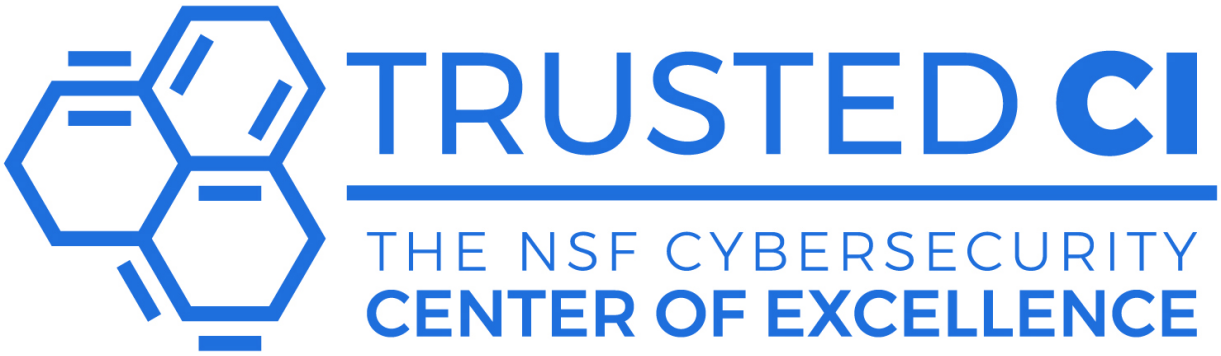

Open OnDemand Engagement

Final Report

July 12, 2021

Distribution: Public

Authors: Tarun Anand, Diana Cimmer, Elisa Heymann, Ryan Kiser, Bart Miller, Ian Ruh, Ritvik Bhawnani, Kelli Shute, John Zage

About Trusted CI

The mission of Trusted CI is to provide the NSF community with a coherent understanding of cybersecurity, its importance to computational science, and what is needed to achieve and maintain an appropriate cybersecurity program.

Acknowledgments

Trusted CI's engagements are inherently collaborative; the authors would like to thank the Open OnDemand team, including Alan Chalker, Eric Franz, Kyle Earley, Jeffrey Ohrstrom, Travis Ravert, and Roy Sizemore for the collaborative effort that made this engagement and document possible.

This document is a product of Trusted CI. Trusted CI is supported by the National Science Foundation under Grant #1920430. For more information about Trusted CI, please visit: http://trustedci.org/. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Using & Citing this Work

Cite this work using the following information:

T. Anand, D. Cimmer, E. Heymann, R. Kiser, B. Miller, I. Ruh, R. Bhawnani, K. Shute, and J. Zage, "Trusted CI: Open OnDemand Engagement Final Report", NSF Cybersecurity Center of Excellence, Trusted CI, trustedci.org, June 2021. Available: http://hdl.handle.net/2022/26590

This work and other engagement reports are available on the web at the following URL:
https://www.trustedci.org/reports

Executive Summary

During this 6-month engagement, Trusted CI collaborated with Open OnDemand (OOD) to develop internal OOD expertise with conducting vulnerability assessments using the FPVA methodology and to incorporate key elements of the FPVA assessment process into their own internal procedures. This will enable Open OnDemand to improve the project's ability to maintain the security of their software as changes are made and to identify and mitigate future vulnerabilities. We focused on evaluating dependency and static analysis tools as well as the vulnerability disclosure and remediation processes. The following report contains a summary of the work performed during the engagement, our findings, and areas Open OnDemand should consider for ongoing security and process improvements. This public version of the report has had sensitive information removed.

1. Background

Open OnDemand[1] is funded by NSF OAC and is an open-source HPC portal based on the original OnDemand portal developed at Ohio Supercomputer Center.[2] The goal of Open OnDemand is to provide an easy way for system administrators to provide web access to their HPC resources. Open OnDemand is now facing increased community adoption. As a result, it is becoming a critical production service for many HPC centers and clients.

Trusted CI began this engagement with Open OnDemand to support their efforts to further develop the project's ability to produce secure software. Trusted CI conducted an in-depth vulnerability assessment in 2018, applying the First Principles Vulnerability Assessment (FPVA) methodology[3] to Open OnDemand software to identify vulnerabilities and provide recommendations for how to address them. The results of this prior assessment helped to inform the activities of this engagement. During the course of the prior vulnerability assessment using the FPVA methodology, Trusted CI staff worked directly to test Open OnDemand's software to identify vulnerabilities with support from the Open OnDemand team.

2. Factual Summary

The sections below describe the primary activities conducted in support of the 2021 engagement. These are followed in Sections 3 and 4 by descriptions of our findings and recommendations, respectively.

2.1. Significant changes identified since prior engagement

Trusted CI engaged with Open OnDemand and performed an in-depth vulnerability assessment of the code base using FPVA in 2018. At the beginning of this current engagement, we collaborated with the Open OnDemand team to identify the major changes or additions to the code base since that engagement. Five primary changes were identified and are summarized below.

1) **Linux host adapter**
   The linux host adapter that was added by the Open OnDemand team allows running jobs and programs that are not amenable to being run by traditional HPC schedulers,

[1] https://openondemand.org/
[2] https://www.osc.edu/
[3] James A. Kupsch, Barton P. Miller, Eduardo César, and Elisa Heymann, "First Principles Vulnerability Assessment", 2010 ACM Cloud Computing Security Workshop (CCSW), Chicago, IL, October 2010.

including IDEs and desktop environments. It also allows Open OnDemand to work with hosts that do not have a supported scheduler installed.

