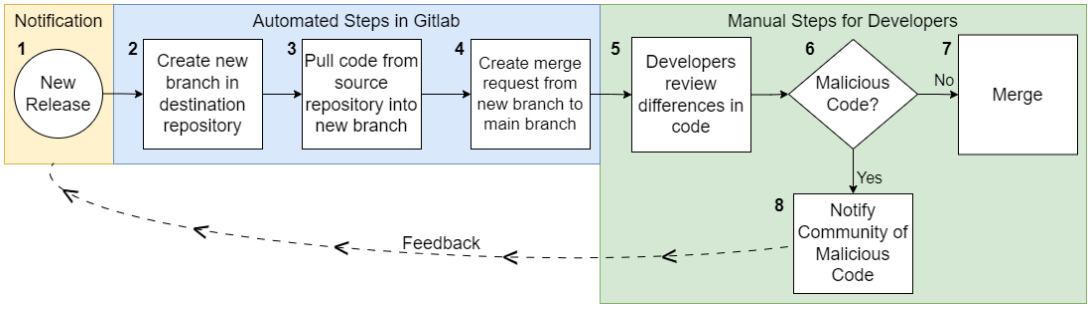


Figure 3.4: Day 1 Process

3.2 Day 1: The Known Unknowns

Once a package moves into Day 1 and has been included in our software, we must leverage additional controls to protect against the known unknowns. A core tenant of security tells us to assume everything will be compromised at some point. Whether by a dependency, a code update with a bug, or a malicious maintainer.

C5: is a policy that dictates that no open-source artifact that is not built by a trusted source will be included. The open-source component's source code will be forked and built in-house, and the artifact will be stored in a central location. When a new package version is released, we will ingest the change, conduct a secure code review [58], and scan the code using C1, C2, and C3. This heavy policy gives the best chance to detect a problem with a dependency, new code that can be exploited, or code added by a malicious maintainer. This is especially true when we can not trust that packages published on sites such as NPM or RubyGems came from their respective communities [13] [59].

Once an organization has initially forked the code and conducted the code review, the organization can automate the code checkout and merge request process for future releases. Figure 3.4 shows the flow of the automation and manual processes required:

1. The organization can create a listener for new releases from the source repository and trigger the destination's CI pipeline for new code commits.
2. The CI pipeline will create a new branch for storing the new code from the source repository.
3. Next, the pipeline will clone the latest release branch from the source repository to the new branch.
4. Then, the CI pipeline will create a merge request back to the release branch of the destination repository.
5. The merge request will allow the organization's developers to conduct a code review identifying the differences between the source and destination. This process will give developers insight into the changes between their trusted repository and the new release from the source repository.
6. The developers will check for malicious code during the code review. The code review is manual because there is not a good way to automate the code review process [60].
7. If they do not detect malicious code, they will merge the new branch to their main branch. The merge will kick off another round of the CI pipeline, which will run the build, test, and publish stages as seen in Figure 3.6.

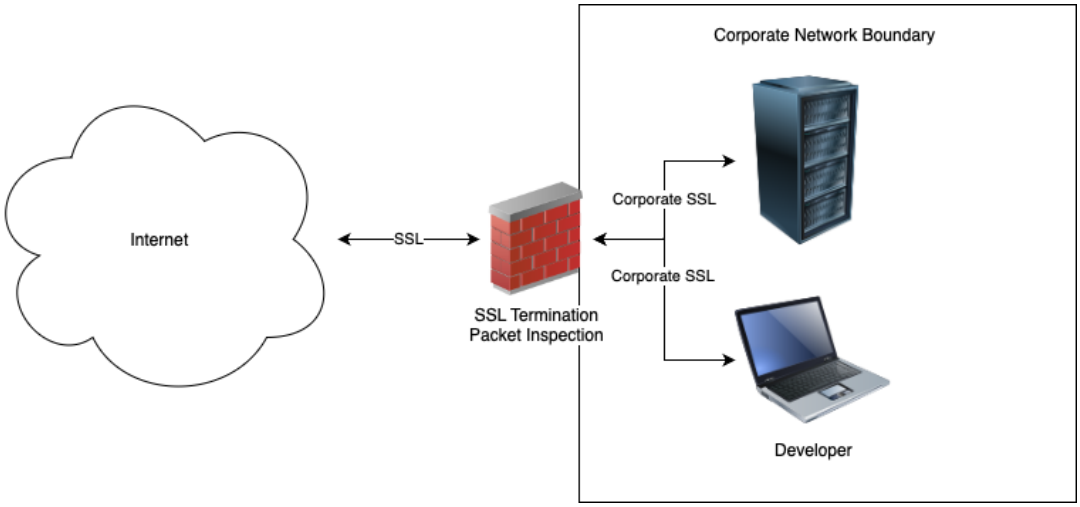


Figure 3.5: C6 Network Diagram

8. Otherwise, the developers will give feedback to the package's community that there is malicious code in the new release. The feedback leads to a new release of the software where the process is repeated.

3.3 Day 2: The Unknown Unknowns

After a package has been built internally and ingested into a software project, we need to protect that project from the unknown unknowns of the package. These unknown unknowns are malicious items we missed using the first five controls and are typically identified during Day 2 operations when the package is in a running environment.

C6: is a network perimeter defense around the development and production environment of the software project. This perimeter runs at the network layer of the OSI model and executes packet inspection. This control monitors the activity of the software as it passes data over the network and onto the internet. The control will act on

unexpected changes in the software's behavior or when the software attempts to pass or retrieve data from unknown sources. Figure 3.5 shows how internet traffic is directed to the corporate network boundary firewall, packets are decrypted, and passed into the corporate network in the clear. This allows the firewall to conduct packet inspection for malicious URLs and payloads. The team monitoring the network traffic controls will alert the software developers when applications are flagged and will automatically block requests made to hosts that are not on a predefined allow list.

3.4 Results

This section describes the results of our findings across the use cases. We applied the methods to five components that were compromised. The components come from different ecosystems such as Javascript, Ruby, and Python. Each of the components had been compromised at some point in the past. We take a look at each of the components using our framework and conduct a tabletop exercise to measure the effectiveness of the controls. Tabletop exercises are an excellent tool to validate roles and responses to incidents in emergency management and when understanding information technology incidents [61] [62].

3.4.1 Case Study: UAParser.js - NPM

UAParser.js is a "JavaScript library to detect Browser, Engine, OS, CPU, and Device type/model from User-Agent data with relatively small footprint" [63]. On October 23, 2021, this NPM package was modified to include malicious code by an

outside actor using a maintainer's compromised account. The malicious code injected into the UAParser.js package attempted to install coinminer and harvest user/credential information [64]. The plugin was used by big tech companies such as Facebook, Slack, IBM, HPE, Dell, Oracle, Mozilla, Shopify, and Reddit. Users reported downloading Trojans to their local environments after updating the compromised release. The package owner was notified by the community, who remediated the problem in a couple of hours and pushed updated releases soon after that removed the malicious components.

We put the package through our controls using the package's community metrics as they are today. For the static code analysis, we checked out the project's repository and went back to a commit that occurred closest to, but before the breach happened, which was a commit from October 6, 2021 [65].

3.4.1.1 Control Execution

We executed the six controls using the UAParser.js project. Below are our findings.

C1. UAParser.js does not utilize third-party dependencies, so this task was simple and limited the attack surface of the package.

C2. We leveraged eslint, semgrep, and nodejs-scan tools to perform static code analysis. As Table 3.1 shows, we identified 14 critical and 18 medium vulnerabilities. Although all these findings may not be actionable by a malicious actor, they give the organization insight into potential attack vectors to which the package opens the organization up. At this point, the organization can accept the risk or triage and confirm the findings.

Score	Name	Reason
10 / 10	Dependencies	0 dependencies found
10 / 10	Dependents	4,993,856 dependents found
0 / 10	Static Code Analysis	14 critical and 18 medium vulnerabilities identified

Table 3.1: UAParser.js - C1 and C2 Metrics

C3. We utilized metrics and scoring from the scorecards. As Figure 3.2 shows, this package has maintenance support from 10 different companies. This provides confidence that the package is maintained because it is in the company's best interests. However, the package does not appear to be well maintained. In the last 90 days, there have only been 2 commits. We also did not detect any activity in project issues by collaborators, members, or project owners in the last 90 days, but 3 new issues opened in that time.

Score	Name	Reason
10 / 10	Contributors	10 different companies found
1/10	Maintained	2 commit(s) and 0 issue activity found by maintainers in the last 90 days

Table 3.2: UAParser.js - C3 Metrics

C4. We utilized the branch protection and dangerous workflow metrics from the scorecards in this control. Unfortunately, the scorecards do not provide a way to check for signed commits or packages outside of GitHub. This project publishes packages to NPM. Our findings for C4 are in Table 3.3. We identified that the project does not include branch protection on the main or release branches. We also identified that the package does not require maintainers to use signed commits, and the packages are not signed.

Score	Name	Reason
0 / 10	Branch-Protection	Protection not enabled
10 / 10	Dangerous-Workflow	No dangerous workflows
3 / 10	Signed-Commits	Used in some cases
0 / 10	Signed-Packages	False

Table 3.3: UAParser.js - C4 Metrics

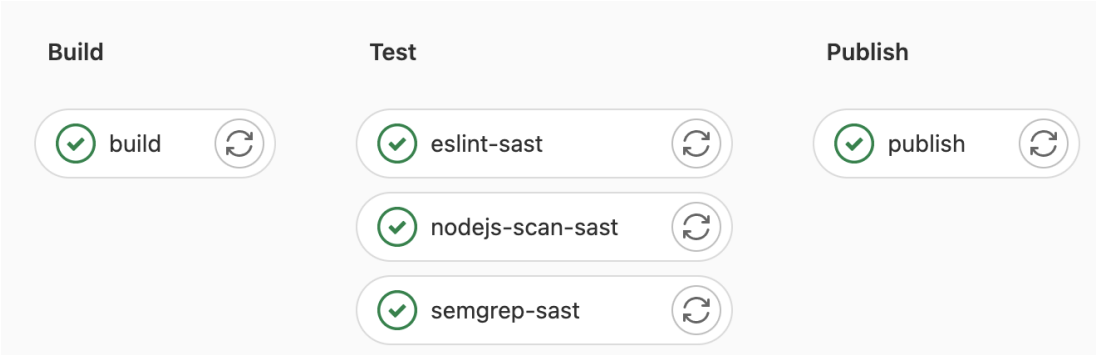


Figure 3.6: UAParser.js - C5 pipeline

C5. Building the package took little effort. We had already forked the repository from GitHub for our analysis. The package does not depend on outside dependencies, so a simple ‘node install‘ was sufficient. We can ingest new changes from the upstream project and validate them using a code review before ingesting and publishing the new package. Figure 3.6 shows our CI pipeline for implementing this and previous controls’ build, test, and publish phases.

C6. This control would have identified the package attempting to contact an unknown URL to download the coinminer.

3.4.1.2 Evaluation

The package was compromised for a couple of hours for us this is a good use case in how effective a community is to responding. Although our controls identified many

potential problems with the package beginning on Day 0, culminating with unexpected behavior identified on Day 2, our Day 0 control for dependents would have made a compelling case to use the package anyways. Comparing our findings to that of the OpenSSF's Concise Guide for Evaluating Open Source Software, in Table 3.4 we can see their recommended controls might have identified most of the Day 0 risks which would make developers believe this package was a problem. Despite that it is used in over 4 million packages. OpenSSF's controls would not have identified the malicious behavior on the network because their framework does not advocate or use continuous network monitoring like our Day 2 controls which is a core tenant of NIST's Risk Management Process.

Operations	**Our Framework**	**Scorecards**
Day 0	Low Risk	Low Risk
Day 1	Effective	N/A
Day 2	Effective	N/A

Table 3.4: UAParser.js, TTR: Hours - Framework Comparison

Day 0. While performing controls 1-4, we identified some indicators that would make it hard to defend the project against weak links. First, there were already critical vulnerabilities in the code base that malicious actors could exploit. The project no longer seemed to be maintained despite many different corporate contributors. The project's community did not protect the main or release branches from maintainers directly committing to those branches and did not enforce signed commits or signed packages. These controls alone may have identified enough risk for an organization to choose not to use the package.

Day 1. Executing C5 and bringing the package under internal control was simple because the package resided in one repository and did not have external dependencies. As a result, we were able to build the project and publish the resulting artifact. If an organization had chosen to use this project before October 23, 2021, this control would have required the organization to conduct a code review before ingesting the malicious code. This code review may have identified the malicious code.

Day 2. The observability that C6 provides would have provided another chance for the organization to identify the malicious behavior of the package and take actions to protect itself had the code review in C5 missed it. The malicious code went outside of the network to download malicious tooling. C6 would have identified the request, flagged it, and created a notification in the architecture.

Overall, the methods identified many risks within the UAParser.js package but the fact that it was such a popular package, used by over 4 million packages could bring the risk down. Ultimately, it is up to the organization to accept the risk. However, our controls would still have provided protection even with an organization accepting the risk of using the UAParser.js before October 23, 2021.

3.4.2 Case Study: REST Client - RubyGems

REST Client is a "simple HTTP and REST client for Ruby, inspired by the Sinatra microframework style of specifying actions: get, put, post, delete" [66]. The package was modified on August 14, 2019, to include a malicious backdoor. The malicious actor had a script that would download code from Pastebin.com that was reportedly used to mine cryptocurrency. On August 19, a CVE was generated, RubyGems removed the

affected Gems, and the community pulled the malicious code [67]. As with the last case study, we checked out the project's repository and returned to a commit that occurred closest to, but before the breach happened, a commit from March 28, 2019 [68].

3.4.2.1 Control Execution

We executed the 6 controls using the rest-client project. Below are our findings.

C1. Like the UAParser.js, this project does not utilize third-party dependencies, so this task was simple and limits the attack surface of the package.

c2. We leveraged the brakeman tool to perform static code analysis. Unsurprisingly, we found no vulnerabilities in the source code, as Table 3.5 shows. The project is mature, and the functionality does not change often.

Score	Name	Reason
10 / 10	Dependencies	0 dependencies found
10 / 10	Dependents	179,265 dependents found
10 / 10	Static Code Analysis	0 vulnerabilities identified

Table 3.5: Rest-Client - C1 and C2 Metrics

C3. We utilized metrics and scoring from the scorecards. As Table 3.6 shows, this package has maintenance support from 26 different companies. There have not been any commits in the last 90 days, and we did not detect any activity in project issues by collaborators, members, or owners of the project in the last 90, and there have not been any issues opened in the last 90 days.

C4. We utilized the branch protection and dangerous workflow metrics from the scorecards in this control. Our findings for C4 are in Table 3.7. We identified that the project does not include branch protection on the main or release branches. We also

Score	Name	Reason
10 / 10	Contributors	26 different companies found
0 / 10	Maintained	0 commit(s) and 0 issue activity found by maintainers in the last 90 days

Table 3.6: Rest-Client - C3 Metrics

identified that the package does not require maintainers to use signed commits, and the packages are not signed.

Score	Name	Reason
0 / 10	Branch-Protection	protection not enabled
10 / 10	Dangerous-Workflow	no dangerous workflows
3 / 10	Signed-Commits	used in some cases
0 / 10	Signed-Packages	false

Table 3.7: Rest-Client - C4 Metrics

C5. Just like the previous case study, we had already forked the repository from GitHub into GitLab for our analysis. As new updates are released, we can conduct a code review and merge the change.

C6. This control would have identified the package attempting to contact an unknown URL at Pastebin.com to download the malicious script.

3.4.2.2 Evaluation

Our methods identified on Day 0, one weak link. Day 1 and 2 controls provided the most protection in this case study.

Day 0. While performing controls 1-4, we identified that the package had not been maintained or modified in over 90 days. This is a weak link, but due to the nature of the component and that REST calls do not often change, the perceived risk could be less.

Day 1. Executing C5 and bringing the package under internal control saved the organization. The organization could conduct a code review of the malicious code before incorporating the changes. If the organization had overlooked the malicious code, merged, and moved to Day 2 operations, the additional controls would have caught it.

Day 2. C6 would have identified the call to Pastebin.com and blocked the outgoing request.

Overall, this project was compromised for 5 days. Our framework's Day 1 and Day 2 controls would have provided the most protection for an organization in this case. Comparing our framework to OpenSSF's guide, OpenSSF says this project is high risk as you can see if Table 3.12 while ours says it is a medium risk we also have additional controls in place for defense in depth. If our Day 1 control did not identify the malicious code then our Day 2 controls would have protected the organization.

Operations	**Our Framework**	**Scorecards**
Day 0	Medium Risk	High Risk
Day 1	Effective	N/A
Day 2	Effective	N/A

Table 3.8: Rest-Client, TTR: Days - Framework Comparison

3.4.3 Case Study: Connect Kit - CDN

According to Connect Kit project, "Connect Kit enables developers to connect their dApps to Ledger hardware wallets using the Ledger Extension or Ledger Live" [69]. Hackers compromised the code behind a cryptocurrency protocol used by multiple web3 applications and services on December 14, 2023 [70]. The community noticed

the hack within 6 hours and released an compromised version within 40 minutes. According to Ledger's spokesperson, a former employee was a victim to a phishing attack which gave the malicious actors access to the employee's NPM account. The malicious code used a rouge wallet to reroute funds to the attackers wallet [70]. Reported losses from the hack are on the order of $850,000 [71].

3.4.3.1 Control Execution

We executed the six controls using the Connect Kit project. Below are our findings.

C1. Connect Kit has 1,749 dependencies with 1 critical vulnerability, 5 high vulnerabilities, and 5 medium vulnerabilities.

C2. We leveraged eslint, semgrep, and nodejs-scan tools to perform static code analysis. As Table 3.9 shows, we identified 1 critical and 5 high vulnerabilities. Although all these findings may not be actionable by a malicious actor, they give the organization insight into potential attack vectors to which the package opens the organization up. At this point, the organization can accept the risk or triage and confirm the findings.

Score	Name	Reason
3 / 10	Dependencies	1,749 dependencies found 1 critical vulnerability, 5 high vulnerabilities, and 5 medium vulnerabilities.
0 / 10	Dependents	0 dependents found
4 / 10	Static Code Analysis	6 existing vulnerabilities identified, 2 critical

Table 3.9: Connect Kit - C1 and C2 Metrics

C3. We utilized metrics and scoring from the scorecards. As Figure 3.10 shows, this package has maintenance support from 2 different companies. This provides confidence that the package is maintained because it is in the company's best interests. Additionally, the package appears to be well maintained. In the last 90 days, there have been 16 commits.

Score	Name	Reason
6 / 10	Contributors	2 companies found
10 / 10	Maintained	16 commit(s) and 0 issue activity found by maintainers in the last 90 days

Table 3.10: Connect Kit - C3 Metrics

C4. We utilized the branch protection and dangerous workflow metrics from the scorecards in this control. Unfortunately, the scorecards do not provide a way to check for signed commits or packages outside of GitHub. This project publishes packages to NPM. Our findings for C4 are in Table 3.11. We identified that the project prevents deletions from the main branch but there is not protection on the release branches. We also identified that the package does not require maintainers to use signed commits, and the packages are not signed.

Score	Name	Reason
1 / 10	Branch-Protection	Deletion protection enabled on ‘main‘ branch
10 / 10	Dangerous-Workflow	No dangerous workflows
0 / 10	Signed-Commits	Not used
0 / 10	Signed-Packages	False

Table 3.11: Connect Kit - C4 Metrics

C5. We had already forked the repository from GitHub for our analysis. We were able to build the project running ‘npm install“. We can ingest new changes from the

upstream project and validate them using a code review before ingesting and publishing the new package.

C6. This control would not have identified the malicious code because the application did not reach outside of the project to download malicious code or make malicious calls when stealing funds.

3.4.3.2 Evaluation

Our methods identified multiple vulnerabilities during Day 0. We also uncovered numerous weak links during Day 1 and Day 2 observations.