2) **SSH Wrapper**
   A mechanism to manage SSH connections to compute nodes was proposed by an Open OnDemand community member and merged into the repository. The wrapper allows operators of Open OnDemand instances to customize how SSH connections are routed to compute nodes. For example, if one login node is used for multiple clusters, this lets the center route SSH connections to the correct cluster.

3) **Authentication providers**
   The default authentication provider for Open OnDemand was changed from Basic Auth to Dex.[4]

4) **Nginx pre hook**
   A 'hook' was added to allow scripts to run immediately before the creation of a per-user Nginx (PUN) instance. This allows additional events to be triggered, such as mounting a user's home directory.

5) **Balance shown in the dashboard**
   The ability to show a user's and/or a project's remaining compute balance was added to the dashboard.

2.2. Automated Assessment Tools

At the beginning of the engagement, the Open OnDemand team indicated that their current use of automated assessment tools was limited to Dependabot[5] for dependency checking and ShellCheck[6] for basic analysis of shell scripts. They further indicated that there were some limitations of Dependabot, specifically its ability to assess multiple branches of the same repository, that they were interested in trying to resolve. The Trusted CI team performed comparisons of different automated dependency tools and of different static analysis tools for both Ruby and JavaScript.

[4] https://github.com/dexidp/dex
[5] https://dependabot.com/
[6] https://github.com/koalaman/shellcheck

Based on the lines of code analysis of the main OnDemand repository[7] (Appendix A), the two primary languages used were found to be Ruby and JavaScript. For both languages, a set of both open source and commercial tools to compare were selected based on their usage by other projects and their maintenance history (only applicable to open source tools). The Ruby static analysis tools selected for comparison were Brakeman,[8] DawnScanner,[9] Semgrep,[10] DeepSource,[11] SonarQube,[12] and Codacy.[13] The JavaScript static analysis tools selected for comparison were ESLint,[14] JSHint,[15] Semgrep, DeepScan,[16] Codacy, and SonarQube.

For the comparisons of tools for both languages, each tool was run on the main OnDemand repository and the issues reported were collected. For each issue reported by a tool, we determined whether or not the issue was a false positive or if it was likely an issue. In some cases, we were not able to determine the relevancy of the reported issue due to the depth of understanding of the codebase it would require.

The findings of our comparisons are covered in Sections 3.1 and 3.2, and the derived recommendations are in Sections 4.1 and 4.2 respectively.

2.3. Current Vulnerability Disclosure Process

When requested, Open OnDemand staff characterized their current vulnerability reporting and management processes as follows:

1. Vulnerabilities and security concerns are reported by sending a message to the ood-users@lists.osc.edu mailing list.
2. Messages to the list are reviewed before distribution by Alan Chalker, Eric Franz, and Jeff Ohrstrom.[17] Any messages regarding security concerns or vulnerabilities are held until triage is completed.
    a. Team members will occasionally receive reports directly. When this occurs, the team member forwards it to Eric and Jeff.

[7] https://github.com/OSC/ondemand
[8] https://brakemanscanner.org/
[9] https://github.com/thesp0nge/dawnscanner
[10] https://semgrep.dev/
[11] https://deepsource.io
[12] https://www.sonarqube.org/
[13] https://www.codacy.com/
[14] https://eslint.org/
[15] https://jshint.com/
[16] https://deepscan.io/
[17] Note that Eric Franz left the project in May.

    b. Vulnerability reports are given high priority and investigated immediately.
3. Once an RPM is available for the updated version the user community is notified through the ood-users@lists.osc.edu mailing list and through an announcement on Open OnDemand's Discourse forum.[18]
4. The Open OnDemand team expects to remediate identified vulnerabilities within two weeks.

2.4. Code Review

First Principles Vulnerability Assessment is an analyst-centric (manual) methodology that aims to focus the analyst's attention on the part of the software system and its resources that are most likely to contain vulnerabilities and that would provide access to high-value assets. FPVA finds new threats to a system and is not dependent on a list of known threats. The FPVA methodology consists of five steps for evaluating a given piece of software.

1. **Architectural Analysis:** Determine the major structural components of the system and how they interact. At this point, we produce architectural diagrams that illustrate the structure of the system.
2. **Resource Identification:** Identify key resources accessed by each component. Examples of these resources include files, databases, logs, and devices. The resource diagrams that are produced illustrate these resources and their connection to system components.
3. **Trust and Privilege Analysis:** Identify the trust assumptions about each component, answering such questions as how they are protected and who can access them. Associated with trust is describing the privilege level at which each executable component runs. The artifact produced at this stage is a further labeling of the basic diagrams with trust levels and labeling of interactions with delegation information.
4. **Component Evaluation:** Examine relevant components in depth. A key aspect of the FPVA methodology is that this step is guided by information obtained in the first three steps, helping to prioritize the work so that high value targets are evaluated first.
5. **Dissemination of Results:** When we perform an assessment using FPVA methodology, we create a report for each vulnerability found and include identified bugs as well as areas that have been investigated but where no bugs or vulnerabilities were found. We then disseminate the final report to the requesting parties (i.e., the lead of the development team).

[18] https://discourse.osc.edu/c/open-ondemand/announcements

From the start of this engagement, our goal was not to perform a second FPVA for the Open OnDemand team but to assist them in learning to conduct the assessments on their own. Throughout the engagement we provided them feedback as they constructed and refined the architectural and resource diagrams, all of which are included in Appendix D at the end of this report. Going into step four of the assessment, component evaluation, we suggested to the Open OnDemand team a list of potential issues to examine in the code (included in Appendix D). Had Trusted CI been performing the assessment, that would have been the starting points for our detailed code analysis.

Throughout the engagement, it was the intent of the Trusted CI team to provide the Open OnDemand team with the relevant knowledge and background on the FPVA methodology to continue to utilize it beyond the end of the engagement.

3. Findings

3.1. Dependency Analysis Tools

Dependency checking tools are commonly employed to ensure the continued security of a project's dependencies as new vulnerabilities are found over time. Due to their widespread use, most common languages are supported by at least one dependency tool. For this engagement, the Trusted CI team identified and assessed a variety of tools that report vulnerable dependencies for JavaScript and Ruby projects. The dependency tools assessed were identified based on their popularity and their integration with continuous integration (CI) tooling. We assessed and compared the results obtained by Dependabot,[19] Snyk,[20] Depfu,[21] and OWASP Dependency-Check[22] when applied to the Open OnDemand code.

The following sections describe the different aspects of these dependency tools that were used in our comparison and selection process.

3.1.1. How do dependency tools work?

A primary attribute of a dependency analysis tool is the sources of information it uses for vulnerabilities. As a dependency tool relies on associating a certain version of a piece of software with a vulnerability, it is essential that the tool has access to as complete a list of

[19] Dependabot
[20] Snyk
[21] Depfu
[22] OWASP Dependency-Check

known security issues as possible. One of the most common sources is the National Vulnerability Database (NVD),[23] which, of the tools we assessed, is used by Dependabot, Snyk, and OWASP Dependency-Check. The NVD records Common Vulnerabilities and Exposures (CVEs) that are assigned to a specific vulnerability that is found in some range of versions of a piece of software. However, not all vulnerabilities are assigned a CVE, so many tools rely on other sources of information as well. Dependabot, Snyk, and OWASP Dependency-Check all use the NPM Security Advisory[24] database, which tracks vulnerabilities found in NPM packages, in addition to the NVD. Furthermore, both Dependabot and Snyk use internal databases of vulnerabilities; Dependabot, as it is owned by GitHub, checks for relevant GitHub Security Advisories that apply to the code, and Snyk uses a proprietary collection of vulnerabilities that they maintain. Unfortunately, it is not clear what sources of information are used by Depfu.

To identify the versions of software dependencies used in a project, dependency tools scan the frameworks and libraries, including transitive dependencies, used throughout the codebase, and then compare the versions used to the versions known to have vulnerabilities. The build systems used by Open OnDemand for both Ruby and JavaScript rely on manifest files that list the dependencies required by the project and allow the build system to resolve the required versions and fetch them from a package repository automatically, rather than require a developer to do so manually. Manifest files may specify specific versions that are required (e.g. 3.2.1.) or there may be a list of versions that are compatible (e.g., 3.x). To determine the actual version that is used by the project, dependency tools depend on the build tool to resolve the versions specified in the manifest file to specific versions in a lock file, detailed in Section 3.1.2.

3.1.2. Lockfiles

Manifest files like package.json contain information about a project's dependencies and versions using semantic versioning. These versions indicate to the build tool how to handle future package updates. When a package author publishes a new version of their package, the package manager will install the most recent version at build time. This update mechanism can result in versioning inconsistencies in the application's dependencies across different developers.

To combat this issue, lockfiles are used to install the exact same versions of packages so that subsequent installs are identical. Thus, lockfiles capture the exact dependency tree used by the

[23] National Vulnerability Database (NVD)
[24] NPM Security Advisory

project. Lockfiles help dependency tools gain a more accurate picture of the project and increase the reliability of their reports.

3.1.3. Command-line and web integration of dependency tools

The tools that we have assessed have two types of integrations: command-line and web. A command-line integration scans for vulnerabilities in projects on the user's local machine. The tool uses the build environment to determine the actual packages used by the project. On the other hand, a web integration scans for vulnerabilities in a repository found on a repository hosting service like GitHub. Unlike command-line integration, a web integration cannot access the user's build environment.

Dependency tools with command-line integration can access more information about a project, such as private dependencies and the specifics of the build environment. Private dependencies are dependencies that are not open source and require authentication. The CLI can access these dependencies because the build tool for a project can access them using credentials in the environmental variables. Tools with web integration only have access to the project's dependency files and try to mimic the operation of the build tool. Thus, the vulnerability reports from a command-line integration can be more accurate.

Both Depfu and Dependabot only provide web integration. OWASP Dependency-Check provides a command-line integration, and Snyk provides a command-line and web integration.

To illustrate the difference between web and command-line integration, Figure 1 shows the ways Snyk can be used when applied to the Open OnDemand code. The outermost group shows the number of vulnerabilities found by Snyk's web integration and includes dev dependencies. The middle group shows the number of vulnerabilities found by Snyk's web integration excluding devDependencies. Last, the innermost group shows the number of vulnerabilities found by Snyk's command-line integration excluding devDependencies.