Day 0. While performing controls 1-4, we identified that the package had 19 vulnerabilities total with 3 critical and 5 high. The package is actively maintained and the community has two active companies contributing to the code but they don't require signed packages or implement robust branch-protection.

Day 1. Executing C5 and bringing the package under internal control saved the organization. The organization could conduct a code review of the malicious code before incorporating the changes. If the organization had overlooked the malicious code, merged, and moved to Day 2 operations, the additional controls would not caught it.

Day 2. C6 would not have identified the malicious package.

Overall, the Day 0 control would have provided the most protection in this use case. Comparing our framework to OpenSSF's guide we can see in Table 3.12 we scored the package as high risk as did the Scorecards.

Operations	Our Framework	Scorecards
Day 0	High Risk	High Risk
Day 1	Ineffective	N/A
Day 2	Ineffective	N/A

Table 3.12: Connect Kit, TTR: Hours - Framework Comparison

3.4.4 Case Study: CTX - PyPi

CTX is a Python package that extends the behavior of a Python dictionary. For example, *'ctx.a'* is identical to *'ctx["a"]'* and *'ctx.a = 5'* is identical to *'ctx["a"] = 5'* [72]. Prior to the package being poisoned the last update was over eight years ago. The attacks built in malicious code that would scan the developer's environment and exfiltrate the environment variables to a Heroku application. The malicious actor was able to gain access to the project by registering an expired domain name from the original maintainer. This allowed the attacker to access the maintainers accounts and publish the malicious code. A community member of PyPi identified the issue within 3 days and notified the community [73].

3.4.4.1 Control Execution

We executed the six controls using the CTX project. Below are our findings.

C1. CTX does not utilize third-party dependencies.

C2. We leveraged semgrep and gemnasium Python Dependency scanning tools to perform static code analysis. As Table 3.13 shows, we identified 0 vulnerabilities. This a good sign that the project and the project's community in healthy and responsive to fixing vulnerabilities.

Score	Name	Reason
10 / 10	Dependencies	0 dependencies found
3 / 10	Dependents	86 dependents found
10 / 10	Static Code Analysis	0 existing vulnerabilities identified

Table 3.13: CTX - C1 and C2 Metrics

C3. We utilized metrics and scoring from the scorecards. As Figure 3.14 shows, this package has maintenance support from 0 different companies. We also did not detect any activity in project issues by collaborators, members, or project owners in the last 90 days, but 3 new issues opened in that time. This project has not been updated in almost 10 years.

Score	Name	Reason
0 / 10	Contributors	0 companies found
0 / 10	Maintained	0 commit(s) and 0 issue activity found by maintainers in the last 90 days

Table 3.14: CTX - C3 Metrics

C4. We utilized the branch protection and dangerous workflow metrics from the scorecards in this control. Unfortunately, the scorecards do not provide a way to check for signed commits or packages outside of GitHub and this project publishes packages to PyPI. Our findings for C4 are in Table 3.15. We identified that the project does not include branch protection on the main or release branches. We also identified that the package does not require maintainers to use signed commits, and the packages are not signed.

C5. Building the package took little effort. We had already forked the repository from GitHub into GitLab for our analysis. The package does not depend on outside dependencies, so a simple ‘pip install‘ was sufficient. We can ingest new changes

Score	Name	Reason
0 / 10	Branch-Protection	Protection not enabled
10 / 10	Dangerous-Workflow	No dangerous workflows
0 / 10	Signed-Commits	Not used
0 / 10	Signed-Packages	False

Table 3.15: CTX - C4 Metrics

from the upstream project and validate them using a code review before ingesting and publishing the new package.

C6. This control would have identified the package attempting to contact an unknown URL that it used as an endpoint to push environment variables.

3.4.4.2 Evaluation

Our methods identified multiple weak links with the CTX package. First, our Day 0 evaluation identified that there are not any companies supporting the project and that the project had not been maintined in many years. Our Day 1 observations found that the package was buildable and that we could deploy a version we built in house. Lastly, our Day 2 observations showed that our network control would have identified the package as malicious in a runtime environment.

Day 0. While performing controls 1-4, we identified that the package did not have any known vulnerabilities. However, we identified that the project did not seem to be maintained and that there were not any companies actively supporting the project.

Day 1. Executing C5 and bringing the package under internal control saved the organization. The organization could conduct a code review of the malicious code before incorporating the changes. If the organization had overlooked the malicious

code, merged, and moved to Day 2 operations, the additional controls would have caught it.

Day 2. C6 would probably have identified the erroneous calls to Heroku preventing the malicious code from exfiltrating environment variables from a developer's laptop.

Overall, the package was compromised for a couple of days and our Day 0 and Day 1 controls provided insights that highlight the risk of using this package. The package was not maintained and had not been touched in over 8 years. Additionally, the package did not used signed commits or provide anyway to validate the authenticity of a release. Lastly, the malicious code made external API requests that our network layer control would have identified and blocked, preventing any sensitive data from leaving the development and production environments. Comparing our framework with the OpenSSF guide we see in Table 3.16 that because we take into account the number of dependents a package has we scored the package as medium while the OpenSSF framework scored it was high.

Operations	**Our Framework**	**Scorecards**
Day 0	Medium Risk	High Risk
Day 1	Effective	N/A
Day 2	Effective	N/A

Table 3.16: CTX, TTR: Days - Framework Comparison

3.4.5 Case Study: Node IPC - NPM

According to the NPM project page, "Node IPC is a nodejs module for local and remote Inter Process Communication with full support for Linux, Mac and Windows.

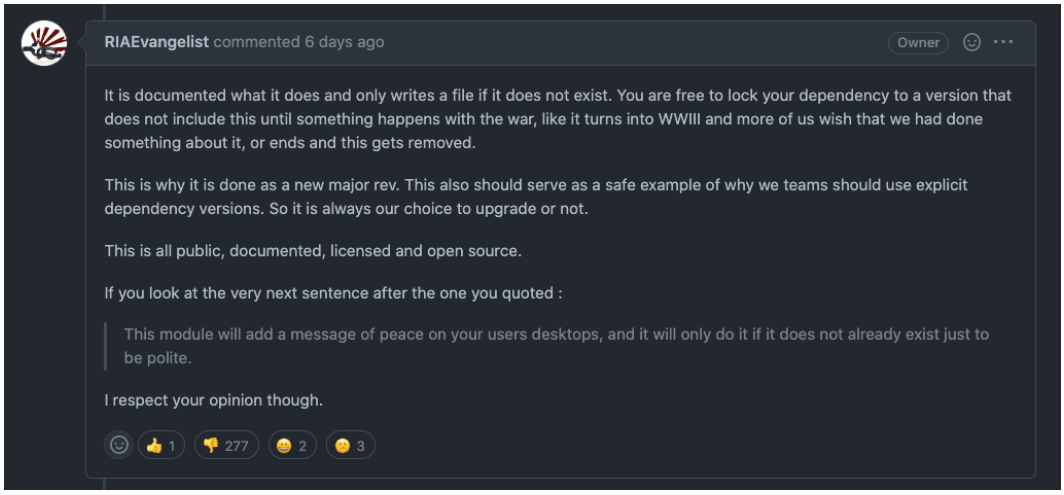


Figure 3.7: Node-IPC Maintainer Response [1]

It also supports all forms of socket communication from low level unix and windows sockets to UDP and secure TLS and TCP sockets" [74]. On March 15, 2022, users of the Vue.js frontend JavaScript framework began experience weird behavior. Node-IPC was a dependency of the Vue.js project and the author of the Node-IPC package had sabotaged the package with another one of his dependencies, Peace Not War. The Peace Not War package is a destructive component that deletes files from computers in specific geographic regions. In this particular instance the code was targeting developers and users in Russia because of the actions of the Russian military in Ukraine, the software is a form of protestware. Prior to users of Vue.js identifying a problem, a community member of the Node-IPC project opened an issue within hours of the malicious code getting committed [1]. In Figure 3.7 the maintainer RIAEvangelist shares his reasoning on why he placed the protestware in his project.

3.4.5.1 Control Execution

C1. Connect Kit has 89 dependencies with 3 high vulnerabilities, and 8 medium vulnerabilities.

C2. We leveraged eslint, semgrep, and nodejs-scan tools to perform static code analysis. As Table 3.17 shows, we identified 3 high and 8 medium vulnerabilities. Although all these findings may not be actionable by a malicious actor, they give the organization insight into potential attack vectors to which the package opens the organization up. At this point, the organization can accept the risk or triage and confirm the findings.

Score	Name	Reason
6 / 10	Dependencies	89 dependencies found with 3 high vulnerabilities, and 8 medium vulnerabilities.
10 / 10	Dependents	860,197 dependents found
7 / 10	Static Code Analysis	3 existing vulnerabilities identified, 0 critical

Table 3.17: Node IPC - C1 and C2 Metrics

C3. We utilized metrics and scoring from the scorecards. As Figure 3.18 shows, this package has maintenance support from 5 different companies. This provides confidence that the package is maintained because it is in the company's best interests. However, the package does not appear to be well maintained. In the last 90 days, there have not been any commits. We also did not detect any activity in project issues by collaborators, members, or project owners in the last 90 days, but 3 new issues opened in that time.

C4. We utilized the branch protection and dangerous workflow metrics from the scorecards in this control. Unfortunately, the scorecards do not provide a way to check

Score	Name	Reason
6 / 10	Contributors	5 companies found
0 / 10	Maintained	0 commit(s) and 0 issue activity found by maintainers in the last 90 days

Table 3.18: Node IPC - C3 Metrics

for signed commits or packages outside of GitHub. This project publishes packages to NPM. Our findings for C4 are in Table 3.19. We identified that the project does not include branch protection on the main or release branches. We also identified that the package does not require maintainers to use signed commits, and the packages are not signed.

Score	Name	Reason
0 / 10	Branch-Protection	No branch protection
10 / 10	Dangerous-Workflow	No dangerous workflows
0 / 10	Signed-Commits	Not used
0 / 10	Signed-Packages	False

Table 3.19: Node IPC - C4 Metrics

C5. Building the package took little effort. We had already forked the repository from GitHub for our analysis. The package does not depend on outside dependencies, so a simple ‘node install‘ was sufficient. We can ingest new changes from the upstream project and validate them using a code review before ingesting and publishing the new package.

C6. This control would not have identified the malicious behavior because the package did not reach out to the internet to download scripts or exfiltrate data.

3.4.5.2 Evaluation

Our methods identified multiple weak links with the Node IPC package. First, our Day 0 evaluation identified that there are 3 high vulnerabilities and 8 medium vulnerabilities in the project that are introduced by the project's dependencies. We also identified 3 vulnerabilities inside of the project using static code analysis. There are 5 companies supporting this project but the project does not appear to be actively maintained.

Day 0. While performing controls 1-4, we identified that the package had a few vulnerabilities and that the project did not appear to be well maintained. The project was an upstream dependency for Vue.js which is a popular Javascript framework. This adds credibility to the package for identifying vulnerabilities.

Day 1. Executing C5 and bringing the package under internal control saved the organization. The organization could conduct a code review of the malicious code before incorporating the changes.

Day 2. C6 would not have identified the malicious code because the code did not go out to the internet to download external scripts or exfiltrate data.

Overall, the Day 0 and Day 1 controls provided insights that highlight the risk of using this package. The package has not been maintained in the last 90 days. The malicious code was added by the project owner, so a company that was adding / building their open-source components in-house should have identified the code prior to incorporating the updated release in their project. Comparing our framework to the Scorecards,

as we see in Table 3.20, we labeled the package low risk while the Scorecards has it at high risk.

Operations	Our Framework	Scorecards
Day 0	Low Risk	High Risk
Day 1	Effective	N/A
Day 2	Ineffective	N/A

Table 3.20: Node IPC, TTR: Hours - Framework Comparison

3.5 Discussion

According to GitHub, today's most popular programming languages are JavaScript, Python, Java, and Ruby. These languages make it easier to reuse code through package managers [75]. In 2005, David Heinemeier Hansson gave a demo via screencast in which he showed developers how to make a blog in 15 minutes using Ruby on Rails [76]. The demo was impressive, and developers were excited about using Ruby on Rails. However, the demo relied on a few dependencies and one package manager. Fast forward 15 years and today, a simple "hello-world" Ruby on Rails application require 966 packages across three separate package managers: Ruby Gems (the default package manager for Ruby), NPM (the default package manager for JavaScript), and Yarn (an additional JavaScript package manager) [77]. Another example of dependency growth is with one of the leading JavaScript frameworks for front-end development, React. Facebook developed and maintained React, which pulls in 1,213 packages. This number does not include the number of dependent packages the dependencies depend on [78]. These packages allow software engineers to start developing faster, enabling

the software to be shipped faster, bringing value more quickly to the customer, but at what cost?

Thousands of software packages are a lot of packages to vet before using open source frameworks. Our life-cycle controls excel at vetting single components. Still, it would probably not be the best for an entire framework because of the cost associated with recursively checking all of the dependencies and those dependencies dependencies. As we noticed in the case studies, the controls on Day 1 and Day 2 provided the most protection, and these are also the most costly controls.

Organizations have for a long time struggled with the build-or-buy decision [79]. Orginization have to make a chouce to build the software they need or buy the software from a third-party vendor. That thought process needs to be applied to open source components now because the components might be free, but they are free as in puppy [80]. So, it is no longer just a question of the cost to maintain the component. It is also the cost of a breach because of the component. It used to be a patch on Friday to prevent a breach on Monday. It was patched on Friday and breached on Monday because of the prevalence of supply chain attacks on open source components.

The perimeter defense from C6 might act as a catch-all in our tabletop exercises and provide protection if an organization cannot spend time checking all of the dependencies. Our controls make a couple of assumptions, with the most significant assumption that the controls are correctly configured, especially for C6. Unfortunately, security misconfigurations happen [81] which makes defense-in-depth so important.

Additionally, as Table 3.21 shows, our Day 0 controls were more likely to accurately predict the time to repair for packages than the OpenSSF's Scorecards when

Project	Our Framework	Scorecards	Time to Repair
UA Parser	Low Risk	Low Risk	Hours
Rest Client	Medium Risk	High Risk	Days
Connect Kit	High Risk	High Risk	Hours
CTX	Medium Risk	High Risk	Days
Node IPC	Low Risk	High Risk	Hours

Table 3.21: Framework Comparison for Time to Repair

comparing the five case studies. 4 out of 5 times our framework scored a package at the appropriate risk level based on the time to repair for the projects. The OpenSSF's Scorecards scored 1 out of 5 projects the appropriate level based on the time to repair of the projects. Both frameworks scored the Connect Kit project as high risk although the community responded within hours of the poisoning. This is probably because Connect Kit is actively used by a company and the impact of the breach resulted in real money being lost.

We understand a framework alone is not enough to protect organizations and that educated software engineers need to understand the context of the risks. This framing is a critical step in protecting any organization according to NIST's Risk Management Process. In the next chapter we share the curriculum we built to frame the problem and the associated risks for software engineers.

CHAPTER IV

UNDERSTANDING AND COMBATING NEXT-GENERATION SUPPLY CHAIN ATTACKS

Software engineers need to understand the risks of incorporating open-source components into projects and many software practitioners are becoming the weakest link in supply chain attacks [51]. Our research focuses on building curriculum that will address this issue by educating our students on the risks, mitigation, and case studies of previous breaches. This curriculum is designed to increase the known knowns and help students understand the known unknowns. We do this with the end goal of better protecting an organization from supply-chain attacks and we were able to deliver the curriculum to students at the University of Colorado Colorado Springs with an 84% effectiveness rate.

4.1 Curriculum

Our curriculum builds off of our previous research highlighted in the last chapter where we shared our verification framework for monitoring open-source components

throughout a project's life. We also utilized many facets of research from the Linux Foundations CHAOSS working group. We split the curriculum in to two parts. The first part focuses on educating students about the foundational risks associated with open-source software. The last part focuses on preventative and active measures organizations can use to protect themselves from attacks once components have been introduced into a project.

4.1.1 Foundational Risks

The first half of our curriculum was built using research that the Linux Foundations CHAOSS working group has released along with Google and the Open Source Security Foundation. The CHAOSS working group has done a great job of categorizing and articulating the broad risks of using open-source components. We built off of these concepts as we developed our curriculum as the foundation as we outlined business risk, code quality, licensing, security, transparency, and dependency risk assessments.

4.1.1.1 Business Risk

The goal of business risk is to understand how active a community is as it provided support to a given software package. There are a couple metrics that we can use to help us understand the health of a project. These metrics consist of the average issue resolution time, bus factor, number of committers, elephant factor, how long issues stay open, how many issues are open, and the number of lines of code in a project.

The average issue resolution time is the time it takes for an issue to be resolved. If issues are resolved quickly this gives us insight that the community is responsive and open to fixing community problems as they arise.

The bus factor measures maintainer activity in a project. It indicates the project's sustainability if active maintainers leave. The more active maintainers the less likely it is that a project will stall if one leaves the project. This metric also strives to understand how work is distributed across maintainers and identifies the ones that are doing the majority of the work.

Typically, the thought has been the more maintainers the better. There is a risk to having many maintainers: the more maintainers a project has the more targets there are to exploit. There is also a risk if there are not enough maintainers, we could see an overloaded maintainer. An overloaded maintainer is someone who manages many projects and can not handle all of the concurrent requests. An attacker might be able to slip malicious code past the maintainer in a pull request.

The number of committers to a project is also a key indicator of project health. Fewer contributors may indicate a project is less inclusive or that the code base is hard to contribute to.

The elephant factor takes in to consideration the distribution of work across companies by examining the number of corporate contributors and the diversity of the corporations.

The length of time an issue has been open and the volume of issues a project has open provide insights into a project's community and may be indicative of the overall health of a project.

Lastly, the number of lines of code that have been contributed could also be helpful in understanding the health of a project. The more lines of code that have been contributed the more likely it is that the project is maintainable and open to contributors.

4.1.1.2 Code Quality

The goal of code quality is to understand the quality of a given software package. A few metrics you can use to measure the code quality of project are things like code complexity, the pull request process, test coverage, and defect resolution time.

Code complexity is a fairly easy metric to gather using static code analysis. The higher the code complexity the more difficult it could be to maintain the project.

Another metric to examine is the pull request process. This is the process that a would-be contributor needs to follow in order to make a contribution back to the project. Does the project have standards that it asks contributors to adhere to? Is there any security scanning tools implemented or static code analysis tools used prior to allowing a pull request? Who needs to review the pull request before it can be merged? These are all questions software engineers should be asking prior to using an open-source component in their projects.

The last metric that CHAOSS recommends looking at is the test coverage of a particular project. This requirement is not unique to CHAOSS as NIST recommends at least 80% code coverage for projects delivering software to the government [82]. Although a project might have high code coverage for testing it does not necessarily mean that it is a well tested project, but it is one indicator of the level of care the

maintainers are putting forth in their projects. An additional indicator of a well tested application would be a mutation score using mutation testing to test the test suite [83].

The last metric for code quality that we recommend looking at is the time it takes for a project to resolve defects in the code once they've been reported. This is not covered by issues or issue resolution time because, although a defect is an issue, it is not the only type of issues. This is separate and needs to be measured separate from issue resolution time as it is critical that projects fix bugs quickly.

4.1.1.3 Licensing

It is imperative that software engineers understand the licenses used in open-source components prior to using the component in their projects. Licenses can have far reaching implications for intellectual property rights and who owns the code and what code must be released back to the community if changes are made. A few metrics that can be used to help understand the licensing implications are license count, coverage, declared licenses, Open Source Initiative approved licenses, and a SPDX document.

The Open Source Initiative (OSI) is a non-profit foundation founded in 1998 that is involved with open-source community building, education, and public advocacy to promote awareness and the importance of non-proprietary software [84].

The license count is important because components and communities might use multiple licenses in different parts of the code base. This also leads to what parts of the code are covered by what license.

OSI licenses are also good to identify because these provide the most open controls for software to use freely used, modified, and shared. A couple of examples of OSI licenses include the Apache 2.0 license and the MIT license.