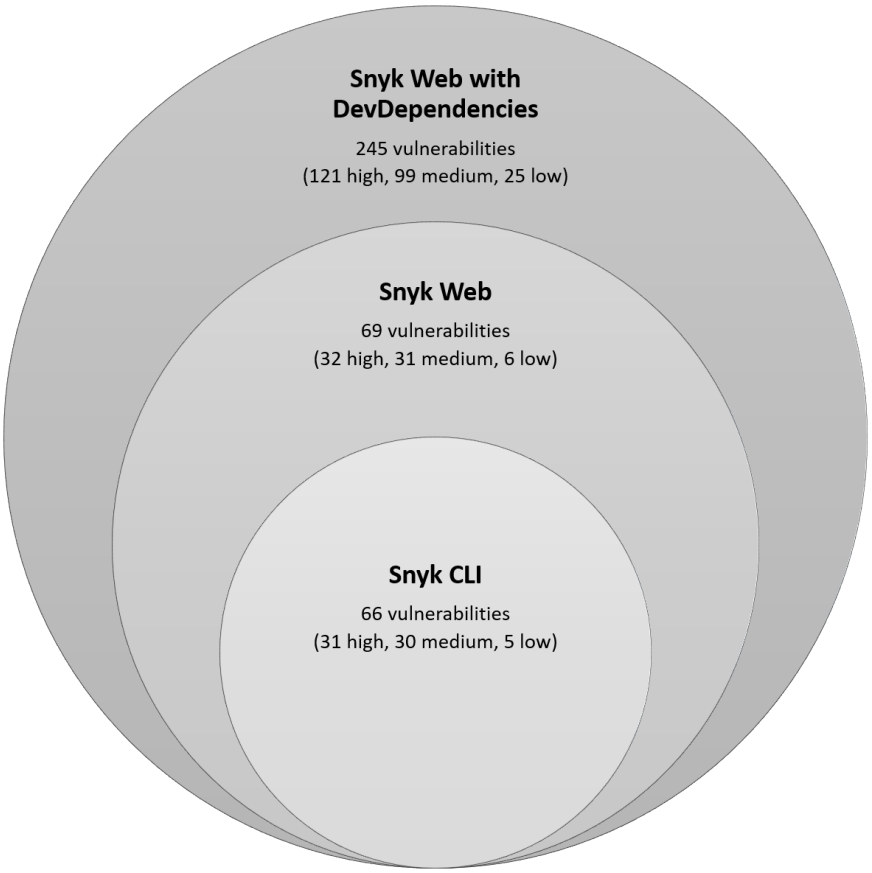

Figure 1. Venn diagram of the different ways Snyk can be used when applied to the Open OnDemand code

3.1.4. Branches

Currently, the Open OnDemand team receives alerts only for vulnerabilities on the main branch. Both Snyk and OWASP Dependency-Check's command-line integration can scan the checked-out branch of a repository, which allows users to find and fix vulnerabilities in multiple branches. Snyk's web integration finds vulnerabilities only in the default branch; support for multiple branches is in beta.

3.1.5. Vulnerability suppression

Suppressing vulnerabilities carries a high level of risk and is generally not recommended. Nevertheless, there may be reasons for ignoring vulnerabilities. For example, a vulnerability may be a false positive or does not have a suitable remediation strategy yet.

Both Snyk and OWASP Dependency-Check support suppressing specific packages, directories, and package managers.

DevDependencies are packages only used during the development of a project and not at runtime. We recommend suppressing devDependencies where possible. Snyk provides an option to ignore devDependencies. Figure 1 shows the difference between including and excluding devDependencies when Snyk is applied on the Open OnDemand code. Currently, OWASP Dependency-Check does not provide an option to skip devDependencies for yarn projects.

3.1.6. How to run dependency tools

Below are the commands we used for each tool when applied to the Open OnDemand code at version v1.8.19.

- Dependabot:

  Used GUI and permitted the tool to access the GitHub repository.

- Snyk:

  ```
  snyk test --all-projects --detection-depth=100 --json-file-output=snyk.json
  ```

- Depfu:

  Used GUI and permitted the tool to access the GitHub repository.

- OWASP Dependency-Check:

  ```
  dependency-check.bat --project "Open OnDemand" --scan "." --bundleAudit
  'bundle-audit.bat' --yarn 'yarn.cmd'
  ```

3.1.7. Comparison of dependency tools

	Dependabot	Snyk (CLI)	Depfu	OWASP Dependency-Check
High	3	31	0	28
Medium	2	30	0	26
Low	0	5	0	14
No Severity Provided	0	0	7	0
Total Errors	**5**	**66**	**7**	**68**
Unique Errors	**3**	**40**	**3**	**35**

Table 1. Comparison of the number of vulnerabilities reported by each tool when applied to the Open OnDemand code at version v1.8.19

We identified significant differences in package names, error identifiers, and severities reported by each tool when applied to the Open OnDemand code. First, Dependabot's and Depfu's results are relatively sparse and fail to identify several high severity vulnerabilities. Table 1 compares the number of vulnerabilities found by each tool.

Second, OWASP Dependency-Check reports several vulnerabilities using NPM identifiers rather than CVE identifiers. NPM identifiers may not always have a one-to-one correspondence with CVE identifiers. This disparity made it challenging to compare tools using the identifiers alone.

Third, Depfu does not provide any severity information. Severity levels help organizations prioritize resources towards high-risk vulnerabilities.