Lastly, it is good to check if a project contains an SPDX document. This document outlines the licenses used in a project and what components in the code are covered by the different licenses. This consolidated document saves time and is an authoritative resource when deciding to use an open-source component.

4.1.1.4 Security

The goal of security is to understand the security processes and procedures associated with the component's development. There are a couple of key indicators we look for when evaluating the security of an open-source component. The first is to see if the component's community is leveraging the Open Source Security Foundation's (OpenSSF) best practices [85]. We then look to understand the scope and breadth of different languages used in the component.

The Open Source Security Foundation has put together a number of controls to help communities secure their open-source software. The foundation recommends different tooling and source code management controls to better protect open-source communities. If a project is utilizing these best practices it gives more credibility to how serious the community takes security.

4.1.1.5 Transparency

The goal of transparency is to understand how transparent a community is about their component. This is usually accomplished by reviewing the component's Software Bill of Materials (SBOM). An SBOM is a list of all components and libraries along with their versions in a project. According to OpenAI The SBOM can be a very helpful tool for a number of reasons listed below:

Enhanced Security: An SBOM provides a comprehensive list of all components, libraries, and dependencies used in the software. This transparency allows developers and users to quickly identify if any component has known security vulnerabilities. By being aware of these vulnerabilities, project maintainers can promptly update or replace insecure components, reducing the risk of security breaches.

Compliance Management: For projects subject to regulatory standards, an SBOM helps in maintaining compliance. It becomes easier to prove that the software adheres to specific standards or regulations regarding the use of certain types of software or components.

Improved Dependency Management: In open-source projects, dependencies can be numerous and complex. An SBOM offers a clear view of all dependencies, making it easier to manage updates, compatibility issues, and avoid dependency conflicts.

Audit and Review Processes: For large projects with numerous contributors, an SBOM aids in the audit process. It provides a detailed map of the software's composition, making it easier for auditors and reviewers to understand the software's structure and components.

Support for License Compliance: Many open-source projects use components under various licenses. An SBOM helps in tracking these licenses, ensuring that the project complies with the legal requirements of each component's license.

Quality Assurance: By having a detailed understanding of what is in the software, developers can more effectively test and assure the quality of the project. This is especially important in open-source projects where contributions might come from various sources.

Enhanced Collaboration and Trust: An SBOM fosters a culture of transparency and trust among the community. Developers and users can see exactly what they are working with, leading to more informed contributions and usage.

Facilitates Patch Management: When a vulnerability is discovered in a component, an SBOM makes it easier to determine if the affected component is part of the project. This facilitates quick patching and updates.

Eases Integration and Compatibility Checks: In projects that integrate with other systems or software, an SBOM helps in assessing compatibility, making integration processes smoother and more efficient.

Support for Lifecycle Management: An SBOM helps in managing the lifecycle of a project by providing a clear view of all components, their versions, and their update status, enabling better planning for upgrades and maintenance.

Overall, an SBOM in an open-source project enhances security, compliance, and management efficiency, contributing to the robustness, reliability, and trustworthiness of the software [86].

Figure 4.1: Semantic Versioning

4.1.1.6 Version Control

The goal of the version control metric is to understand how and if community is handling version control. A popular version control scheme is semantic versioning. Semantic versioning takes 3 numbers to identify software releases that are major, minor, and patch. Figure 4.1 shows the breakdown of the versioning scheme. This scheme comes in later when we discuss software releases and attack vectors.

4.1.1.7 Package Management

The goal of package management is to understand how a community manages their dependencies. Is there a package manager involved? How does the community specify what version of dependencies to ingest? Does the community leverage continuous integration to pull the latest versions of all the software? How do they ensure compatibility when using a version of a dependency?

4.1.1.8 Software Releases

The goal of software releases is to understand how a component gets released. Is there a process for the release? Are releases signed for authenticity? Does the community leverage continuous delivery mechanisms and pipelines?

4.1.1.9 Dependency Risk Assessment

The goal of dependency risk assessment is to understand the risks of the dependencies the component is using. First, we need to understand what the dependencies are. Then we need to understand the delta between the version of a dependency the component is using and what the latest stable release is.

This goal is important because it could undermine every other goal. If an open-source component is using a dependency that is insecure or poorly coded it would make the entire project insecure. We will want to vet all of our other goals against each dependency that we identify.

Many developers do not realize the number of dependencies used in today's software for simple applications. For example, React uses more than 1,200 dependencies for it's Hello-World application or that Ruby on Rails uses more than 966 dependencies for it's Hello-World application.

4.1.2 Attack Vectors

The second half of our curriculum focuses on attack vectors and mitigation techniques that can be used by organizations to defend themselves against the attack vectors.

Many of the attack vectors we discuss were highlighted by Microsoft engineers that conducted a throughout review of the NPM ecosystem [4]. We have also identified a novel supply chain attack which leverages security best practices to execute.

4.1.2.1 Malicious Package Release

A package can have a malicious release for a variety of reasons. Artifactory shares 5 examples of infection methods [87]:

1. Typosquatting - Typosquatting is the practice of obtaining (or squatting) a famous name with a slight typographical error. This practice applies to many resources like web pages, executable names, and software package names.

   Let's say you buy the domain name gougle.com instead of the legitimate domain google.com, hoping that users will make typing errors and reach the illegitimate domain, used for any attack payload, such as phishing and code injection attacks. Attacks like this happen on the web all the time. The same thing happens now with malicious packages, where attackers register them with names that are similar to popular packages names, but with a small typographical change. Attackers register these malicious packages in popular packages repositories such as NPM and PyPI hoping that developers will occasionally make typing errors and install them.

2. Masquerading - Masquerading is when malware authors, or the attackers, publish a malicious package that impersonates a known package. They duplicate both the code and the metadata of the original project, which they want to impersonate, and

add a small piece of malicious code to this duplicate, essentially building trojan packages.

This infection method is similar to the typosquatting infection method in how it uses a similar name to the legitimate package. Still, the difference is that they aim to deceive developers through similarity to the legitimate package rather than seeking accidental use due to typos.

3. Trojan Package - In the Trojan package infection method, the attacker publishes a fully functional library but hides malicious code in it. Similar to masquerading techniques, malicious code is usually small or obfuscated. Therefore, it's hard to detect and differentiate between legitimate functionality of the package.

4. Dependency Confusion - Dependency confusion exploits a vulnerability in the way that many package managers download dependencies during a build process. In the dependency confusion method, the attacker uses specific package names of internal packages of a target and publishes a malicious package on an external public repository with the exact name. The attacker then assigns a very high version number to this published package. Most default package managers prefer downloading the external malicious package because of its high version number rather than downloading a low version from the legitimate internal repository.

5. Software Package Hijacking - This method involves taking over a legitimate known package and pushing malicious code into it. While this is not an easy task, it's very effective because it can take advantage of the popularity of available packages for a high infection rate.

Software packages hijacking is usually performed by hacking maintainers and developers accounts or by injecting hidden or obfuscated malicious code as part of a legitimate code contribution to an open-source project.

In addition to discussing these 5 techniques that hackers use to exploit software we also discuss how these exploits are executed. Many of these techniques start with an attacker taking over a maintainers' account or passing malicious code in a pull request where maintainers ingest the code. These weak links were explored by the group of researchers from Microsoft [4].

4.1.2.2 Account Takeover

One of the weak links the Microsoft researches identified was that maintainers were using expired domains for their email accounts. This opens the door for attackers to register the expired domain and gain access to the emails of the maintainer. Once the attacker owns the maintainer's email account the attacker can reset the maintainer's password on GitHub. This would give the attacker full account access to the maintainer's projects.

4.1.2.3 Ownership Transfer

Once an account has been taken over, in addition to injecting malicious code, an attacker can also transfer the project out from under the maintainer's account. This would allow the attacker to have persistent control of the package allowing them to release malicious packages at will.

4.1.2.4 Remote Execution

A significant weak link the Microsoft researchers identified was one where components had installation scripts. These scripts allow malicious actors to execute malicious code on a developer's computer in the development, test, or production environments. These scripts present a unique threat to organizations because the code the scripts execute are generally not in the source code repositories. These scripts go to the internet to download the malicious code. As we demonstrated in Chapter 4 with our case studies, these attacks have already been used against open-source components.

4.1.3 Mitigating Attacks

In order to mitigate these attacks developers and engineering managers need to be aware of the risks. They also need to take a holistic approach to vetting open-source components. In Chapter 4 we introduced our framework and in our curriculum we taught our students about it.

1. Check for Known Vulnerabilities: Use tools like Dependabot or Snyk to query the CVE database for known vulnerabilities in the package's dependencies and the package itself.
2. Perform Static Code Analysis: Leverage CWE information to analyze the source code for known weaknesses.
3. Evaluate the Package's Community: Assess the health of the project's community by looking at the number of companies maintaining the project and recent activity within the last 90 days.

4. Assess Package Hygiene: Check for dangerous workflows, signed commits, signed packages, and branch protection.
5. Policy for Trusted Source Builds: Ensure that no open-source artifact is included unless it is built by a trusted source.
6. Implement Network Perimeter Defense: Monitor the activity of the software as it passes data over the network, looking for unexpected changes in behavior or attempts to communicate with unknown sources.

Shifting security left means to understand and implement security considerations earlier in the development process of software. Traditionally, security is one of the last steps prior to deploying a software release. This becomes problematic if there are critical security findings and can hold up a software release. The goal of shifting security to the left is for developers and security professionals to work together earlier in the project's timeline so that security can be baked into a product.

Many of the mitigation techniques can be implemented in an automated fashion. Organizations can use continuous integration pipelines and leverage tooling to accomplish many of the mitigation aspects. The pipeline can run automated checks for known vulnerabilities, conduct static code analysis, and regularly update dependencies.

Education is also a large part of shifting security left. Software engineering managers and software engineers need to understand the risks of open-source components earlier. Regular training sessions have been shown to be beneficial in the information security field.

4.2 Curriculum Delivery

We taught our curriculum at the University of Colorado Colorado Springs during the fall semester of 2023 in CS 4300 / CS 5300 Advanced Software Engineering course. We taught the curriculum to 39 students and we began by asking the students to take a 10 question multiple choice test so that we could gauge their understanding of the topics prior to delivering the curriculum. We then delivered the curriculum over the span of 4 classes using a slide presentation and group exercise. On our last class with the students we re-evaluated the students using the same questions from the pre-test.

We divided the class in to five groups. The groups were given the following prompt:

Through the grapevine you learn that your competition has been hacked. You know they follow CI/CD practices and that they build / release new software at least twice a day. The team says they did not bring in any new packages and that they haven't touched their package.json file for over 6 months. How could they have been breached? The teams had to explain and present in a five minute presentation how the competitor might have been hacked and brainstorm 3 different mechanisms they could implement to protect their organization from such an attack.

4.3 Evaluation

We used the pre and post tests to understand the level of learning the occurred based on the curriculum we delivered over the 4 class sessions. The tests were comprised of 10 questions, all multiple choice.

4.3.1 Evaluation Exam Questions

The students were instructed not to use any notes or outside material to answer the questions and they were given 20 minutes to answer all of the questions. The students were not able to see the correct answer after taking the test. The pre and post tests contained the following questions and choices:

1. What percentage of software used open-source components?

   (a) 50%

   (b) 85%

   (c) 95%

   (d) 100%

2. What update can occur when a caretîs next to a package version in a package.json file?

   (a) Major

   (b) Patch

   (c) Minor

(d) All of the above

3. NPM is a form of what?

(a) Disease

(b) Package Manager

(c) Programming Language

(d) Large Language Model

4. What is the practice of integrating code changes from multiple contributors into a single software project where automated tests occur before integration?

(a) CD

(b) CI

(c) Merge

(d) Push

5. An attacker can hack an open-source project using what?

(a) Localhost

(b) Merge Conflict

(c) GitHub Issues

(d) Expired Domain

6. Which of the following is NOT an example of a Code Quality check?

(a) Software Bill of Materials

(b) Code Complexity

(c) Test Coverage

(d) Defect Resolution Time

7. The middle number in Semantic Versioning represents what version?

(a) Major

(b) Minor

(c) Patch

(d) Release

8. What is the Bus Factor with regards to business risk?

(a) How long is an issue open for before it is closed

(b) What is the volume of open issues

(c) What is the distribution of work in the community

(d) How high is the risk to a project should the most active people leave

9. What is the Elephant Factor with regards to business risk?

(a) How long an issue has been open

(b) How many lines of code a project has

(c) Distribution of work across a community

(d) How long it takes for an issue to be resolved

10. How my packages are required to make a simple ‘hello-world’ React application?

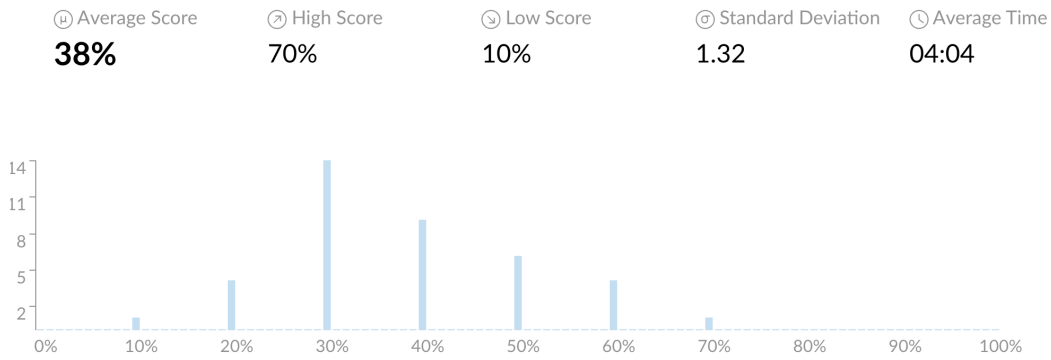


Figure 4.2: Pre-Test Quiz Summary

(a) 1500

(b) 1200

(c) 900

(d) 650

4.3.2 Pre-Test

The students scored an average of 38% during the pre-test. The highest score was a 70% and the lowest score was 10%. The students took an average of 4 minutes to take the exam. Figure 4.2 illustrates the test summary.

4.3.2.1 Pre-Test Question Breakdown

Table 4.1 shows the percentage of questions answered correctly during the Pre-Test. The most missed question was question 7 while question 3 was the most correctly answered question. This told us we needed to focus our curriculum on external frameworks and tools that developers could use to evaluate open-source components. It also

showed us that this group of students seemed to have a good understanding of package managers and what they're used for.

Question	Question Text	Correctly Answered
1	What percentage of software used open-source components?	33%
2	What update can occur when a caret is next to a package version in a package.json file?	5%
3	NPM is a form of what?	95%
4	What is the practice of integrating code changes from multiple contributors into a single software project where automated tests occur before integration?	56%
5	An attacker can hack an open-source project using what?	23%
6	The middle number in Semantic Versioning represents what version?	59%
7	What is the Elephant Factor with regards to business risk?	0%
8	What is the Bus Factor with regards to business risk?	26%
9	Which of the following is NOT an example of a Code Quality check?	56%
10	How many packages are required to make a simple 'hello-world' React application?	26%

Table 4.1: Pre-Test Question Breakdown

4.3.3 Post-Test

The students scored an average of 70% during the post-test. The highest score was a 100% and the lowest score was 30%. The students took an average of 2 minutes 57 seconds to take the exam. Figure 4.3 illustrates the test summary.

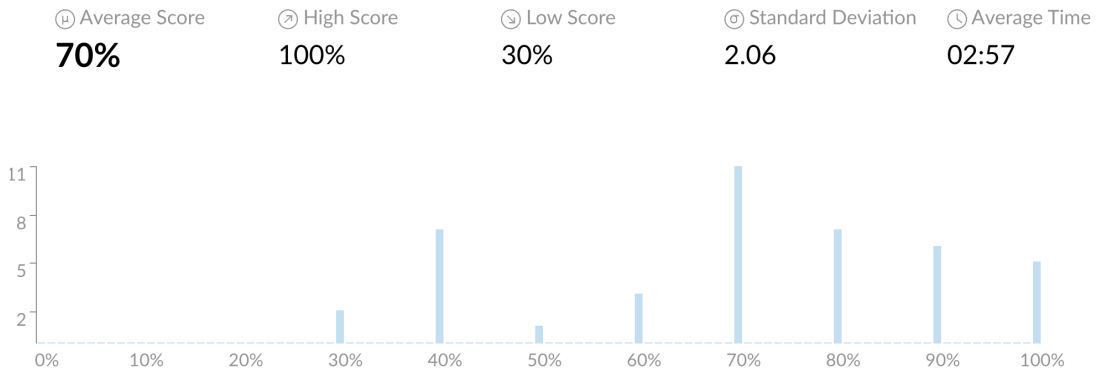


Figure 4.3: Post-Test Quiz Summary

4.3.3.1 Post-Test Question Breakdown

Table 4.1 shows the percentage of questions answered correctly during the Post-Test. The students improved on almost all test questions when comparing the post-test with the pre-test. The average increased 32% and the students were able to complete the post-test 25% faster than the pre-test. The per question average rose by 29.6% after receiving the curriculum.

4.4 Discussion

The curriculum successfully addressed the initial knowledge gaps identified in the pre-test. This strengthened the students' foundational knowledge and understanding of the risks associated with open-source components with an 84% effectiveness rate. Figure 4.4 show the formula was used to compute the effectiveness rate [88]. The students' average increased 32%, the per question average rose by 29.6%, and the students were able to finish the post-test 25% faster.

Question	Question Text	Correctly Answered
1	What percentage of software used open-source components?	71% (+38%)
2	What update can occur when a caret is next to a package version in a package.json file?	52% (+19%)
3	NPM is a form of what?	100% (+5%)
4	What is the practice of integrating code changes from multiple contributors into a single software project where automated tests occur before integration?	76% (+20%)
5	An attacker can hack an open-source project using what?	86% (+63%)
6	The middle number in Semantic Versioning represents what version?	98% (+42%)
7	What is the Elephant Factor with regards to business risk?	36% (+36%)
8	What is the Bus Factor with regards to business risk?	60% (+34%)
9	Which of the following is NOT an example of a Code Quality check?	52% (-4%)
10	How many packages are required to make a simple ‘hello-world‘ React application?	69% (+43%)

Table 4.2: Post-Test Question Breakdown

$$Ei = effectivness$$

$$Ei = \left(\frac{\text{Average Post-Test Score} - \text{Average Pre-Test Score}}{\text{Average Pre-Test Score}} \right) \times 100$$

Figure 4.4: Effectiveness Formula

In this chapter we shared our curriculum to increase the known knowns for software engineers so that they can better safeguard their organizations from emerging vulnerabilities in their software supply chain. In the next chapter we share a new learning platform we developed to enhance the learning experience among our students and remove barriers to our curriculum.

CHAPTER V

AN INCLUSIVE PLATFORM FOR SECURE SOFTWARE ENGINEERING EDUCATION

While teaching our curriculum, highlighted in the last chapter, we realized we needed to remove barriers for students. Many students only had access to school-provided Chromebooks that lacked the necessary resources to run modern programming languages such as Ruby on Rails. We developed a user-friendly platform for teachers and students and implemented the platform on preexisting hardware already owned by the University of Colorado Colorado Springs. The platform has been used for over three years and four different sections of CS 3300, Software Engineering, and CS 4300 / 5300 Advanced Software Engineering.

In addition to resources, a significant challenge with using a modern programming language in education is working with dependencies. This is especially true when executing applied approaches by asking students to work on group projects. All students in a group need to use identical software versions, which can be tricky depending on their operating systems.

```
https://editor.thasting-13-advses2023-1.uccs.devedu.io
```

Figure 5.1: Example URL

Different operating systems provide different software versions depending on how the students install the software. For example, students running Microsoft Windows 10 and downloading the software from the Windows Application Store will get a different version of Ruby than those running Ubuntu Linux 22.04 and install using the APT package manager. This becomes a problem when the newer versions of Ruby are no longer compatible with previous versions. Students run into unexpected problems with versioning, which detracts from the course's educational objectives.