For these reasons, we created a table (Appendix B) that compares the tools based on the vulnerable packages they reported. Each row in the table shows the corresponding identifiers and severity for a package. For example, row 2 of Appendix B indicates that Dependabot, Snyk, and Depfu identified actioncable as a vulnerability. Moreover, only Dependabot and Snyk reported it as a high severity vulnerability.

3.1.8. Pricing considerations

Snyk Web provides unlimited tests for public open-source repositories. CLI projects are treated as private repositories and are limited to 200 tests per month. OWASP Dependency-Check is an

open-source and free tool. Dependabot is a free tool that is integrated in GitHub. Depfu provides unlimited tests for public open-source repositories.

3.2. Static Analysis Tools

Static analysis tools are commonly employed in continuous integration (CI) workflows and help detect poor coding style, potential errors, and security flaws in the project. Due to their widespread use, many common languages are supported by at least one static analysis tool. For this engagement, the Trusted CI team identified and assessed a variety of tools that report security issues for JavaScript and Ruby. The static analysis tools assessed were selected based on their popularity and their integration in CI workflows. We compared the results produced by these tools when applied to the Open OnDemand code at commit 4541d6b. For JavaScript, files we compared Semgrep, Codacy, and SonarQube; for Ruby files we compared Semgrep, Brakeman, DawnScanner, and DeepSource.

The following sections describe the different aspects of these static analysis tools used in our comparison and selection process.

3.2.1. How do static analysis tools work?

A primary attribute of a static analysis tool is the set of language-specific rules and patterns it uses to search for style, error-prone, and security issues in a project. In this engagement, we mainly focused on security issues. A reliable tool defines rules that align with industry standards and incorporate rules that find security issues in the languages it supports. The tools we have assessed primarily rely on the developer community to create and maintain rules. Open source tools like Semgrep, Brakeman, and DawnScanner use contributions from developers and commercial tools like Codacy, SonarQube, and DeepSource use community forums to manage rules. Additionally, Codacy and Semgrep use built-in tools and plugins to increase their coverage, as detailed in Section 3.2.4.

Unlike dynamic analysis, static analysis tools do not execute the source code. As a result, these tools may not have complete visibility into a project's execution environment. Moreover, some tools do not perform as comprehensive or semantic analysis as other tools, returning many false positives, detailed in Section 3.2.3. The tools that we assessed also provide suppression capabilities and support custom rules to reduce unreliable and unactionable issues.

3.2.2. Code formatting and minified files

Tool	Number of Security Issues Before Formatting	Number of Security Issues After Formatting
Codacy	19	830
SonarQube	10	18

Table 2. Comparison of the number of security issues reported by Codacy and SonarQube before and after formatting the Open OnDemand code at commit 4541d6b

While assessing static analysis tools for JavaScript files, we discovered that code formatting profoundly affects the number of results found by some static analysis tools. Code formatting organizes expressions into different lines without changing its functionality. Without formatting the code, some static analysis tools report a lower number of security issues. The Open OnDemand code was formatted using the Prettier[25] tool. Table 2 compares the number of security issues returned by Codacy and SonarQube when applied to the original repository and formatted repository. For both tools, the security issues from the non-formatted code are a subset of the security issues from the formatted code.

This problem is amplified when tools analyze minified JavaScript files,[26] which contain large amounts of code on the same line. Minification is the process of removing unnecessary characters from the source code without changing its functionality. The process is needed to improve web performance and results in smaller file sizes. Minified files also make it challenging to understand and resolve reported security issues because of its optimizations to functions and variables. Minified files create a local burden for tracking and applying updates. Open OnDemand uses about 208 minified files imported from Ace Editor.[27] Those files haven't been modified in a long time, and entail security risks, as they are not maintained.

3.2.3. False positives

Codacy calls other static analysis tools, and Semgrep uses plugins with rules from other tools to analyze a codebase. These tools are configurable and can be adapted to meet an organization's coding standard. ESLint is one of the tools used by Codacy and Semgrep. It contains rules that identify poor coding practices and security issues in JavaScript code. While all the tools we assessed report false positives, ESLint appears to have a limited understanding of the code's

[25] Prettier
[26] Why minify JavaScript code? - Cloudflare
[27] Ace Editor

context and triggers a higher number of false positives. Understanding and resolving an overwhelming number of results consumes a significant amount of time and hides true positives. Moreover, if most of these results are false positives, this tool can negatively impact the team's productivity. Instead, using a tool that is able to examine the context of the code and tracks data flow can reduce the number of false positives.

3.2.4. Tool configuration

The static analysis tools that we assessed provide multiple configuration options to reduce false positives and unactionable results. Semgrep offers both a GUI and CLI that allow users to exclude files and directories, specify which rulesets to use, build custom rules, and specify which language and frameworks to analyze. Additionally, Semgrep also provides rules ported from open-source security tools like ESLint. These configurations offer users flexibility but also make Semgrep challenging to set up. In our comparison, we used 2 rulesets for JavaScript and 1 ruleset for Ruby.

SonarQube offers a GUI that allows users to filter results by issue type, severity, and language. Furthermore, it supports custom rules and importing third-party rulesets.

Codacy offers a GUI and CLI that allows users to exclude files and directories, specify which language and frameworks to analyze, and build custom rules. Furthermore, this tool allows users to incorporate and configure a variety of sub-tools like ESLint and Brakeman to increase their issue coverage.