We solved this problem and increased inclusiveness in software engineering education by developing a novel platform for educators and students. This platform increases accessibility, brings equity in learning opportunities, and reinforces technological inclusivity for all while focusing on education objectives.

5.1 Accessibility

Students and teachers access the platform from the web. Teachers can create courses, and each course can have multiple environments. Each environment has one container or more containers containing all the tools required for that particular environment. Once an environment has been defined, teachers can create student accounts for each environment as containers. The student accounts consist of a username and password. After a student account has been created, a unique URL consisting of the course identifier, environment identifier, and the student's username is generated. Fig-

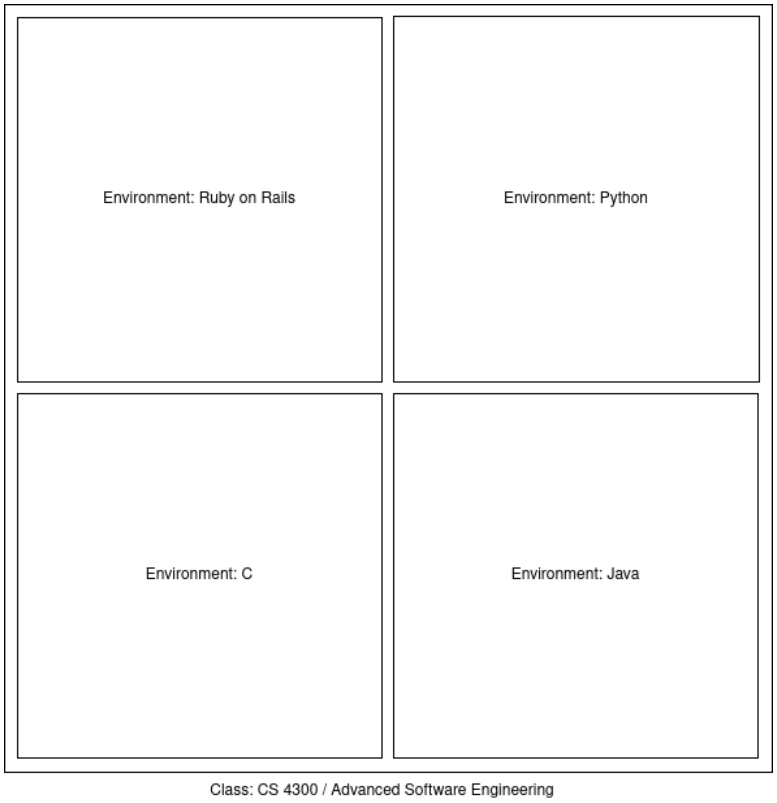


Figure 5.2: A class with Multiple Environments

ure 5.1 shows an example URL that is sent to each student. This URL is then given to each student, and it is where they can access a web-based IDE based on Microsoft's VSCode application.

Figure 5.2 shows an example of what an Advanced Software Engineering course might look like where they allow the students to program in different environments like Ruby on Rails, Python, C, and Java.

The student environment is run on a computing platform with Kubernetes as the orchestrator. We deployed the platform to two universities. First, we deployed to the University of Colorado Colorado Springs using their existing infrastructure. Then we deployed the platform for the United States Air Force Academy using cloud-based

```
1  apiVersion: v1
2  kind: PersistentVolumeClaim
3  metadata:
4    name: student-env-pv-claim
5    labels:
6      app: student-env
7  spec:
8    storageClassName: openebs-hostpath
9    accessModes:
10     - ReadWriteOnce
11   resources:
12     requests:
13       storage: 250Mi
```

Figure 5.3: Kustomize Template for Storage Guardrails

servers. We developed the platform on x86 and ARM64 architecture to provide the most comprehensive and most cost-effective approach. The platform can use any hardware as long as it has access to the internet. This means institutions can leverage the application on hardware they already own, or they can rent space in the Cloud.

Each of the student environment deployments has guardrails in place to prevent students from starving other environments on the host. These guardrails include protections for storage space, RAM utilization, and CPU cycles. These guardrails are in place to protect the other environments running on a given host. In Figure 5.4, we define the min and max memory and CPU usage per environment. In Figure 5.3, we specify the maximum storage each environment is allocated on line 13 when making the persistent volume claim.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: student-env
  labels:
    app: student-env
spec:
  selector:
    matchLabels:
      app: student-env
      tier: frontend
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: student-env
        tier: frontend
    spec:
      containers:
      - image: tghastings/code-esaas
        name: student-env
        resources:
          requests:
            memory: "64Mi"
            cpu: "250m"
          limits:
            memory: "256Mi"
            cpu: "500m"
        ports:
        - containerPort: 80
          name: web
        - containerPort: 3000
          name: rails
        volumeMounts:
        - name: student-env-persistent-storage
          mountPath: /root
      volumes:
      - name: student-env-persistent-storage
        persistentVolumeClaim:
          claimName: student-env-pv-claim
```

Figure 5.4: Kustomize Template for RAM and CPU Guardrails

5.2 Equity in Learning Opportunities

The only prerequisite for students to use the platform is having access to a computer that is capable of running Google Chrome. We utilize VSCode in the browser, which is built and optimized for Chrome. This opens the aperture for students who cannot afford expensive computers or who only have access to a computer in a public space like the library.

Students no longer need to worry about upgrading to the newest releases, and teachers no longer need to troubleshoot student environments. Using the platform, teachers can push the latest releases of the software to student environments in real-time.

5.3 Architecture

The platform is built using Kubernetes, Docker, Ruby on Rails, and Go. Three core components comprise the platform. The first component is the container and container orchestrator. The second component is the user interface that the instructors interact with to create classes and student environments. The last component is the middleware the user interface connects to that executes commands to spin up and down environments. Figure 5.5 shows a high level view of the architecture and the relationships between the components.

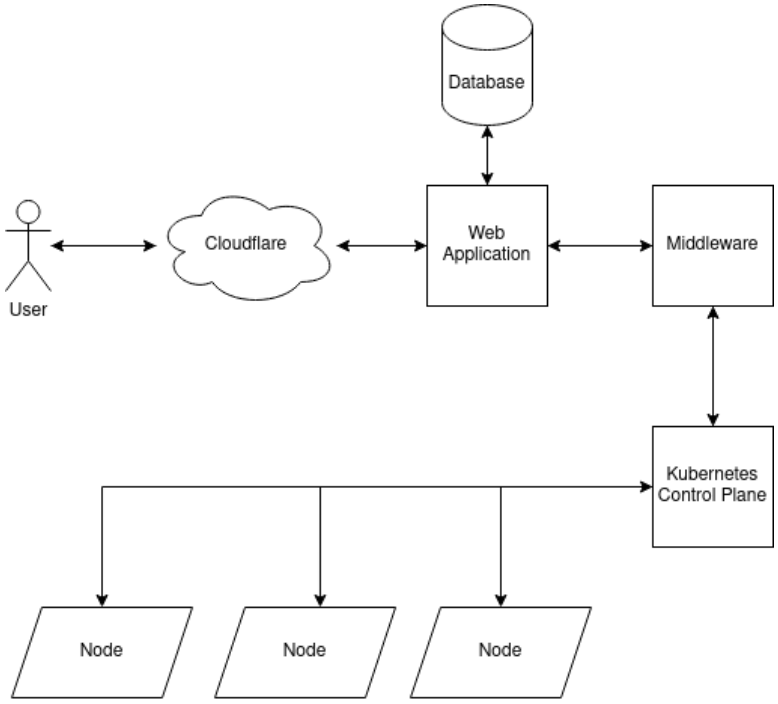


Figure 5.5: Platform Architecture

5.3.1 Containers

Our platform employs Docker to create development environments for students, offering many benefits that enhance their learning experience. Initially, students faced challenges setting up their local environments due to incompatible software versions, lack of computing power, or unsupported operating systems, leading to inconsistency and confusion. With Docker, we have established a uniform and consistent environment for each student, eliminating these issues. Containers provide isolation, ensuring that one student's work does not interfere with another's while allowing us to scale resources efficiently allowing us to be better stewards with our computing capabilities. Docker's choice was strategic, not only for its technical advantages such as version control,

rollback capabilities, and compatibility features, but also due to its widespread industry adoption, preparing students with relevant, marketable skills.

The core of our system is a Docker image built from an Ubuntu base, which includes all necessary tools for Ruby on Rails development. This image is the foundation for all the student containers, ensuring that each student's environment has the exact resources needed for their coursework. In Figure 5.6, we showcase the Dockerfile we use to create the Docker image for the student development environments. In addition to the robust backend, we integrate Coder's web-based Integrated Development Environment (IDE), allowing students to access and work within their environment directly through their web browsers. This combination of a solid, consistent Docker environment with an accessible web interface significantly simplifies the technical overhead for students, enabling them to focus more on learning and less on configuration. To date, our Docker image has been downloaded over 8,400 times and has been forked three times on GitHub [22] [23].

5.3.2 Container Orchestration

We have harnessed the power of Kubernetes for container orchestration to enhance our remote learning platform significantly. Kubernetes has streamlined the deployment of individual student environments across various nodes, which can be located on-premise or hosted on cloud services. This flexibility ensures that each student has a stable and consistent environment, no matter where the resources are or how they are distributed. Moreover, Kubernetes's automation capabilities have been instrumental in scaling these environments. As the demand for student resources grows, Kubernetes

```
FROM ubuntu:jammy
ARG DEBIAN_FRONTEND=noninteractive
ENV SHELL /bin/bash
ENV GIT_EDITOR=nano
RUN apt update \
  && apt install -y zlib1g-dev libsqlite3-dev
   ↪  libffi-dev \
    libpq-dev nodejs ruby-full npm curl supervisor
     ↪  zsh
    git nano \
  && gem install rails -v 7.0.3 \
  && gem install bundler -v 2.0.2 \
  && curl -o- -L https://yarnpkg.com/install.sh |
   ↪  bash \
  && export PATH="$HOME/.yarn/bin:$HOME/
    .config/yarn/global/node_modules/.bin:$PATH" \
  && curl https://cli-assets.heroku.com/install.sh |
   ↪  sh \
  && curl -fsSL https://code-server.dev/install.sh |
   ↪  sh \
  && mkdir -p /var/log/supervisor

ADD supervisord.conf /etc/

EXPOSE 80 3000

ENTRYPOINT ["supervisord", "--nodaemon",
 ↪  "--configuration",
    "/etc/supervisord.conf"]
```

Figure 5.6: Dockerfile for Ruby on Rails Environment

seamlessly scales the number of containers up or down, optimizing resource use and ensuring every student can access the tools they need without delay.

Kubernetes also excels in resource and configuration management. We have been able to efficiently add new nodes to our existing Kubernetes cluster, significantly enhancing our platform's capacity and performance. This means that as more students join or as the computational demands of their projects grow, our system intuitively ex-

pands, maintaining high performance and reliability. Furthermore, Kubernetes has been invaluable in managing sensitive data like student passwords. By utilizing Kubernetes secrets, we have ensured that student environments are secure and that students can safely access their work through a predefined URL without compromising security or convenience.

Lastly, the service discovery feature of Kubernetes has simplified the way students access their environments. Each environment is accessible via a stable and predefined URL, making it easy for students to connect to their workspaces without navigating complicated network configurations or addresses. This user-friendly access is crucial to maintaining an efficient and uninterrupted learning experience.

We used MicroK8s from Canonical running on Ubuntu 22.04 LTS for our Kubernetes solution. We had two nodes: one that hosted the control plane in addition to acting as a node and one that was a worker node. A Kubernetes cluster consists of a set of worker machines, called nodes, that run containerized applications. Every cluster has at least one worker node. The worker node(s) host the Pods, which are the components of the application workload. The control plane manages the worker nodes and the Pods in the cluster. The pods are where the student development environment containers are run. The control plane's components make global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (for example, starting up a new pod when a deployment's replicas field is unsatisfied) [89].

5.3.3 Web Application

The web application is what educators use to interact with the platform. The application provides user authentication and allows multitenancy so that educators from different organizations can use the same application while maintaining the confidentially, integrity, and availability of other educators from different organizations. The web application allows educators to create, view, update, and remove classes.

5.3.3.1 Stack

For our platform, we leveraged Ruby on Rails along with PostgreSQL for the database. Ruby on Rails is a robust server-side web application framework known for its convention over configuration philosophy, ease of use, and strong community support. While building the platform, several factors, including rapid development, scalability, and ease of maintenance, were all considerations.

5.3.3.2 Deployment

Our deployment mechanism for the web application is docker-compose. We chose this method of deployment because the application was built as a monolith and packaged in a Docker image. Docker-compose allows us to rapidly deploy our application along with the PostgreSQL database and Nginx in minutes. The orchestration provides seamless connectivity between the two containers and allows us an easy upgrade path by replacing individual containers. Figure 5.7 shows an example Docker compose file for our application.

We use Nginx as a reverse proxy to serve content to and from the Ruby on Rails application. Nginx also provides the SSL/TLS termination to the Ruby on Rails application. This allows us to avoid using certificates directly in the Ruby on Rails application, which would require us to rebuild and redeploy the application each time certificates change.

5.3.4 Application Programming Interfaces

We used Cloudflare as our domain name system (DNS) for our domain name *devedu.io*. The containers model makes the API calls that interact with Cloudflare and the Middleware.

5.3.4.1 Cloudflare

In the ever-evolving digital landscape, the performance, security, and reliability of web applications are paramount. For our platform, these aspects directly impact the learning experience, accessibility, and trustworthiness of our service. Cloudflare, a leading web performance and security company, offers a range of services designed to optimize and protect websites.

One of the primary advantages of using Cloudflare is its API. We are able to dynamically create and remove DNS entries for each student environment that gets created from our platform via RESTful calls to the Cloudflare service. This automation removed the complexity of manual DNS account creation for each student account.

Cloudflare also provides significant improvement in web application performance. Cloudflare's Content Delivery Network (CDN) distributes the application's content

```yaml
version: '3'
services:
  nginx:
    image: nginx:1.15-alpine
    privileged: true
    ports:
      - "80:80"
      - "8080:8080"
    volumes:
      - ./nginx:/etc/nginx/conf.d
    restart: always
  devedu:
    image: tghastings/devedu:latest
    network_mode: host
    depends_on:
      - "devedu_postgres"
    environment:
      DEVEDU_CLUDFLARE_ZONE: SECRET
      CLOUDFLARE_KEY: SECRET
      CLOUDFLARE_EMAIL: SOMEUSER@UCCS.EDU
      POSTGRES_USER: POSTGRES_USER
      POSTGRES_PASSWORD: SECRET
      POSTGRES_HOST: IP_FOR_POSTGRES_HOST
      POSTGRES_DB: POSTGRES_DB
      SECRET_KEY_BASE: SECRET
      RAILS_SERVE_STATIC_FILES: 'true'
      SENDGRID_API: SECRET
      GO_API_URL: HOST_FOR_GO:8090
      CLOUDFLARE_RECORD: CLOUDFLARE_CLUSTER_RECORD
    restart: unless-stopped
  devedu_postgres:
    image: postgres:12
    ports:
      - "5432:5432"
    volumes:
      - ./postgres:/var/lib/postgresql/data
    environment:
      POSTGRES_DB: POSTGRES_DB
      POSTGRES_USER: POSTGRES_USER
      POSTGRES_PASSWORD: SECRET
```

Figure 5.7: Docker Compose for Web Application

```
slug = self.slug.split('.')
username = slug[0]
org = slug[1]
RestClient.post
 ↪  "https://#{ENV['GO_API_URL']}/deployment",{"title"
 ↪  => self.slug,"password" =>
 ↪  self.password_digest,"containerName" =>
 ↪  env.containerName.downcase}.to_json,{content_type:
 ↪  :json, accept: :json}
```

Figure 5.8: Middleware API Call

```
response = RestClient::Request.new({
  method: :post,
  url:_
  "https://api.cloudflare.com/client/v4/zones/
     #{ENV['DEVEDU_CLUDFLARE_ZONE']}/dns_records",
  payload: { "type": "CNAME", "name":
   ↪  "editor.#{username}.#{org}", "content": org +
   ↪  "." + ENV["CLOUDFLARE_RECORD"], "ttl": 120,
   ↪  "priority": 10, "proxied": false }.to_json,
  headers: { :accept => :json, content_type: :json,
   ↪  'X-Auth-Email' => ENV["CLOUDFLARE_EMAIL"],
   ↪  'X-Auth-Key' => ENV["CLOUDFLARE_KEY"]}
}).execute do |response, request, result|
  case response.code
  when 400
     response.to_str
  when 200
     = JSON.parse(response, object_class: OpenStruct)
     "---------------"
    .update_column(:editor_dns_id,
     ↪  resp.result.id.to_str)
     resp.result.id.to_str
     "---------------"
  else
     "Invalid response #{response.to_str} received."
  end
end
```

Figure 5.9: Cloudflare DNS Entry

across a global network of servers, ensuring that it is closer to users worldwide. This geographical proximity reduces latency, leading to faster load times and a smoother, more responsive user experience. For students and educators accessing the platform, this means quicker access to resources, less waiting time, and a more efficient learning environment.

In addition to its CDN, Cloudflare provides various optimization features like automatic image compression, minification of JavaScript, CSS, and HTML files, and mobile optimization. These features further enhance the speed and responsiveness of the web application, making it more agile and user-friendly.

Security is a critical concern for our platform, and Cloudflare offers robust security features to safeguard against a wide array of threats, including distributed denial of service (DDoS) attacks, malicious bots, and data breaches. By acting as a reverse proxy, Cloudflare filters incoming traffic to the web application, blocking malicious requests and mitigating the risk of attacks. Its DDoS protection is particularly vital, as it ensures that the platform remains available and accessible even under attack, a crucial factor for maintaining uninterrupted educational services.

Cloudflare also provides SSL/TLS encryption, which secures the data transmitted between the user and the web application. This encryption is essential for protecting sensitive information, such as student records, login credentials, and personal data. By ensuring a secure and encrypted connection, Cloudflare helps build trust with users and complies with various privacy and security regulations.

Cloudflare's architecture is designed to enhance the reliability and uptime of web applications. Its vast network acts as a redundant system, ensuring that if one

server goes down, the traffic can be quickly rerouted to another server in the network. This redundancy minimizes downtime and ensures that the educational platform is always available when students and educators need it. Moreover, Cloudflare's automatic failover mechanism provides an extra layer of reliability, further reducing the potential for service interruptions.

Incorporating Cloudflare in front of our platform is a strategic decision that yields numerous benefits. From enhancing performance through its CDN, providing an API that we use to create and edit DNS records, and optimizing features to bolster security against an array of cyber threats, Cloudflare ensures that the platform is fast, secure, and reliable. For educational institutions looking to provide a superior online learning experience, Cloudflare offers a comprehensive solution that meets the critical needs of performance, security, and reliability. By leveraging Cloudflare's services, educational platforms can improve their service quality, safeguard user data, and maintain a continuous, efficient learning environment.

5.3.4.2 Middleware

The middleware is written in Go and interacts with the Kubernetes Control Plane to schedule the deployment of student environments. The web application sends relevant data to the middleware via a Representational State Transfer (REST) call. These calls are what allow the educator to create and delete student environments in the Kubernetes cluster. Figure 5.10 shows the main function of the middleware. The application listens on port 8090 for incoming traffic and routes the traffic based on the path. If a deployment comes in on /deployment, then the deployment function captures the request,

```go
func main() {
  http.HandleFunc("/deployment/", deployment.Route)
  http.HandleFunc("/deployment", deployment.Route)
  // Start the application
  fmt.Println("The application has started.")
  http.ListenAndServe(":8090", nil)
  log.Fatal(http.ListenAndServe(":8090", nil))
}
```

Figure 5.10: main.go

```go
func Route(w http.ResponseWriter, r *http.Request) {
  switch r.Method {
  case http.MethodPost:
          // Create a new record.
          create(w, r)
  case http.MethodDelete:
          // Remove the record.
          delete(w, r)
  default:
          http.Error(w, "Error with post routing",
           ↪  400)
          return
  }
}
```

Figure 5.11: deployment.go Router

parses it, and executes it. Figure 5.11 shows the code for the deployment router. This code matches the request based on the RESTful action, in this case, POST or DELETE.

5.4 Discussion

In this chapter, we discussed the need to remove barriers for students in secure software engineering education. To achieve this goal, we shared about the platform we built and have used across six different courses and sections at the University of Colorado Colorado Springs and at one course at the United States Air Force Academy. Our

approach alleviates many of the barriers for students who do not have access to modern computing resources that are required for software engineering. Our platform also alleviates many of the burdens teachers have when teaching complex topics with real-world applications in group scenarios. The novel platform addresses these challenges by providing a uniform environment that promotes accessibility, equity in learning opportunities, and technological inclusivity, ensuring educational objectives are met without compromise.

CHAPTER VI

CONCLUSION

This thesis develops and evaluates two dynamic and related approaches to secure organizations from software supply chain risks that stem from open-source software using NIST's Risk Management Process.

In our first approach, we created a repeatable framework for organizations to use access, monitor, and respond to open-source component risks through the life-cycle of a project. Our second approach leverages curriculum and an inclusive programming platform for secure software engineering to frame the problem for software engineers. Our evaluation reveals that these two approaches, when combined, are ideal for protecting organizations from software supply chain attacks.

6.1 Conclusion

In our first approach, we acknowledge the indispensable role of open-source components in modern software development and advocate for a multi-faceted, vigilant approach to secure software supply chains against emerging threats. We advocate for a

proactive, informed, and dynamic approach to software security and present a method to manage the life cycle of open-source components. Our framework leverages six repeatable controls capable of protecting organizations that incorporate open-source components by identifying risk during Day 0, Day 1, and Day 2 of the open-source component's operational life. In identifying these six controls, we can help organizations avoid risky components and mitigate the fallout from malicious supply chain attacks, as demonstrated in our case studies.

Our findings identified that our Day 0 controls were more likely to accurately predict the time to repair for packages than the OpenSSF's Scorecards when comparing the five case studies. 4 out of 5 times, our framework scored a package at the appropriate risk level based on the time to repair for the case studies. The OpenSSF's Scorecards scored 1 out of 5 projects the appropriate level based on the time to repair of the projects. Our findings also identified that our Day 1 and Day 2 controls effectively identified malicious behavior in most case studies.

In our second approach, we successfully identified and addressed initial knowledge gaps undergraduate and graduate software engineering students have when understanding and framing the problem of open-source software risks. We used lectures, slides, and a group exercise to deliver the curriculum. We utilized a pre-test to gauge the student's understanding of the material before delivering the curriculum. Then, we use a post-test to gauge the curriculum's effectiveness by measuring student success and comparing the post-test with the pre-test scores. The curriculum provides a scalable and replaceable model for educational interventions in other areas of software engineering and cybersecurity. The approach demonstrated in this study underscores the

importance of proactive and continuous education in the ever-evolving domain of software engineering, particularly in the context of cybersecurity and open-source software management.

Our findings show that the curriculum strengthened the student's foundational knowledge with an 84% effectiveness rate. The student's average increased by 32%, the per-question average rose by 29.6%, and the students were able to finish the post-test 25% faster.

Lastly, we demonstrated a novel approach to remove educational barriers for students as we deliver our curriculum. We developed a web-based development platform that increases accessibility, brings equity in learning opportunities, and reinforces technological inclusivity while focusing on education objectives. The platform has been used to teach seven courses at the University of Colorado Colorado Springs and the United States Air Force Academy across undergraduate and graduate software engineering and advanced software engineering courses. Our environment has been downloaded over 8,400 times on Docker, and the open-source project on GitHub has been forked by three developers [22] [23].

6.2 Limitations and Challenges

When we started this thesis our goal was to identify trust within open-source communities based on specific criteria. We learned that trust is almost impossible to quantify regardless of what metrics are used to measure trust. Additionally, in 2021, Google released their Know, Prevent, Fix framework that overhauled their open-source

Scorecards [49]. The new version of Scorecards incorporated much of what we were planning on building our thesis around but we learned that we could leverage the work Google had already done with the Scorecards and incorporate other work that we were working on with the Linux Foundation and combine the two approaches and make a useful risk management framework for organizations.

When developing the risk framework, the vastness of open-source components made it difficult to monitor all packages across all ecosystems for vulnerabilities or compromises. We did not have scores from the Scorecards for the C3 or C4 metrics from the community when they were compromised. We used metrics as they are today. However, we do not believe this has jeopardized our results because we are looking at the community as it is today, having gone through a breach. The community is probably more prepared and has more substantial community scores today than it did leading up to the breach. We relied on third-party analysis of the breaches in our evaluations. We used the analysis from reputable sources such as GitHub, MITRE, and security vendors.

We also learned through teaching our software engineering courses that many students did not understand or realize the risk of open-source adoption. This lead us to building curriculum based on requirements from the NIST. When delivering the curriculum, we only taught one section of a course. This is a small number of students, and we will discuss more of our thoughts in our future work section.

Lastly, the platform had its own set of challenges. As the development progressed, we realized that our first iteration of the platform would not allow user interfaces to be developed due to technical limitations of running VSCode in the browser. We devised a

way to run two containers in a single pod to accommodate this challenge. This allowed us to launch one Linux container running Ubuntu with the Xfce Windows manager and another container using NoVNC so students could access the full Linux environment from their browser.

6.3 Future Work

There are a couple of key areas where we would like to continue growing our research based on our framework, curriculum, and platform. In the future, we would like to scale the controls from the risk framework so that they can better evaluate open-source components and frameworks with many dependencies. We would also like to implement the controls in an organizational setting with other developers to see how it fits into their workflows. We could use a survey to help us understand the usefulness and efficiency of our framework. Lastly, we would like to look at training an artificial intelligence model on our framework so that we could automate the evaluation of each package. This would also allow us to recursively check dependencies giving us a much wider view of a package.

We would like to teach our curriculum to more students across additional sections and classes at different universities. The curriculum could be built into an entire semester course where students learn about secure software engineering principles and how to handle business risk. We would also like to follow up with our students after they have joined the workforce. We would like to survey them to understand if they found value in the curriculum.

Lastly, we would like to build an open-source community around the platform we developed. The students at the University of Colorado Colorado Springs and at the United States Air Force Academy found value in using the platform. We would like to bring the software to every university that is looking for a way to knock down barrier in their computer science education.

REFERENCES

[1]L. Tal, "Alert: peacenotwar module sabotages npm developers in the node-ipc package to protest the invasion of ukraine," Mar. 2022.

[2]S. J. Vaughan-Nichols, "Github: All open-source developers anywhere are welcome," Oct 2019.

[3]Franklin, "How hackers infiltrate open source projects."

[4]N. e. a. Zahan, "What are weak links in the npm supply chain?," *arXiv:2112.10165 [cs]*, Feb 2022. arXiv: 2112.10165.

[5]M. Souppaya, K. Scarfone, and D. Dodson, "Draft nist special publication 800-218 - secure software development 2 3 framework (ssdf) version 1.1: Recommendations for mitigating the risk of software 5 vulnerabilities," Sep 2021.

[6]Mircosoft, "Github advisory database."

[7]N. I. of Standards and Technology, "Nist special publication 800-161 revision 1: Cybersecurity supply chain risk management," 2022. Accessed: January 15, 2024.

[8]Executive Office of the President, "Improving the nation's cybersecurity," May 2021. Federal Register, vol. 86, no. 93.

[9]Sonatype, "State of the 2020 software supply chain - the 6th annual report on global open source software development."

[10]K. Lewandowski, "Security scorecards for open source projects," Nov 2020.

[11]"Malicious code found in npm package event-stream downloaded 8 million times in the past 2.5 months," Jul 2020.

[12]MITRE, "cve-website."

[13]M. Hanley, “Github’s commitment to npm ecosystem security,” Nov 2021.

[14]C. Cimpanu, “Hacking 20 high-profile dev accounts could compromise half of the npm ecosystem.”

[15]MITRE, “About the cve program.”

[16]MITRE, “Cwe - about - cwe overview.”

[17]S. a. Rose, *Zero Trust Architecture*. No. NIST Special Publication (SP) 800-207, Aug 2020.

[18]C. C. Editor, “defense-in-depth - glossary — csrc.”

[19]W. Ubanski, “Day 0/day 1/day 2 operations meaning - software lifecycle in the cloud age,” Nov. 2021.

[20]plmrry, “vendorize,” Jun 2019.

[21]T. Hastings and K. R. Walcott, “Continuous verification of open source components in a world of weak links,” in *2022 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pp. 201–207, 2022.

[22]T. Hastings, “Docker,” Jan. 2024.

[23]T. Hastings, “code-esaas.” `https://github.com/tghastings/code-esaas`, 2023.

[24]D. Keller, “Will low code developers replace traditional developers? — five,” Feb. 2023.

[25]A. Mockus, “Large-scale code reuse in open source software,” p. 7, IEEE Computer Society, May 2007.

[26]telliott27, “The state of the octoverse: top programming languages of 2018,” Nov 2018.

[27]S. Haefliger, G. von Krogh, and S. Spaeth, "Code reuse in open source software," *Management Science*, vol. 54, p. 180–193, Nov 2007.

[28]J. Katz, "A brief history of package management."

[29]C. Bogart, C. Kästner, and J. Herbsleb, "When it breaks, it breaks: How ecosystem developers reason about the stability of dependencies," in *Automated Software Engineering Workshop (ASEW), 2015 30th IEEE/ACM International Conference on*, pp. 86–89, IEEE, 2015.

[30]R. Abdalkareem, O. Nourry, S. Wehaibi, S. Mujahid, and E. Shihab, "Why do developers use trivial packages? an empirical case study on npm," p. 385–395, ACM, Aug 2017.

[31]S. Wang and M. A. Capretz, "A dependency impact analysis model for web services evolution," in *2009 IEEE International Conference on Web Services*, pp. 359–365, 2009.

[32]J. Colazo and Y. Fang, "Impact of license choice on open source software development activity," *Journal of the American Society for Information Science and Technology*, vol. 60, p. 997–1011, Feb 2009.

[33]CHAOSS Community, "About chaoss." `https://chaoss.community/about-chaoss/`, 2023. Accessed: 2024-01-15.

[34]CHAOSS, Jul 2018.

[35]H. Min Khoo and D. Robey, "Deciding to upgrade packaged software: a comparative case study of motives, contingencies and dependencies," *European Journal of Information Systems*, vol. 16, p. 555–567, Oct 2007.

[36]S. McCamant, M. D. Ernst, S. McCamant, and M. D. Ernst, "Predicting problems caused by component upgrades," *ACM SIGSOFT Software Engineering Notes*, vol. 28, p. 287–296, Sep 2003.

[37]V. H. Nguyen and L. M. S. Tran, "Predicting vulnerable software components with dependency graphs," p. 3, ACM, Sep 2010.

[38]S. Corporation, "Open source security management — sca tool."

[39]A. Sabetta and M. Bezzi, "A practical approach to the automatic classification of security-relevant commits," *arXiv:1807.02458 [cs]*, Jul 2018. arXiv: 1807.02458.

[40]C. Cimpanu, "Somebody tried to hide a backdoor in a popular javascript npm package," May 2018.

[41]A. Decan, T. Mens, and E. Constantinou, "On the impact of security vulnerabilities in the npm package dependency network," in *International Conference on Mining Software Repositories*, 2018.

[42]E. i. C. . M. . a. . Williams, Chris), "How one developer just broke node, babel and thousands of projects in 11 lines of javascript."

[43]D. Haney, "Npm," Mar 2016.

[44]J. Corporation, "Local repositories."

[45]M. Mustonen, "Copyleft—the economics of linux and other open source software," *Information Economics and Policy*, vol. 15, p. 99–121, Mar 2003.

[46]T. A. Alspaugh, H. U. Asuncion, and W. Scacchi, "Analyzing software licenses in open architecture software systems," in *2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development*, p. 54–57, IEEE, May 2009.

[47]D. M. German, Y. Manabe, and K. Inoue, "A sentence-matching method for automatic license identification of source code files," p. 437–446, ACM, Sep 2010.

[48]OpenSSF Best Practices Working Group, "Concise guide for evaluating open source software," 2021. Accessed: 2024-01-15.

[49]S. Vaughan-Nicholas, "Google releases new open-source security software program: Scorecards," July 2021.

[50]Open Source Security Foundation, "Openssf scorecard - security health metrics for open source." `https://github.com/ossf/scorecard`, 2024. Accessed: 2024-01-15.

[51]H. Ortiz, "Why developers are becoming the weakest link in supply chain attacks," Aug. 2022.

[52]CHAOSS Community, "Risk working group repository." `https://github.com/chaoss/wg-risk`, 2024. Accessed: 2024-01-15.

[53]X. Yuan, L. Yang, B. Jones, H. Yu, and B.-T. Chu, "Secure software engineering education: Knowledge area, curriculum and resources," *Journal of Cybersecurity Education, Research and Practice*, vol. 2016, June 2016.

[54]M. A. Talib, A. Khelifi, and L. Jololian, "Secure software engineering: A new teaching perspective based on the swebok," *Interdisciplinary Journal of Information, Knowledge, and Management*, vol. 5, pp. 83–99, 2010.

[55]M. L. Stamat and J. W. Humphries, "Training education: putting secure software engineering back in the classroom," in *Proceedings of the 14th Western Canadian Conference on Computing Education*, WCCCE '09, (New York, NY, USA), p. 116–123, Association for Computing Machinery, 2009.

[56]OpenSSF, “Openssf scorecards,” n.d.

[57]N. Zahan, S. Shohan, D. Harris, and L. Williams, “Do software security practices yield fewer vulnerabilities?,” in *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pp. 292–303, 2023.

[58]MITRE, “Secure code review,” Aug 2013.

[59]RubyGems, “Build software better, together,” May 2022.

[60]R. Tufano, L. Pascarella, M. Tufano, D. Poshyvanyk, and G. Bavota, “Towards automating code review activities,” in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pp. 163–174, 2021.

[61]T. Grance, T. Nolan, K. Burke, R. Dudley, G. White, and T. Good, *Guide to Test, Training, and Exercise Programs for IT Plans and Capabilities*. No. NIST Special Publication (SP) 800-84, Sept. 2006.

[62]Ready.gov, “Exercises — ready.gov,” n.d.

[63]F. Salamn, “ua-parser-js.”

[64]C. Cimpanu, “Malware found in npm package with millions of weekly downloads,” Oct 2021.

[65]F. Salman, “Merge pull request #528 from jparismorgan/oculus,” Oct 2021.

[66]Rest-Client, “Rest-client/rest-client: Simple http and rest client for ruby, inspired by microframework syntax for specifying actions.,” Jun 2019.

[67]J. Koljonen, “[cve-2019-15224] version 1.6.13 published with malicious backdoor. · issue #713 · rest-client/rest-client,” Aug 2019.

[68]A. Brody, "Update rubocop config for rubocop 0.54. rest-client/rest-client@d177784," Mar 2018.

[69]NPM, "connect-kit-loader," Dec. 2023.

[70]L. Franceschi-Bicchierai, "Supply chain attack targeting ledger crypto wallet leaves users hacked," Dec. 2023.

[71]T. Claburn, "Ledger js library poisoned to steal $650k+ from wallets," Dec. 2023.

[72]R. Ledger, "ctx." `https://github.com/figlief/ctx`, 2023.

[73]T. Claburn, "Ctx python package compromised with info-stealing code," May 2022.

[74]NPM, "node-ipc," Mar. 2022.

[75]GitHub. `https://octoverse.github.com/`, 2022.

[76]D. Heinemeier Hansson, "Ruby on rails demo."

[77]T. Hastings, "tghastings/freshror."

[78]T. Hastings, "tghastings/freshreactapp."

[79]V. Cortellessa, F. Marinelli, and P. Potena, "An optimization framework for "build-or-buy" decisions in software architecture," *Computers & Operations Research*, vol. 35, p. 3090–3106, Oct 2008.

[80]B. Cotton, "Free as in puppy: The hidden costs of free software — opensource.com," Feb 2017.

[81]S. Loureiro, "Security misconfigurations and how to prevent them," *Network Security*, vol. 2021, p. 13–16, Nov 2021.

[82]P. Black, B. Guttman, and V. Okun, *Guidelines on Minimum Standards for Developer Verification of Software*. No. NIST Internal or Interagency Report (NISTIR) 8397, Oct. 2021.

[83]P. Delgado-Pérez, I. Medina-Bulo, and M. Núñez, "Using evolutionary mutation testing to improve the quality of test suites," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 596–603, 2017.

[84]O. S. Initiative, Sept. 2006.

[85]O. S. S. Foundation (OpenSSF), "Source code management platform configuration best practices," Aug. 2023.

[86]ChatGPT, OpenAI, "Discussion on the strengths of having a software bill of materials in an open-source project." Personal Communication via OpenAI's ChatGPT, 2023. Accessed on: 13-12-2023.

[87]J. Sar Shalom, "5 examples of malicious package infection methods attackers use," Oct. 2022.

[88]P. Salah, "Measuring student growth: A practical guide to educator evaluation." Online. Accessed: 01/04/2024.

[89]Kubernetes, "Kubernetes components."

Appendix A

SCORECARDS RESULTS

```
### UA Parser ###
→ ~ docker run -e GITHUB_AUTH_TOKEN=******* gcr.io/openssf/scorecard:stable --show-details --
repo=https://github.com/faisalman/ua-parser-js >> ua_parser.txt
Starting [CI-Tests]
Starting [Signed-Releases]
Starting [Pinned-Dependencies]
Starting [Contributors]
Starting [Binary-Artifacts]
Starting [Dependency-Update-Tool]
Starting [Fuzzing]
Starting [Maintained]
Starting [Token-Permissions]
Starting [Packaging]
Starting [Code-Review]
Starting [CII-Best-Practices]
Starting [SAST]
Starting [Branch-Protection]
Starting [Security-Policy]
Starting [License]
Starting [Vulnerabilities]
Starting [Dangerous-Workflow]
Finished [Security-Policy]
Finished [License]
Finished [Vulnerabilities]
Finished [Dangerous-Workflow]
Finished [CI-Tests]
Finished [Signed-Releases]
Finished [Pinned-Dependencies]
Finished [Contributors]
Finished [Binary-Artifacts]
Finished [Dependency-Update-Tool]
Finished [Fuzzing]
Finished [Maintained]
Finished [Token-Permissions]
Finished [Packaging]
Finished [Code-Review]
Finished [CII-Best-Practices]
Finished [SAST]
Finished [Branch-Protection]

RESULTS
-------
→ ~ cat ua_parser.txt
Aggregate score: 8.1 / 10

Check scores:
|---------|-------------------------|---------------------------------|-----------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------|



|  SCORE  |          NAME           |             REASON              |                    DETAILS                    |
DOCUMENTATION/REMEDIATION                                                                                        |
|---------|-------------------------|---------------------------------|-----------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------|
| 10 / 10 | Binary-Artifacts        | no binaries found in the repo   |                                               |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#binary-artifacts       |
|---------|-------------------------|---------------------------------|-----------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------|
| 6 / 10  | Branch-Protection       | branch protection is not        | Info: 'force pushes' disabled                 |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#branch-protection      |
|         |                         | maximal on development and all | on branch 'master' Info:                      |
|
|         |                         | release branches                | 'allow deletion' disabled                     |
|
|         |                         |                                 | on branch 'master' Info: PRs                  |
|
|         |                         |                                 | are required in order to make                 |
|
|         |                         |                                 | changes on branch 'master'                    |
|
|         |                         |                                 | Warn: no status checks found                  |
|
|         |                         |                                 | to merge onto branch 'master'                 |
|
|         |                         |                                 | Warn: number of required                      |
|
|         |                         |                                 | reviewers is 1 on branch                      |
|
|         |                         |                                 | 'master', while the ideal                     |
|
|         |                         |                                 | suggested is 2 Warn: codeowner                |
|
|         |                         |                                 | review is not required on                     |
|
|         |                         |                                 | branch 'master'                               |
|
|---------|-------------------------|---------------------------------|-----------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------|
| 10 / 10 | CI-Tests                | 4 out of 4 merged PRs           |                                               |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#ci-tests              |
|         |                         | checked by a CI test -- score   |                                               |
|
|         |                         | normalized to 10                |                                               |
|
|---------|-------------------------|---------------------------------|-----------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------|
| 5 / 10  | CII-Best-Practices      | badge detected: Passing         |                                               |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#cii-best-practices    |
|---------|-------------------------|---------------------------------|-----------------------------------------------|---
```