Brakeman is a CLI tool that allows users to exclude files and directories, filter issues by confidence level, and exclude certain rules.

DawnScanner is a CLI tool that allows users to restrict certain checks like code-style issues. Due to the way DawnScanner operates, it was necessary to run the tool on each app in Open OnDemand individually.

DeepSource is a CLI tool that allows users to exclude files and directories in its configuration file.

Unlike Dependency Tools, running a tool on a GUI or CLI does not impact the reliability of the results. A GUI integration offers a more user-friendly experience with collaboration tools, background notifications, and interactive graphs and diagrams. Additionally, it is simpler to make configuration changes in a GUI.

3.2.5. Running static analysis tools

3.2.5.a. Commands used for JavaScript Tools

Below are the commands we used for each JavaScript tool when applied to the Open OnDemand code at commit 4541d6b.

- Semgrep (ESLint plugin):

```
docker run --rm -v "${PWD}:/src" returntocorp/semgrep --lang=js
--config=p/eslint-plugin-security --json -o semgrep-test.json ./
```

- Semgrep (filtered JavaScript ruleset):

  Used GUI with filtered pre-written rules. Figure 2 shows how we added a new ruleset or policy using Semgrep's GUI.

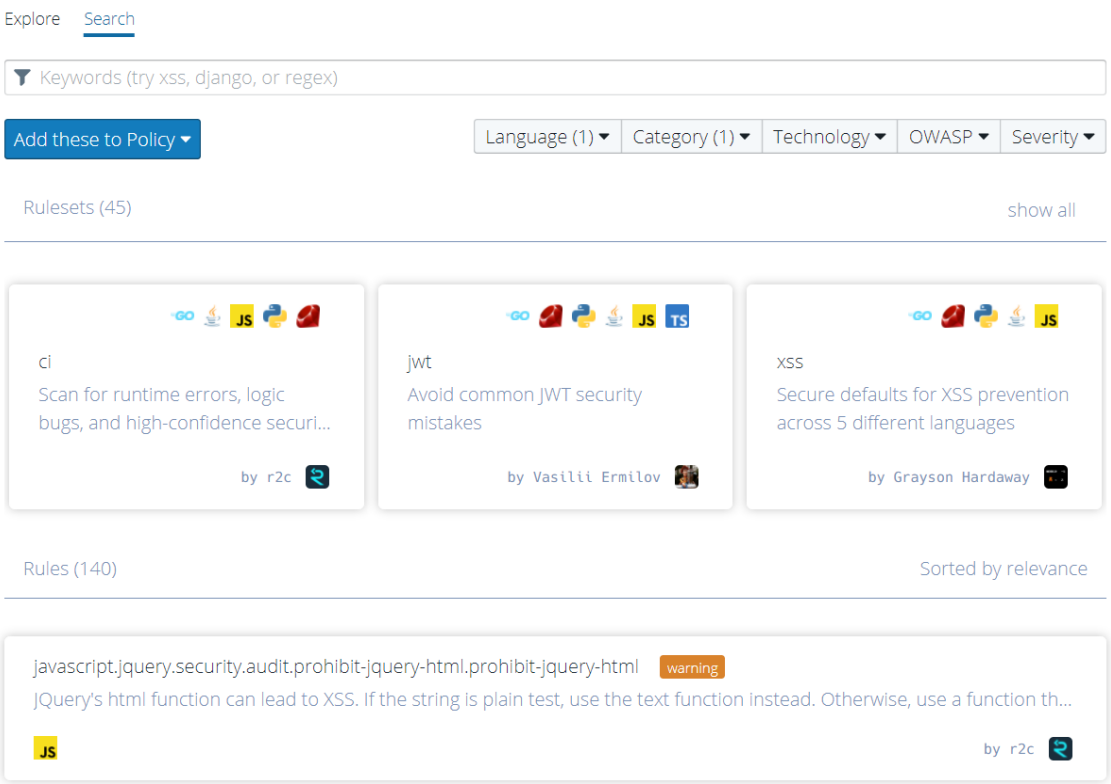

Figure 2. Search page of Semgrep's GUI showing 140 rules after filtering all the pre-written rules by the JavaScript language and security category

- Codacy:

  Used GUI with the automatically selected built-in tools. Figure 3 shows all the built-tools that were used by Codacy when applied to the Open OnDemand code.

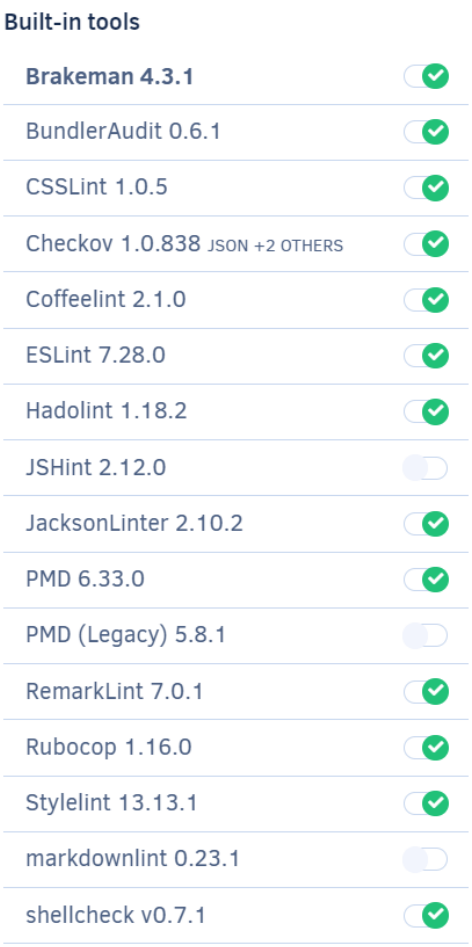

Figure 3. Built-in tools automatically selected by Codacy during the setup process for the Open OnDemand code