```
|---------|-------------------------|---------------------------------|-----------------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------------|
| 0 / 10  | Code-Review             | found 26 unreviewed changesets |                                                       |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#code-review            |
|         |                         | out of 28 -- score normalized  |                                                       |
|
|         |                         | to 0                           |                                                       |
|
|---------|-------------------------|---------------------------------|-----------------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------------|
| 10 / 10 | Contributors            | project has 14 contributing    | Info:                                                 |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#contributors           |
|         |                         | companies or organizations     | WirVsVirus-Hackathon-Corona-Calendar                  |
|
|         |                         |                                | contributor org/company found,                        |
|
|         |                         |                                | clio-lang contributor org/company                     |
|
|         |                         |                                | found, iCasa contributor org/company                  |
|
|         |                         |                                | found, starship contributor                           |
|
|         |                         |                                | org/company found, texas-robocamp                     |
|
|         |                         |                                | contributor org/company found,                        |
|
|         |                         |                                | coderwall-lemmings100 contributor                     |
|
|         |                         |                                | org/company found, coderwall-beaver                   |
|
|         |                         |                                | contributor org/company found,                        |
|
|         |                         |                                | cyberssl contributor org/company                      |
|
|         |                         |                                | found, google contributor                             |
|
|         |                         |                                | org/company found, kivra ab                           |
|
|         |                         |                                | contributor org/company found,                        |
|
|         |                         |                                | SpongePowered contributor                             |
|
|         |                         |                                | org/company found, appmotion                          |
|
|         |                         |                                | contributor org/company found,                        |
|
|         |                         |                                | MetaParticle contributor org/company                  |
|
|         |                         |                                | found, hflabs contributor                             |
|
```

```
|         |                         |                                | org/company found,                                    |
|
|---------|-------------------------|---------------------------------|-----------------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------------|
| 10 / 10 | Dangerous-Workflow      | no dangerous workflow patterns |                                                       |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#dangerous-workflow      |
|         |                         | detected                       |                                                       |
|
|---------|-------------------------|---------------------------------|-----------------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------------|
| 10 / 10 | Dependency-Update-Tool | update tool detected           | Info: tool 'Dependabot' is                            |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#dependency-update-tool |
|         |                         |                                | used: :0                                              |
|
|---------|-------------------------|---------------------------------|-----------------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------------|
| 10 / 10 | Fuzzing                 | project is fuzzed              | Info: OSSFuzz integration                             |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#fuzzing                 |
|         |                         |                                | found                                                 |
|
|---------|-------------------------|---------------------------------|-----------------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------------|
| 10 / 10 | License                 | license file detected          | Info: FSF or OSI recognized                           |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#license                 |
|         |                         |                                | license: LICENSE.md:1 Info:                           |
|
|         |                         |                                | License file found in expected                        |
|
|         |                         |                                | location: LICENSE.md:1                                |
|
|---------|-------------------------|---------------------------------|-----------------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------------|
| 10 / 10 | Maintained              | 20 commit(s) and 5 issue       |                                                       |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#maintained              |
|         |                         | activity found in the last 90  |                                                       |
|
|         |                         | days -- score normalized to 10 |                                                       |
|
|---------|-------------------------|---------------------------------|-----------------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------------|
| 10 / 10 | Packaging               | packaging workflow detected    | Info: Project packages its                            |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#packaging               |
|         |                         |                                | releases by way of Github Actions.:                   |
|
|         |                         |                                | .github/workflows/publish-npm-packages.yml:8          |
|
|---------|-------------------------|---------------------------------|-----------------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------------|
| 5 / 10  | Pinned-Dependencies     | dependency not pinned by hash  | Warn: GitHub-owned GitHubAction                       |
```

```
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#pinned-dependencies    |
|         |                       | detected -- score normalized   | not pinned by hash:                                  |
|         |                       | to 5                           | .github/workflows/analysis-codeql.yml:45 Warn:       |
|         |                       |                                | GitHub-owned GitHubAction not pinned by hash:        |
|         |                       |                                | .github/workflows/analysis-codeql.yml:49 Warn:       |
|         |                       |                                | GitHub-owned GitHubAction not pinned by hash:        |
|         |                       |                                | .github/workflows/analysis-codeql.yml:76 Warn:       |
|         |                       |                                | GitHub-owned GitHubAction not pinned by hash:        |
|         |                       |                                | .github/workflows/analysis-dependency.yml:18         |
|         |                       |                                | Warn: GitHub-owned GitHubAction                      |
|         |                       |                                | not pinned by hash:                                  |
|         |                       |                                | .github/workflows/analysis-dependency.yml:20         |
|         |                       |                                | Warn: GitHub-owned GitHubAction                      |
|         |                       |                                | not pinned by hash:                                  |
|         |                       |                                | .github/workflows/publish-github-packages.yml:14     |
|         |                       |                                | Warn: GitHub-owned GitHubAction                      |
|         |                       |                                | not pinned by hash:                                  |
|         |                       |                                | .github/workflows/publish-github-packages.yml:15     |
|         |                       |                                | Warn: GitHub-owned GitHubAction                      |
|         |                       |                                | not pinned by hash:                                  |
|         |                       |                                | .github/workflows/publish-npm-packages.yml:14        |
|         |                       |                                | Warn: GitHub-owned GitHubAction                      |
|         |                       |                                | not pinned by hash:                                  |
|         |                       |                                | .github/workflows/publish-npm-packages.yml:15        |
|         |                       |                                | Warn: GitHub-owned GitHubAction not pinned           |
|
```

```
|         |                       |                                | by hash: .github/workflows/test-ci.yml:15            |
|         |                       |                                | Warn: GitHub-owned GitHubAction not pinned           |
|         |                       |                                | by hash: .github/workflows/test-ci.yml:16            |
|         |                       |                                | Warn: npmCommand not pinned by hash:                 |
|         |                       |                                | .github/workflows/publish-npm-packages.yml:20        |
|         |                       |                                | Info:   3 out of  14 GitHub-owned GitHubAction       |
|         |                       |                                | dependencies pinned Info:   1 out of   1             |
|         |                       |                                | third-party GitHubAction dependencies pinned         |
|         |                       |                                | Info:   3 out of   4 npmCommand dependencies         |
|         |                       |                                | pinned                                               |
|
|---------|-----------------------|--------------------------------|------------------------------------------------------|---
-------------------------------------------------------------------------------------------------------------|
| 10 / 10 | SAST                  | SAST tool is run on all        | Info: SAST tool installed:                           |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#sast                |
|         |                       | commits                        | CodeQL Info: all commits (4)                         |
|         |                       |                                | are checked with a SAST tool                         |
|
|---------|-----------------------|--------------------------------|------------------------------------------------------|---
-------------------------------------------------------------------------------------------------------------|
| 10 / 10 | Security-Policy       | security policy file detected  | Info: security policy file                           |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#security-policy        |
|         |                       |                                | detected: SECURITY.md:1                              |
|         |                       |                                | Info: Found linked content:                          |
|         |                       |                                | SECURITY.md:1 Info: Found                            |
|         |                       |                                | disclosure, vulnerability,                           |
|         |                       |                                | and/or timelines in security                         |
|         |                       |                                | policy: SECURITY.md:1 Info:                          |
|         |                       |                                | Found text in security policy:                       |
|         |                       |                                | SECURITY.md:1                                        |
|
|---------|-----------------------|--------------------------------|------------------------------------------------------|---
```

```
|---------|-------------------------|---------------------------------|-----------------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------------|
| ?       | Signed-Releases         | no releases found               |                                                     |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#signed-releases        |
|---------|-------------------------|---------------------------------|-----------------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------------|
| 9 / 10  | Token-Permissions       | detected GitHub workflow        | Info: topLevel permissions set to 'read-all':       |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#token-permissions      |
|         |                         | tokens with excessive           | .github/workflows/analysis-codeql.yml:23 Info:      |
|         |                         | permissions                     | jobLevel 'contents' permission set to 'read':       |
|         |                         |                                 | .github/workflows/analysis-codeql.yml:31 Info:      |
|         |                         |                                 | jobLevel 'actions' permission set to 'read':        |
|         |                         |                                 | .github/workflows/analysis-codeql.yml:30 Info:      |
|         |                         |                                 | topLevel 'contents' permission set to 'read':       |
|         |                         |                                 | .github/workflows/analysis-dependency.yml:11        |
|         |                         |                                 | Info: topLevel permissions set to 'read-all':       |
|         |                         |                                 | .github/workflows/analysis-scorecard.yml:18         |
|         |                         |                                 | Warn: no topLevel permission defined:               |
|         |                         |                                 | .github/workflows/publish-github-packages.yml:1     |
|         |                         |                                 | Info: jobLevel 'contents'                           |
|         |                         |                                 | permission set to 'read':                           |
|         |                         |                                 | .github/workflows/publish-github-packages.yml:11    |
|         |                         |                                 | Warn: jobLevel 'packages'                           |
|         |                         |                                 | permission set to 'write':                          |
|         |                         |                                 | .github/workflows/publish-github-packages.yml:12    |
|         |                         |                                 | Warn: no topLevel permission defined:               |
|         |                         |                                 | .github/workflows/publish-npm-packages.yml:1        |
|         |                         |                                 | Info: jobLevel 'contents'                           |
|         |                         |                                 | permission set to 'read':                           |
|


|         |                         |                                 | .github/workflows/publish-npm-packages.yml:11       |
|         |                         |                                 | Info: topLevel 'contents' permission set to         |
|         |                         |                                 | 'read': .github/workflows/test-ci.yml:6             |
|---------|-------------------------|---------------------------------|-----------------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------------|
| 5 / 10  | Vulnerabilities         | 5 existing vulnerabilities      | Warn: Project is vulnerable                         |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#vulnerabilities       |
|         |                         | detected                        | to: GHSA-gxpj-cx7g-858c Warn:                       |
|         |                         |                                 | Project is vulnerable to:                           |
|         |                         |                                 | GHSA-jchw-25xp-jwwc Warn:                           |
|         |                         |                                 | Project is vulnerable to:                           |
|         |                         |                                 | GHSA-f8q6-p94x-37v3 Warn:                           |
|         |                         |                                 | Project is vulnerable to:                           |
|         |                         |                                 | GHSA-qrpm-p2h7-hrv2 Warn:                           |
|         |                         |                                 | Project is vulnerable to:                           |
|         |                         |                                 | GHSA-c2qf-rxjj-qqgw                                 |
|---------|-------------------------|---------------------------------|-----------------------------------------------------|---
-----------------------------------------------------------------------------------------------------------------------|

### REST CLIENT ###
→ ~ docker run -e GITHUB_AUTH_TOKEN=******* gcr.io/openssf/scorecard:stable --show-details --repo=https://github.com/rest-
client/rest-client >> rest_client.txt
Starting [CI-Tests]
Starting [Security-Policy]
Starting [Code-Review]
Starting [SAST]
Starting [CII-Best-Practices]
Starting [Contributors]
Starting [Token-Permissions]
Starting [Signed-Releases]
Starting [Binary-Artifacts]
Starting [Branch-Protection]
Starting [Pinned-Dependencies]
Starting [Vulnerabilities]
Starting [Maintained]
Starting [Dependency-Update-Tool]
Starting [Fuzzing]
```

```
Starting [License]
Starting [Dangerous-Workflow]
Starting [Packaging]
Finished [Dependency-Update-Tool]
Finished [Fuzzing]
Finished [Vulnerabilities]
Finished [Maintained]
Finished [Packaging]
Finished [License]
Finished [Dangerous-Workflow]
Finished [SAST]
Finished [CII-Best-Practices]
Finished [CI-Tests]
Finished [Security-Policy]
Finished [Code-Review]
Finished [Branch-Protection]
Finished [Pinned-Dependencies]
Finished [Contributors]
Finished [Token-Permissions]
Finished [Signed-Releases]
Finished [Binary-Artifacts]

RESULTS
-------
→ ~ cat rest_client.txt
Aggregate score: 3.1 / 10

Check scores:
|---------|-----------------------|--------------------------------|--------------------------------|----------------------
-------------------------------------------------------------------------------------------|
|  SCORE  |          NAME         |             REASON             |            DETAILS             |
DOCUMENTATION/REMEDIATION                                    |
|---------|-----------------------|--------------------------------|--------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| 10 / 10 | Binary-Artifacts      | no binaries found in the repo  |                                |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#binary-artifacts       |
|---------|-----------------------|--------------------------------|--------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| 0 / 10  | Branch-Protection     | branch protection not enabled  | Warn: branch protection not    |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#branch-protection      |
|         |                       | on development/release         | enabled for branch 'master'    |
|
|         |                       | branches                       |                                |
|
|---------|-----------------------|--------------------------------|--------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| 0 / 10  | CI-Tests              | 0 out of 10 merged PRs         |                                |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#ci-tests               |
|         |                       | checked by a CI test -- score  |                                |




|
|         |                       | normalized to 0                |                                |
|
|---------|-----------------------|--------------------------------|--------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| 0 / 10  | CII-Best-Practices    | no effort to earn an OpenSSF   |                                |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#cii-best-practices     |
|         |                       | best practices badge detected  |                                |
|
|---------|-----------------------|--------------------------------|--------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| 2 / 10  | Code-Review           | found 10 unreviewed changesets |                                |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#code-review            |
|         |                       | out of 14 -- score normalized  |                                |
|
|         |                       | to 2                           |                                |
|
|---------|-----------------------|--------------------------------|--------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| 10 / 10 | Contributors          | project has 23 contributing    | Info: rspec contributor        |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#contributors           |
|         |                       | companies or organizations     | org/company found, shruggers   |
|
|         |                       |                                | contributor org/company        |
|
|         |                       |                                | found, CraftedCode contributor |
|
|         |                       |                                | org/company found, hcs         |
|
|         |                       |                                | contributor org/company        |
|
|         |                       |                                | found, government contributor  |
|
|         |                       |                                | org/company found, worldwide   |
|
|         |                       |                                | fishladder & sons contributor  |
|
|         |                       |                                | org/company found, calendly    |
|
|         |                       |                                | contributor org/company        |
|
|         |                       |                                | found, pat-go contributor      |
|
|         |                       |                                | org/company found, crafted     |
|
|         |                       |                                | code ltd / explorative ltd     |
|
|         |                       |                                | contributor org/company found, |
|
```

```
|         |                        |                               | pkgr contributor org/company   |
|         |                        |                               | found, rest-client contributor |
|         |                        |                               | org/company found, maintainers |
|         |                        |                               | contributor org/company        |
|         |                        |                               | found, Tacktical contributor   |
|         |                        |                               | org/company found, pullpreview |
|         |                        |                               | contributor org/company        |
|         |                        |                               | found, ouestcode contributor   |
|         |                        |                               | org/company found, verily      |
|         |                        |                               | contributor org/company        |
|         |                        |                               | found, sinatra contributor     |
|         |                        |                               | org/company found, systeminit  |
|         |                        |                               | contributor org/company found, |
|         |                        |                               | opf contributor org/company    |
|         |                        |                               | found, atlrug contributor      |
|         |                        |                               | org/company found, theforeman  |
|         |                        |                               | contributor org/company        |
|         |                        |                               | found, solo.io contributor     |
|         |                        |                               | org/company found, sfosc       |
|         |                        |                               | contributor org/company found, |
|         |                        |                               |                                |
|---------|------------------------|-------------------------------|--------------------------------|----------------------
-------------------------------------------------------------------------------------|
| ?       | Dangerous-Workflow     | no workflows found            |                                |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#dangerous-workflow     |
|---------|------------------------|-------------------------------|--------------------------------|----------------------
-------------------------------------------------------------------------------------|
| 0 / 10  | Dependency-Update-Tool | no update tool detected       | Warn: tool 'RenovateBot'       |



https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#dependency-update-tool |
|         |                        |                               | is not used Warn: tool         |
|         |                        |                               | 'Dependabot' is not used Warn: |
|         |                        |                               | tool 'PyUp' is not used        |
|---------|------------------------|-------------------------------|--------------------------------|----------------------
-------------------------------------------------------------------------------------|
| 0 / 10  | Fuzzing                | project is not fuzzed         | Warn: no OSSFuzz integration   |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#fuzzing                |
|         |                        |                               | found Warn: no GoBuiltInFuzzer |
|         |                        |                               | integration found Warn:        |
|         |                        |                               | no PythonAtherisFuzzer         |
|         |                        |                               | integration found Warn:        |
|         |                        |                               | no CLibFuzzer integration      |
|         |                        |                               | found Warn: no CppLibFuzzer    |
|         |                        |                               | integration found Warn: no     |
|         |                        |                               | SwiftLibFuzzer integration     |
|         |                        |                               | found Warn: no RustCargoFuzzer |
|         |                        |                               | integration found Warn: no     |
|         |                        |                               | JavaJazzerFuzzer integration   |
|         |                        |                               | found Warn: no ClusterFuzzLite |
|         |                        |                               | integration found Warn: no     |
|         |                        |                               | HaskellPropertyBasedTesting    |
|         |                        |                               | integration found Warn: no     |
|         |                        |                               | TypeScriptPropertyBasedTesting |
|         |                        |                               | integration found Warn: no     |
|         |                        |                               | JavaScriptPropertyBasedTesting |
|         |                        |                               | integration found              |
|
```

```
|---------|-------------------------|---------------------------------|---------------------------------|----------------------
-----------------------------------------------------------------------------------------------|
| 10 / 10 | License                 | license file detected           | Info: FSF or OSI recognized     |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#license                |
|         |                         |                                 | license: LICENSE:1 Info:        |
|
|         |                         |                                 | License file found in expected  |
|
|         |                         |                                 | location: LICENSE:1             |
|
|---------|-------------------------|---------------------------------|---------------------------------|----------------------
-----------------------------------------------------------------------------------------------|
| 0 / 10  | Maintained              | 0 commit(s) and 0 issue         |                                 |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#maintained              |
|         |                         | activity found in the last 90   |                                 |
|
|         |                         | days -- score normalized to 0   |                                 |
|
|---------|-------------------------|---------------------------------|---------------------------------|----------------------
-----------------------------------------------------------------------------------------------|
| ?       | Packaging               | packaging workflow not          | Warn: no GitHub/GitLab          |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#packaging               |
|         |                         | detected                        | publishing workflow detected.   |
|
|---------|-------------------------|---------------------------------|---------------------------------|----------------------
-----------------------------------------------------------------------------------------------|
| ?       | Pinned-Dependencies     | no dependencies found           |                                 |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#pinned-dependencies    |
|---------|-------------------------|---------------------------------|---------------------------------|----------------------
-----------------------------------------------------------------------------------------------|
| 0 / 10  | SAST                    | SAST tool is not run on all     | Warn: 0 commits out of 26 are   |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#sast                  |
|         |                         | commits -- score normalized to  | checked with a SAST tool        |
|
|         |                         | 0                               |                                 |
|
|---------|-------------------------|---------------------------------|---------------------------------|----------------------
-----------------------------------------------------------------------------------------------|
| 0 / 10  | Security-Policy         | security policy file not        | Warn: no security policy file   |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#security-policy        |
|         |                         | detected                        | detected Warn: no security      |
|
|         |                         |                                 | file to analyze Warn: no        |
|
|         |                         |                                 | security file to analyze Warn:  |
|
|         |                         |                                 | no security file to analyze     |
|
|---------|-------------------------|---------------------------------|---------------------------------|----------------------
```

```
-----------------------------------------------------------------------------------------------|
| ?       | Signed-Releases         | no releases found               |                                 |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#signed-releases        |
|---------|-------------------------|---------------------------------|---------------------------------|----------------------
-----------------------------------------------------------------------------------------------|
| ?       | Token-Permissions       | no tokens found                 |                                 |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#token-permissions      |
|---------|-------------------------|---------------------------------|---------------------------------|----------------------
-----------------------------------------------------------------------------------------------|
| 10 / 10 | Vulnerabilities         | 0 existing vulnerabilities      |                                 |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#vulnerabilities        |
|         |                         | detected                        |                                 |
|
|---------|-------------------------|---------------------------------|---------------------------------|----------------------
-----------------------------------------------------------------------------------------------|
```

```
### Connect Kit ###
→ ~ docker run -e GITHUB_AUTH_TOKEN=****** gcr.io/openssf/scorecard:stable --show-details --
repo=https://github.com/LedgerHQ/connect-kit
Starting [CI-Tests]
Starting [Vulnerabilities]
Starting [Binary-Artifacts]
Starting [SAST]
Starting [Code-Review]
Starting [CII-Best-Practices]
Starting [License]
Starting [Maintained]
Starting [Signed-Releases]
Starting [Dependency-Update-Tool]
Starting [Pinned-Dependencies]
Starting [Token-Permissions]
Starting [Security-Policy]
Starting [Packaging]
Starting [Fuzzing]
Starting [Dangerous-Workflow]
Starting [Branch-Protection]
Starting [Contributors]
Aggregate score: 3.6 / 10

Check scores:
Finished [CII-Best-Practices]
Finished [License]
Finished [Dependency-Update-Tool]
Finished [Pinned-Dependencies]
Finished [Maintained]
Finished [Signed-Releases]
Finished [Branch-Protection]
Finished [Contributors]
Finished [Token-Permissions]
```

```
Finished [Security-Policy]
Finished [Packaging]
Finished [Fuzzing]
Finished [Dangerous-Workflow]
Finished [SAST]
Finished [Code-Review]
Finished [CI-Tests]
Finished [Vulnerabilities]
Finished [Binary-Artifacts]

RESULTS
-------
|---------|------------------------|--------------------------------|-----------------------------------------------
|-----------------------------------------------------------------------------------------------------------|
|  SCORE  |          NAME          |             REASON             |                      DETAILS                      |
DOCUMENTATION/REMEDIATION                                  |
|---------|------------------------|--------------------------------|-----------------------------------------------
|-----------------------------------------------------------------------------------------------------------|
| 10 / 10 | Binary-Artifacts       | no binaries found in the repo  |                                                   |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#binary-artifacts       |
|---------|------------------------|--------------------------------|-----------------------------------------------
|-----------------------------------------------------------------------------------------------------------|
| 1 / 10  | Branch-Protection      | branch protection is not       | Warn: 'force pushes' enabled                      |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#branch-protection      |
|         |                        | maximal on development and all | on branch 'main' Info: 'allow                     |
|
|         |                        | release branches               | deletion' disabled on branch                      |
|
|         |                        |                                | 'main' Warn: PRs are not                          |
|
|         |                        |                                | required to make changes on                       |
|
|         |                        |                                | branch 'main'; or we don't                        |
|
|         |                        |                                | have data to detect it.If you                     |
|
|         |                        |                                | think it might be the latter,                     |
|
|         |                        |                                | make sure to run Scorecard                        |
|
|         |                        |                                | with a PAT or use Repo Rules                      |
|
|         |                        |                                | (that are always public)                          |
|
|         |                        |                                | instead of Branch Protection                      |
|
|         |                        |                                | settings Warn: no status                          |
|
|         |                        |                                | checks found to merge onto                        |



|
|         |                        |                                | branch 'main' Warn: number                        |
|
|         |                        |                                | of required reviewers is 0 on                     |
|
|         |                        |                                | branch 'main', while the ideal                    |
|
|         |                        |                                | suggested is 2 Warn: codeowner                    |
|
|         |                        |                                | review is not required on                         |
|
|         |                        |                                | branch 'main'                                     |
|
|---------|------------------------|--------------------------------|-----------------------------------------------
|-----------------------------------------------------------------------------------------------------------|
| 0 / 10  | CI-Tests               | 0 out of 9 merged PRs          |                                                   |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#ci-tests               |
|         |                        | checked by a CI test -- score  |                                                   |
|
|         |                        | normalized to 0                |                                                   |
|
|---------|------------------------|--------------------------------|-----------------------------------------------
|-----------------------------------------------------------------------------------------------------------|
| 0 / 10  | CII-Best-Practices     | no effort to earn an OpenSSF   |                                                   |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#cii-best-practices     |
|         |                        | best practices badge detected  |                                                   |
|
|---------|------------------------|--------------------------------|-----------------------------------------------
|-----------------------------------------------------------------------------------------------------------|
| 4 / 10  | Code-Review            | found 9 unreviewed changesets  |                                                   |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#code-review            |
|         |                        | out of 17 -- score normalized  |                                                   |
|
|         |                        | to 4                           |                                                   |
|
|---------|------------------------|--------------------------------|-----------------------------------------------
|-----------------------------------------------------------------------------------------------------------|
| 6 / 10  | Contributors           | project has 2 contributing     | Info: ramyeb-learning                             |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#contributors           |
|         |                        | companies or organizations --  | contributor org/company found,                    |
|
|         |                        | score normalized to 6          | ledger contributor org/company                    |
|
|         |                        |                                | found,                                            |
|
|---------|------------------------|--------------------------------|-----------------------------------------------
|-----------------------------------------------------------------------------------------------------------|
| 10 / 10 | Dangerous-Workflow     | no dangerous workflow patterns |                                                   |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#dangerous-workflow     |
```

```
|         |                        | detected                       |                                                    |
|
|---------|------------------------|--------------------------------|----------------------------------------------------
|-----------------------------------------------------------------------------------------------------------------|
| 0 / 10  | Dependency-Update-Tool | no update tool detected        | Warn: tool 'RenovateBot'                           |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#dependency-update-tool |
|         |                        |                                | is not used Warn: tool                             |
|
|         |                        |                                | 'Dependabot' is not used Warn:                     |
|
|         |                        |                                | tool 'PyUp' is not used                            |
|
|---------|------------------------|--------------------------------|----------------------------------------------------
|-----------------------------------------------------------------------------------------------------------------|
| 0 / 10  | Fuzzing                | project is not fuzzed          | Warn: no OSSFuzz integration                       |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#fuzzing               |
|         |                        |                                | found Warn: no GoBuiltInFuzzer                     |
|
|         |                        |                                | integration found Warn:                            |
|
|         |                        |                                | no PythonAtherisFuzzer                             |
|
|         |                        |                                | integration found Warn:                            |
|
|         |                        |                                | no CLibFuzzer integration                          |
|
|         |                        |                                | found Warn: no CppLibFuzzer                        |
|
|         |                        |                                | integration found Warn: no                         |
|
|         |                        |                                | SwiftLibFuzzer integration                         |
|
|         |                        |                                | found Warn: no RustCargoFuzzer                     |
|
|         |                        |                                | integration found Warn: no                         |
|
|         |                        |                                | JavaJazzerFuzzer integration                       |
|
|         |                        |                                | found Warn: no ClusterFuzzLite                     |
|
|         |                        |                                | integration found Warn: no                         |
|
|         |                        |                                | HaskellPropertyBasedTesting                        |
|
|         |                        |                                | integration found Warn: no                         |
|
|         |                        |                                | TypeScriptPropertyBasedTesting                     |
|
|         |                        |                                | integration found Warn: no                         |


|
|         |                        |                                | JavaScriptPropertyBasedTesting                     |
|
|         |                        |                                | integration found                                  |
|
|---------|------------------------|--------------------------------|----------------------------------------------------
|-----------------------------------------------------------------------------------------------------------------|
| 0 / 10  | License                | license file not detected      | Info: project does not have                        |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#license               |
|         |                        |                                | a license file Info: project                       |
|
|         |                        |                                | does not have a license file                       |
|
|---------|------------------------|--------------------------------|----------------------------------------------------
|-----------------------------------------------------------------------------------------------------------------|
| 10 / 10 | Maintained             | 16 commit(s) and 0 issue       |                                                    |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#maintained            |
|         |                        | activity found in the last 90  |                                                    |
|
|         |                        | days -- score normalized to 10 |                                                    |
|
|---------|------------------------|--------------------------------|----------------------------------------------------
|-----------------------------------------------------------------------------------------------------------------|
| ?       | Packaging              | packaging workflow not         | Warn: no GitHub/GitLab                             |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#packaging             |
|         |                        | detected                       | publishing workflow detected.                      |
|
|---------|------------------------|--------------------------------|----------------------------------------------------
|-----------------------------------------------------------------------------------------------------------------|
| 0 / 10  | Pinned-Dependencies    | dependency not pinned by hash  | Warn: GitHub-owned GitHubAction not pinned by      |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#pinned-dependencies   |
|         |                        | detected -- score normalized   | hash: .github/workflows/check_npm_repo.yml:13      |
|
|         |                        | to 0                           | Warn: GitHub-owned GitHubAction not pinned by      |
|
|         |                        |                                | hash: .github/workflows/check_npm_repo.yml:14      |
|
|         |                        |                                | Warn: third-party GitHubAction not pinned by hash: |
|
|         |                        |                                | .github/workflows/check_npm_repo.yml:54 Warn:      |
|
|         |                        |                                | GitHub-owned GitHubAction not pinned by hash:      |
|
|         |                        |                                | .github/workflows/release_connect-kit-loader.yml:19 |
|
|         |                        |                                | Warn: GitHub-owned GitHubAction not pinned by hash: |
|
|         |                        |                                | .github/workflows/release_connect-kit-loader.yml:25 |
|
```

```
|         |                        |                                | Warn: third-party GitHubAction not pinned by hash:  |
|         |                        |                                | .github/workflows/release_connect-kit-loader.yml:50 |
|         |                        |                                | Warn: third-party GitHubAction not pinned by hash:  |
|         |                        |                                | .github/workflows/release_connect-kit-loader.yml:83 |
|         |                        |                                | Warn: third-party GitHubAction not pinned by hash:  |
|         |                        |                                | .github/workflows/release_connect-kit-loader.yml:91 |
|         |                        |                                | Warn: GitHub-owned GitHubAction not pinned by       |
|         |                        |                                | hash: .github/workflows/release_connect-kit.yml:27  |
|         |                        |                                | Warn: GitHub-owned GitHubAction not pinned by       |
|         |                        |                                | hash: .github/workflows/release_connect-kit.yml:33  |
|         |                        |                                | Warn: third-party GitHubAction not pinned by        |
|         |                        |                                | hash: .github/workflows/release_connect-kit.yml:59  |
|         |                        |                                | Warn: third-party GitHubAction not pinned by        |
|         |                        |                                | hash: .github/workflows/release_connect-kit.yml:92  |
|         |                        |                                | Warn: third-party GitHubAction not pinned by hash:  |
|         |                        |                                | .github/workflows/release_connect-kit.yml:100       |
|         |                        |                                | Warn: npmCommand not pinned by hash:                |
|         |                        |                                | .github/workflows/check_npm_repo.yml:20 Info:   0   |
|         |                        |                                | out of   6 GitHub-owned GitHubAction dependencies   |
|         |                        |                                | pinned Info:   0 out of   7 third-party             |
|         |                        |                                | GitHubAction dependencies pinned Info:   0 out of   |
|         |                        |                                | 1 npmCommand dependencies pinned                    |
|---------|------------------------|--------------------------------|-----------------------------------------------------
|-------------------------------------------------------------------------------------------------------------|
| 0 / 10  | SAST                   | SAST tool is not run on all    | Warn: 0 commits out of 22 are                       |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#sast                  |
|         |                        | commits -- score normalized to | checked with a SAST tool                            |
```

```
|         |                        | 0                              |                                                     |
|---------|------------------------|--------------------------------|-----------------------------------------------------
|-------------------------------------------------------------------------------------------------------------|
| 0 / 10  | Security-Policy        | security policy file not       | Warn: no security policy file                       |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#security-policy       |
|         |                        | detected                       | detected Warn: no security                          |
|         |                        |                                | file to analyze Warn: no                            |
|         |                        |                                | security file to analyze Warn:                      |
|         |                        |                                | no security file to analyze                         |
|---------|------------------------|--------------------------------|-----------------------------------------------------
|-------------------------------------------------------------------------------------------------------------|
| ?       | Signed-Releases        | no releases found              |                                                     |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#signed-releases       |
|---------|------------------------|--------------------------------|-----------------------------------------------------
|-------------------------------------------------------------------------------------------------------------|
| 0 / 10  | Token-Permissions      | detected GitHub workflow       | Warn: no topLevel permission defined:               |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#token-permissions     |
|         |                        | tokens with excessive          | .github/workflows/check_npm_repo.yml:1              |
|         |                        | permissions                    | Warn: no topLevel permission defined:               |
|         |                        |                                | .github/workflows/release_connect-kit-loader.yml:1  |
|         |                        |                                | Warn: no topLevel permission defined:               |
|         |                        |                                | .github/workflows/release_connect-kit.yml:1 Info:   |
|         |                        |                                | no jobLevel write permissions found                 |
|---------|------------------------|--------------------------------|-----------------------------------------------------
|-------------------------------------------------------------------------------------------------------------|
| 4 / 10  | Vulnerabilities        | 6 existing vulnerabilities     | Warn: Project is vulnerable                         |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#vulnerabilities       |
|         |                        | detected                       | to: GHSA-67hx-6x53-jw92 Warn:                       |
|         |                        |                                | Project is vulnerable to:                           |
|         |                        |                                | GHSA-7fh5-64p2-3v2j Warn:                           |
|         |                        |                                | Project is vulnerable to:                           |
|         |                        |                                | GHSA-m95q-7qp3-xv42 Warn:                           |
```

```
|         |                        |                                | Project is vulnerable to:                                |
|
|         |                        |                                | GHSA-x9w5-v3q2-3rhw Warn:                                |
|
|         |                        |                                | Project is vulnerable to:                                |
|
|         |                        |                                | GHSA-mpj8-q39x-wq5h Warn:                                |
|
|         |                        |                                | Project is vulnerable to:                                |
|
|         |                        |                                | GHSA-92r3-m2mg-pj97                                      |
|
|---------|------------------------|--------------------------------|----------------------------------------------------------
|-------------------------------------------------------------------------------------------------------------------------|

### CTX ###
→ ~ docker run -e GITHUB_AUTH_TOKEN=****** gcr.io/openssf/scorecard:stable --show-details --
repo=https://github.com/figlief/ctx
Starting [Vulnerabilities]
Starting [Fuzzing]
Starting [CII-Best-Practices]
Starting [Packaging]
Starting [Security-Policy]
Starting [Dangerous-Workflow]
Starting [SAST]
Starting [Pinned-Dependencies]
Starting [Code-Review]
Starting [Branch-Protection]
Starting [Maintained]
Starting [Binary-Artifacts]
Starting [CI-Tests]
Starting [Dependency-Update-Tool]
Starting [Contributors]
Starting [Token-Permissions]
Starting [Signed-Releases]
Starting [License]
Finished [CI-Tests]
Finished [Dependency-Update-Tool]
Finished [Maintained]
Finished [Binary-Artifacts]
Finished [Signed-Releases]
Finished [License]
Finished [Contributors]
Finished [Token-Permissions]
Finished [CII-Best-Practices]
Finished [Packaging]
Finished [Security-Policy]
Finished [Dangerous-Workflow]
Finished [Vulnerabilities]




Finished [Fuzzing]
Finished [Code-Review]
Finished [Branch-Protection]
Finished [SAST]
Finished [Pinned-Dependencies]

RESULTS
-------
Aggregate score: 2.6 / 10

Check scores:
|---------|------------------------|--------------------------------|--------------------------------|----------------------
-------------------------------------------------------------------------------------------|
|  SCORE  |          NAME          |             REASON             |            DETAILS             |
DOCUMENTATION/REMEDIATION                                    |
|---------|------------------------|--------------------------------|--------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| 10 / 10 | Binary-Artifacts       | no binaries found in the repo  |                                |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#binary-artifacts       |
|---------|------------------------|--------------------------------|--------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| 0 / 10  | Branch-Protection      | branch protection not enabled  | Warn: branch protection not    |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#branch-protection      |
|         |                        | on development/release         | enabled for branch 'master'    |
|
|         |                        | branches                       |                                |
|
|---------|------------------------|--------------------------------|--------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| ?       | CI-Tests               | no pull request found          |                                |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#ci-tests               |
|---------|------------------------|--------------------------------|--------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| 0 / 10  | CII-Best-Practices     | no effort to earn an OpenSSF   |                                |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#cii-best-practices     |
|         |                        | best practices badge detected  |                                |
|
|---------|------------------------|--------------------------------|--------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| 0 / 10  | Code-Review            | found 7 unreviewed changesets  |                                |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#code-review           |
|         |                        | out of 7 -- score normalized   |                                |
|
|         |                        | to 0                           |                                |
|
|---------|------------------------|--------------------------------|--------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| 0 / 10  | Contributors           | project has 0 contributing     |                                |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#contributors          |
```

```
|         |                       | companies or organizations --  |                               |
|
|         |                       | score normalized to 0          |                               |
|
|---------|-----------------------|--------------------------------|-------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| ?       | Dangerous-Workflow    | no workflows found             |                               |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#dangerous-workflow     |
|---------|-----------------------|--------------------------------|-------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| 0 / 10  | Dependency-Update-Tool | no update tool detected        | Warn: tool 'RenovateBot'      |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#dependency-update-tool |
|         |                       |                                | is not used Warn: tool        |
|
|         |                       |                                | 'Dependabot' is not used Warn: |
|
|         |                       |                                | tool 'PyUp' is not used       |
|
|---------|-----------------------|--------------------------------|-------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| 0 / 10  | Fuzzing               | project is not fuzzed          | Warn: no OSSFuzz integration  |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#fuzzing                |
|         |                       |                                | found Warn: no GoBuiltInFuzzer |
|
|         |                       |                                | integration found Warn:       |
|
|         |                       |                                | no PythonAtherisFuzzer        |
|
|         |                       |                                | integration found Warn:       |
|
|         |                       |                                | no CLibFuzzer integration     |
|
|         |                       |                                | found Warn: no CppLibFuzzer   |
|
|         |                       |                                | integration found Warn: no    |
|
|         |                       |                                | SwiftLibFuzzer integration    |
|
|         |                       |                                | found Warn: no RustCargoFuzzer |
|
|         |                       |                                | integration found Warn: no    |
|
|         |                       |                                | JavaJazzerFuzzer integration  |
|
|         |                       |                                | found Warn: no ClusterFuzzLite |
|
|         |                       |                                | integration found Warn: no    |
|
|         |                       |                                | HaskellPropertyBasedTesting   |



|
|         |                       |                                | integration found Warn: no    |
|
|         |                       |                                | TypeScriptPropertyBasedTesting |
|
|         |                       |                                | integration found Warn: no    |
|
|         |                       |                                | JavaScriptPropertyBasedTesting |
|
|         |                       |                                | integration found             |
|
|---------|-----------------------|--------------------------------|-------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| 10 / 10 | License               | license file detected          | Info: FSF or OSI recognized   |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#license                |
|         |                       |                                | license: LICENSE:1 Info:      |
|
|         |                       |                                | License file found in expected |
|
|         |                       |                                | location: LICENSE:1           |
|
|---------|-----------------------|--------------------------------|-------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| 0 / 10  | Maintained            | 0 commit(s) and 0 issue        |                               |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#maintained             |
|         |                       | activity found in the last 90  |                               |
|
|         |                       | days -- score normalized to 0  |                               |
|
|---------|-----------------------|--------------------------------|-------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| ?       | Packaging             | packaging workflow not         | Warn: no GitHub/GitLab        |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#packaging              |
|         |                       | detected                       | publishing workflow detected.  |
|
|---------|-----------------------|--------------------------------|-------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| ?       | Pinned-Dependencies   | no dependencies found          |                               |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#pinned-dependencies    |
|---------|-----------------------|--------------------------------|-------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| 0 / 10  | SAST                  | no SAST tool detected          | Warn: no pull requests merged  |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#sast                 |
|         |                       |                                | into dev branch               |
|
|---------|-----------------------|--------------------------------|-------------------------------|----------------------
-------------------------------------------------------------------------------------------|
| 0 / 10  | Security-Policy       | security policy file not       | Warn: no security policy file  |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#security-policy       |
```