- SonarQube:

  Used GUI which was setup using Sonarqube's docker container. The repository was added manually by creating a token (as shown in Figure 4) and running the docker command provided in step 2 of 'Analyze your project'. This command is shown below.

```
sonar-scanner \
-Dsonar.projectKey=<key> \
-Dsonar.sources=. \
-Dsonar.host.url=http://localhost:9000 \
-Dsonar.login=<token>
```

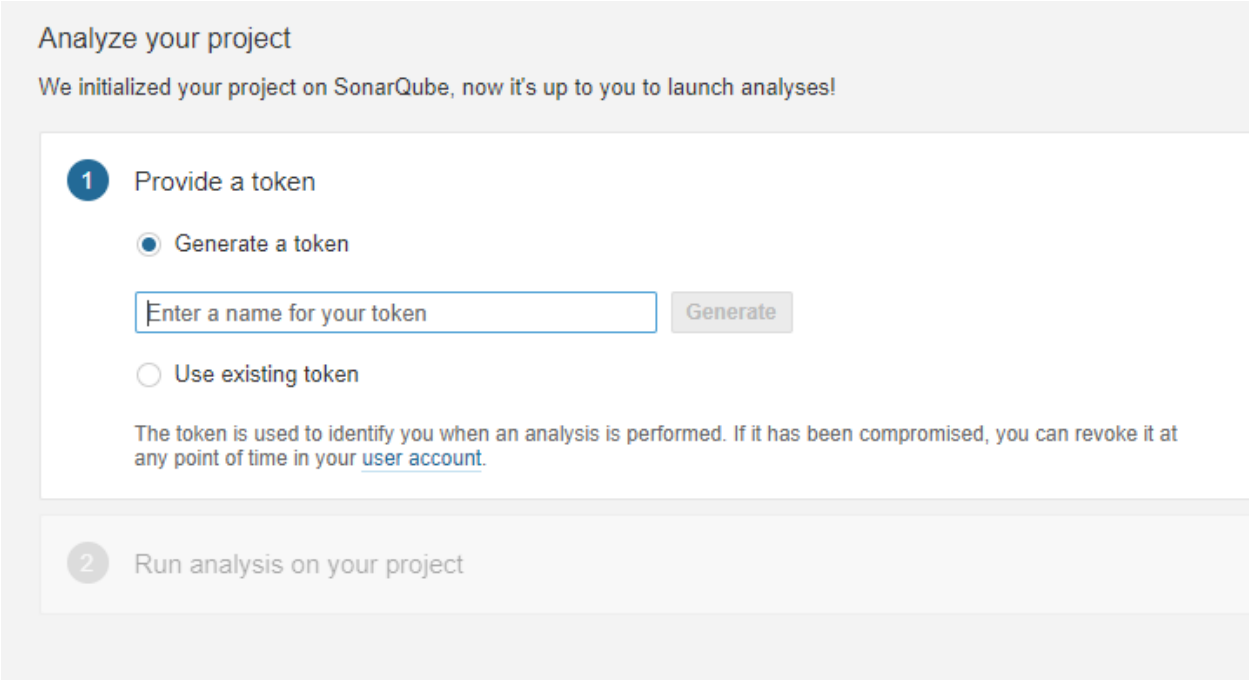

Figure 4. SonarQube's token creation step

- We looked at the Security Hotspots and Security Vulnerabilities to review the security issues found. Figure 5 shows the menu items and filters for the same.