```
|         |                       | detected                      | detected Warn: no security     |
|
|         |                       |                               | file to analyze Warn: no       |
|
|         |                       |                               | security file to analyze Warn: |
|
|         |                       |                               | no security file to analyze    |
|
|---------|-----------------------|-------------------------------|--------------------------------|--------------------
-------------------------------------------------------------------------------------------|
| ?       | Signed-Releases       | no releases found             |                                |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#signed-releases        |
|---------|-----------------------|-------------------------------|--------------------------------|--------------------
-------------------------------------------------------------------------------------------|
| ?       | Token-Permissions     | no tokens found               |                                |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#token-permissions      |
|---------|-----------------------|-------------------------------|--------------------------------|--------------------
-------------------------------------------------------------------------------------------|
| 10 / 10 | Vulnerabilities       | 0 existing vulnerabilities    |                                |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#vulnerabilities        |
|         |                       | detected                      |                                |
|
|---------|-----------------------|-------------------------------|--------------------------------|--------------------
-------------------------------------------------------------------------------------------|

### Node IPC ###
→ ~ docker run -e GITHUB_AUTH_TOKEN=****** gcr.io/openssf/scorecard:stable --show-details --
repo=https://github.com/RIAEvangelist/node-ipc
Starting [CI-Tests]
Starting [Dangerous-Workflow]
Starting [Signed-Releases]
Starting [Packaging]
Starting [Token-Permissions]
Starting [Dependency-Update-Tool]
Starting [Binary-Artifacts]
Starting [License]
Starting [Code-Review]
Starting [Vulnerabilities]
Starting [Pinned-Dependencies]
Starting [Branch-Protection]
Starting [Contributors]
Starting [Fuzzing]
Starting [CII-Best-Practices]
Starting [SAST]
Starting [Maintained]
Starting [Security-Policy]
Aggregate score: 3.2 / 10

Check scores:



Finished [Dangerous-Workflow]
Finished [CI-Tests]
Finished [Token-Permissions]
Finished [Dependency-Update-Tool]
Finished [Binary-Artifacts]
Finished [License]
Finished [Code-Review]
Finished [Vulnerabilities]
Finished [Signed-Releases]
Finished [Packaging]
Finished [Branch-Protection]
Finished [Contributors]
Finished [Pinned-Dependencies]
Finished [CII-Best-Practices]
Finished [SAST]
Finished [Maintained]
Finished [Security-Policy]
Finished [Fuzzing]

RESULTS
-------
|---------|-----------------------|-------------------------------|--------------------------------|--------------------
-------------------------------------------------------------------------------------------|
|  SCORE  |         NAME          |             REASON            |            DETAILS             |
DOCUMENTATION/REMEDIATION                                  |
|---------|-----------------------|-------------------------------|--------------------------------|--------------------
-------------------------------------------------------------------------------------------|
| 10 / 10 | Binary-Artifacts      | no binaries found in the repo |                                |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#binary-artifacts       |
|---------|-----------------------|-------------------------------|--------------------------------|--------------------
-------------------------------------------------------------------------------------------|
| 0 / 10  | Branch-Protection     | branch protection not enabled | Warn: branch protection not    |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#branch-protection      |
|         |                       | on development/release        | enabled for branch 'master'    |
|
|         |                       | branches                      |                                |
|
|---------|-----------------------|-------------------------------|--------------------------------|--------------------
-------------------------------------------------------------------------------------------|
| ?       | CI-Tests              | no pull request found         |                                |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#ci-tests              |
|---------|-----------------------|-------------------------------|--------------------------------|--------------------
-------------------------------------------------------------------------------------------|
| 0 / 10  | CII-Best-Practices    | no effort to earn an OpenSSF  |                                |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#cii-best-practices    |
|         |                       | best practices badge detected |                                |
|
|---------|-----------------------|-------------------------------|--------------------------------|--------------------
-------------------------------------------------------------------------------------------|
```

```
| 0 / 10  | Code-Review            | found 30 unreviewed changesets |                                 |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#code-review            |
|         |                        | out of 30 -- score normalized  |                                 |
|         |                        | to 0                           |                                 |
|
|---------|------------------------|--------------------------------|---------------------------------|------------------
-------------------------------------------------------------------------------|
| 10 / 10 | Contributors           | project has 5 contributing     | Info: director of front end     |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#contributors           |
|         |                        | companies or organizations     | technology and javascript       |
|         |                        |                                | engineering; creator @youtube;  |
|         |                        |                                | usaf veteran contributor        |
|         |                        |                                | org/company found, line         |
|         |                        |                                | contributor org/company         |
|         |                        |                                | found, beyond limits            |
|         |                        |                                | contributor org/company found,  |
|         |                        |                                | reploy-solucoes-corporativas    |
|         |                        |                                | contributor org/company         |
|         |                        |                                | found, lozep tecnologia         |
|         |                        |                                | iblue consulting contributor    |
|         |                        |                                | org/company found,              |
|
|---------|------------------------|--------------------------------|---------------------------------|------------------
-------------------------------------------------------------------------------|
| 10 / 10 | Dangerous-Workflow     | no dangerous workflow patterns |                                 |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#dangerous-workflow     |
|         |                        | detected                       |                                 |
|
|---------|------------------------|--------------------------------|---------------------------------|------------------
-------------------------------------------------------------------------------|
| 0 / 10  | Dependency-Update-Tool | no update tool detected        | Warn: tool 'RenovateBot'        |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#dependency-update-tool |
|         |                        |                                | is not used Warn: tool          |
|         |                        |                                | 'Dependabot' is not used Warn:  |
|         |                        |                                | tool 'PyUp' is not used         |
```

```
|
|---------|------------------------|--------------------------------|---------------------------------|------------------
-------------------------------------------------------------------------------|
| 0 / 10  | Fuzzing                | project is not fuzzed          | Warn: no OSSFuzz integration    |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#fuzzing               |
|         |                        |                                | found Warn: no GoBuiltInFuzzer  |
|         |                        |                                | integration found Warn:         |
|         |                        |                                | no PythonAtherisFuzzer          |
|         |                        |                                | integration found Warn:         |
|         |                        |                                | no CLibFuzzer integration       |
|         |                        |                                | found Warn: no CppLibFuzzer     |
|         |                        |                                | integration found Warn: no      |
|         |                        |                                | SwiftLibFuzzer integration      |
|         |                        |                                | found Warn: no RustCargoFuzzer  |
|         |                        |                                | integration found Warn: no      |
|         |                        |                                | JavaJazzerFuzzer integration    |
|         |                        |                                | found Warn: no ClusterFuzzLite  |
|         |                        |                                | integration found Warn: no      |
|         |                        |                                | HaskellPropertyBasedTesting     |
|         |                        |                                | integration found Warn: no      |
|         |                        |                                | TypeScriptPropertyBasedTesting  |
|         |                        |                                | integration found Warn: no      |
|         |                        |                                | JavaScriptPropertyBasedTesting  |
|         |                        |                                | integration found               |
|
|---------|------------------------|--------------------------------|---------------------------------|------------------
-------------------------------------------------------------------------------|
| 10 / 10 | License                | license file detected          | Info: FSF or OSI recognized     |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#license              |
|         |                        |                                | license: licence:1 Info:        |
|
```

```
|         |                        |                                | License file found in expected   |
|
|         |                        |                                | location: licence:1              |
|
|---------|------------------------|--------------------------------|----------------------------------|--------------------
-------------------------------------------------------------------------------------------|
| 0 / 10  | Maintained             | 0 commit(s) and 0 issue        |                                  |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#maintained             |
|         |                        | activity found in the last 90  |                                  |
|
|         |                        | days -- score normalized to 0  |                                  |
|
|---------|------------------------|--------------------------------|----------------------------------|--------------------
-------------------------------------------------------------------------------------------|
| ?       | Packaging              | packaging workflow not         | Warn: no GitHub/GitLab           |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#packaging              |
|         |                        | detected                       | publishing workflow detected.    |
|
|---------|------------------------|--------------------------------|----------------------------------|--------------------
-------------------------------------------------------------------------------------------|
| 3 / 10  | Pinned-Dependencies    | dependency not pinned by hash  | Warn: GitHub-owned GitHubAction  |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#pinned-dependencies    |
|         |                        | detected -- score normalized   | not pinned by hash:              |
|
|         |                        | to 3                           | .github/workflows/node.js.yml:26 |
|
|         |                        |                                | Warn: GitHub-owned GitHubAction  |
|
|         |                        |                                | not pinned by hash:              |
|
|         |                        |                                | .github/workflows/node.js.yml:28 |
|
|         |                        |                                | Info:   0 out of   2             |
|
|         |                        |                                | GitHub-owned GitHubAction        |
|
|         |                        |                                | dependencies pinned Info:        |
|
|         |                        |                                |   1 out of   1 npmCommand        |
|
|         |                        |                                | dependencies pinned              |
|
|---------|------------------------|--------------------------------|----------------------------------|--------------------
-------------------------------------------------------------------------------------------|
| 0 / 10  | SAST                   | no SAST tool detected          | Warn: no pull requests merged    |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#sast                 |
|         |                        |                                | into dev branch                  |
|
|---------|------------------------|--------------------------------|----------------------------------|--------------------
```

```
-------------------------------------------------------------------------------------------|
| 0 / 10  | Security-Policy        | security policy file not       | Warn: no security policy file    |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#security-policy        |
|         |                        | detected                       | detected Warn: no security       |
|
|         |                        |                                | file to analyze Warn: no         |
|
|         |                        |                                | security file to analyze Warn:   |
|
|         |                        |                                | no security file to analyze      |
|
|---------|------------------------|--------------------------------|----------------------------------|--------------------
-------------------------------------------------------------------------------------------|
| ?       | Signed-Releases        | no releases found              |                                  |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#signed-releases        |
|---------|------------------------|--------------------------------|----------------------------------|--------------------
-------------------------------------------------------------------------------------------|
| 0 / 10  | Token-Permissions      | detected GitHub workflow       | Warn: no topLevel                |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#token-permissions      |
|         |                        | tokens with excessive          | permission defined:              |
|
|         |                        | permissions                    | .github/workflows/node.js.yml:1  |
|
|         |                        |                                | Info: no jobLevel write          |
|
|         |                        |                                | permissions found                |
|
|---------|------------------------|--------------------------------|----------------------------------|--------------------
-------------------------------------------------------------------------------------------|
| 7 / 10  | Vulnerabilities        | 3 existing vulnerabilities     | Warn: Project is vulnerable      |
https://github.com/ossf/scorecard/blob/658a77b501feea0c4f0fcedbf84e7cb266066849/docs/checks.md#vulnerabilities        |
|         |                        | detected                       | to: GHSA-93q8-gq69-wqmw Warn:    |
|
|         |                        |                                | Project is vulnerable to:        |
|
|         |                        |                                | GHSA-f8q6-p94x-37v3 Warn:        |
|
|         |                        |                                | Project is vulnerable to:        |
|
|         |                        |                                | GHSA-c2qf-rxjj-qqgw              |
|
|---------|------------------------|--------------------------------|----------------------------------|--------------------
-------------------------------------------------------------------------------------------|
```

Appendix B

CURRICULUM SLIDES

OPEN-SOURCE RISK & MITIGATION

Linux Foundations CHAOSS – Risk Working Group

INTRODUCTION

- Business Risk
- Code Quality
- Licensing
- Security
- Transparency
- Dependency Risk Assessment

BUSINESS RISK

Goal: Understand how active a community exists around/to support a given software package.

- Average Issue Resolution Time
- Bus Factor
- Committers
- Elephant Factor
- Issue Open Age
- Issue Volume
- Lines of Code

BUSINESS RISK- AVERAGE ISSUE RESOLUTION TIME

- How long does it take for an issue to be resolved?

BUSINESS RISK- BUS FACTOR

- How high is the risk to a project should the most active people leave?
- Visualizes the question "how many contributors can we lose before a project stalls?
- Identify how widely the work in a project is distributed across contributors.
- Identify the key people in a project that are doing the majority of the work.

BUSINESS RISK- COMMITTERS

- How robust are the contributors to a community?
- Fewer code contributors may indicate projects less open to outside contribution, or simply projects that have a small number of individuals who understand and contribute to the code base.

BUSINESS RISK- ELEPHANT FACTOR

- What is the distribution of work in the community across companies?
- Compare how dependent a project is on a small set of corporate contributors

BUSINESS RISK- ISSUES OPEN AGE

- How long is an issue open for before it is closed?

BUSINESS RISK- VOLUME ISSUES

- What is the volume of open issues?

BUSINESS RISK- LINES OF CODE

- How many lines of code have been contributed?

CODE QUALITY

Goal: Understand the quality of a given software package.

- Code Complexity
- Pull Request Process
- Test Coverage
- Defect Resolution Time

CODE QUALITY- CODE COMPLEXITY

- How complex is the code?
- High code complexity brings with it a higher level of code defects, making the code costlier to maintain
- Static code analysis

CODE QUALITY- PULL REQUEST PROCESS

- What is the process of making a Pull Request?

CODE QUALITY- TEST COVERAGE

- How well is the code tested?
- NIST recommends 80% code coverage

CODE QUALITY- DEFECT RESOLUTION TIME

- How much time does a project take to resolve defects once they have been reported and recorded?

LICENSING

Goal: Understand the potential IP issues associated with a given software package use. exists around/to support a given software package.

- License Count
- License Coverage
- License Declared
- OSI Approved Licenses
- SPDX Document

LICENSING - LICENSE COUNT

- How much time does a project take to resolve defects once they have been reported and recorded?

SECURITY

Goal: Understand security processes and procedures associated with the software's development.

- OpenSSF Best Practices
- Language Declaration README
- Language Source Proportion

SECURITY – OPENSSF BEST PRACTICES

- OpenSSF – Open-Source Security Foundation
- What is the current OpenSSF Best Practices status for the project?
- https://www.bestpractices.dev/en/criteria/0

SECURITY – LANGUAGE DECLARATION README

- How many languages were used?

SECURITY – LANGUAGE SOURCE PROPORTION

- What is the proportion of language sources used?

TRANSPARENCY

Goal: Understand how transparent a given software package is with respect to dependencies, licensing, security processes, etc.

- Software Bill of Materials

TRANSPARENCY - SOFTWARE BILL OF MATERIALS

- Does the software package have a standard expression of dependencies, licensing, and security-related issues?

DEPENDENCY RISK ASSESSMENT

Goal: Understand software dependency risk.

- Upstream Code Dependencies
- Libyears

DEPENDENCY RISK MANAGEMENT - UPSTREAM CODE DEPENDENCIES

- What projects and libraries does my project depend on?

DEPENDENCY RISK MANAGEMENT - LIBYEARS

- What is the age of the project's dependencies compared to current stable releases?

REFERENCES

- [1] https://github.com/chaoss/wg-risk/tree/main
- [2] https://nvlpubs.nist.gov/nistpubs/ir/2021/NIST.IR.8397.pdf

Weak Links and Supply Chain Attacks

Patch || Breach
Patch && Breach

- Used to be patch on a Friday to prevent a breach on Monday
- Now it's patch on a Friday and breach on Monday

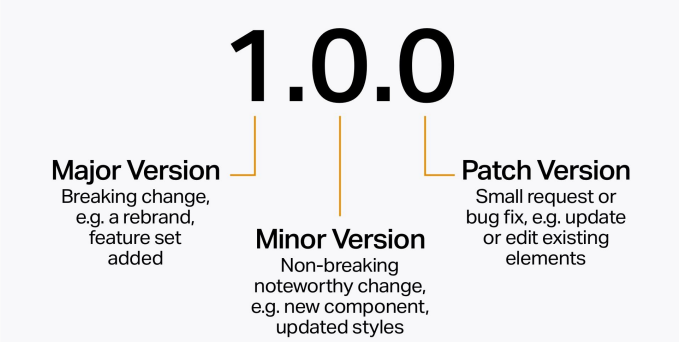
Open-Source Stats

- 96% of packages use 3rd-party components
- 80% of code in the software supply chain comes from 3rd-party packages
- Sonatype - Supply Chain attacks have increased by 650% in 2021

Package Managers

- 1993 - Linux
- Help engineers manage dependencies
- Pull dependencies from multiple sources
- Many use semantic versioning

Semantic Versioning



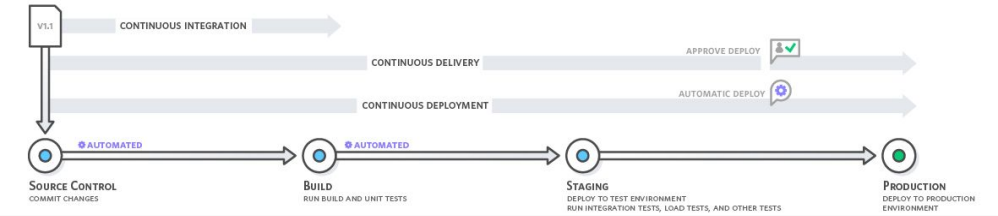
Continuous Integration - Atlassians Def

- Practice of automating the integration of code changes from multiple contributors into a single software project.
- Allowing developers to frequently merge code changes into a central repository where builds and tests then run.
- Automated tools are used to assert the new code's correctness before integration.

Continuous Delivery - Amazon's Def

- Software development practice where code changes are automatically prepared for a release to production.
- A pillar of modern application development, continuous delivery expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage.
- When properly implemented, developers will always have a deployment-ready build artifact that has passed through a standardized test process.

Continuous Delivery - Amazon's Def



NPM

- Package manager for NodeJS
- > 1.8 million packages
- Package.json
  - Caret (^) - 1.0.2 install version 1.0.2 or the latest minor such as 1.1.0 … 1.*.*
  - Tilde (~) 1.0.2 - Install version 1.0.2 or latest patch such as 1.0.4 … 1.0.*

React

- React is a JavaScript library for building user interfaces.
- https://github.com/facebook/react

Attack Vectors

- Malicious Package Release
- Social Engineering
- Account Takeover
- Ownership Transfer
- Remote Execution

Weak Links - Expired Maintainer Domain

- An attacker can hijack a component if a maintainer's domain is expired and does not have 2FA authentication set up on their account.

Weak Links - Installation Script

- An attacker can use installation scripts to run commands that perform malicious acts through the package installation step

Weak Links - Unmaintained Package

- Attackers can target packages that are more likely to take over and sneak in malware due to lack of maintenance.

Weak Links - Too many Maintainers

- A package with too many maintainers will provide an attacker many targets to exploit account takeover and social engineering attacks.

Weak Links - Too many contributors

- An attacker can sneak in malicious code, bypassing the maintainer's radar when a maintainer is responsible for many contributors

Weak Links - Overloaded Maintainer

- An attacker may target a maintainer who owns many packages because the maintainer may not have enough time to maintain security of all the packages.

Exercise - 30 minutes

On Monday you heard about a competitor that was hacked. Through the grapevine you learn that they follow CI/CD practices and that they build / release new software at least twice a day. The team swears they did not bring in any new packages and that they haven't touched their package.json file for over 6 months. How could they have been breached?

You want to defend your organization against such an attack.

- **In groups:** Describe how the competitor might have been hacked and brainstorm 3 different mechanisms you could implement to protect your organization from such an attack.
- Be prepared to brief your 5 minute plan to your organization's AppSec Director

Review

- Patch and Breach
- Package Managers
- Semantic Versioning
- CI & CD
- Attack Vectors
- Weak Links in Packages