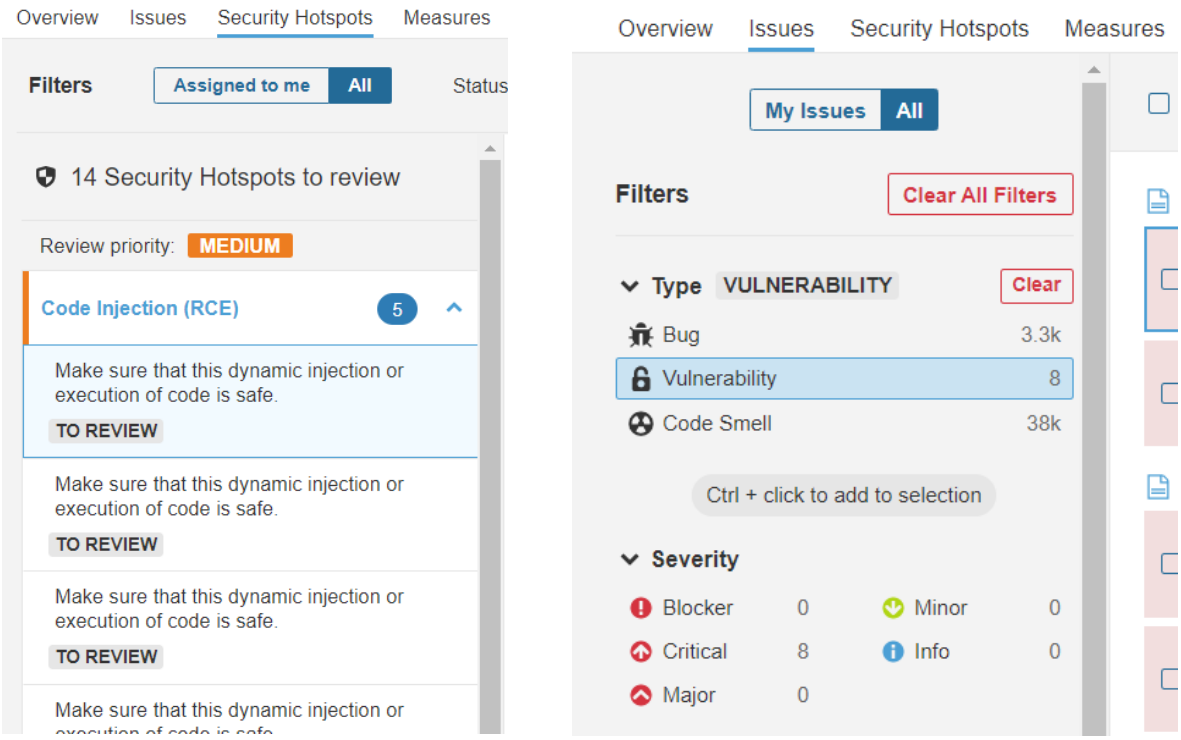

Figure 5. Security Hotspots (left) and Security Vulnerabilities (right) on the SonarQube GUI

3.2.5.b. Commands used for Ruby Tools

- Semgrep

```
docker run --rm -v "${PWD}:/src" returntocorp/semgrep --lang=rb --config=r/ruby
--json -o semgrep-test.json ./
```

- Brakeman

```
docker run -v "$(pwd)":./ presidentbeef/brakeman -o brakeman_results.html
```

- DawnScanner

  Due to the way DawnScanner operates, it was necessary to run it on each app with the commands shown below.

```
dawn --html --file dawn_scanner.html --disable-code-quality
--disable-code-style ./
dawn --html --file dawn_scanner_activejobs.html --disable-code-quality
--disable-code-style apps/activejobs/
dawn --html --file dawn_scanner_dashboard.html --disable-code-quality
--disable-code-style apps/dashboard/
dawn --html --file dawn_scanner_file-editor.html --disable-code-quality
--disable-code-style apps/file-editor/
dawn --html --file dawn_scanner_myjobs.html --disable-code-quality
--disable-code-style apps/myjobs/
dawn --html --file dawn_scanner_ood-portal-generator.html
--disable-code-quality --disable-code-style ood-portal-generator/
```

- DeepSource

  Used GUI and permitted the tool to access the GitHub repository. To share the Open OnDemand repository, a private copy was made on GitHub and shared with DeepSource.

3.2.6. Comparison of static analysis tools for JavaScript

	Semgrep	Semgrep (alternate ruleset)	SonarQube	Codacy
Total Issues (Style, Error-prone, and Security Issues)	48	887	~40,000	~20,000
Number of Security Issues	48	887	18	830
Number of files with Security Issues	18	181	11	206

Table 3. Comparison of the number of issues found by JavaScript tools when applied to the formatted Open OnDemand code at commit 4541d6b

We identified notable differences in the number and type of security issues reported by each tool when applied to the Open OnDemand code. Furthermore, we formatted this code in our analysis to account for the significant number of minified JavaScript files.

First, Semgrep and Codacy contain an overwhelming number of false positives. This problem can make understanding and resolving security issues frustrating and confusing. On the other

hand, SonarQube reports relatively fewer results and includes probable security issues. Additionally, it provides examples and helpful links to better understand the issues.

Appendix C (Tables C1 to C4) shows the security issues and their classifications from our assessment of JavaScript static analysis tools. Due to a large number of results from Codacy and Semgrep (with alternate ruleset), we could not evaluate and classify all of them; however, we included their most prominent security issues along with their classifications.

3.2.7. Comparison of Static Analysis Tools for Ruby

	Brakeman	DawnScanner	Semgrep	DeepSource
Total Issues (Style, Error-prone, and Security Issues)	24	9	67	5
Number of Security Issues	24	9	67	5
Number of files with Security Issues	12	5	49	5

Table 4. Comparison of the number of issues found by Ruby tools when applied to the Open OnDemand code at commit 4541d6b

Like with JavaScript tools, we identified notable differences in the number and type of security issues for Ruby tools. Additionally, each tool reports at least one probable security issue.

As before, Semgrep struggles to understand the context of the code and is difficult to evaluate. Despite reporting the highest number of security issues, most of these reports are false positives.

Appendix C (Tables C5 to C8) shows all the security issues and their classifications from our assessment of Ruby static analysis tools.

3.2.8. Pricing considerations

Semgrep, Brakeman, and DawnScanner are free and open-source tools. SonarQube, Codacy, and DeepSource are commercial tools that provide a free tier for open-source projects.

3.3. Vulnerability Disclosure Processes

During this engagement we identified several opportunities for Open OnDemand to improve their vulnerability disclosure processes.

3.3.1. Community interactions and expectations

Vulnerability reports may contain sensitive information and should not be disclosed publicly until users of the software are able to prevent exploitation on their systems. The existing practices noted in Section 2.3. introduce risk of premature disclosure. Although the ood-users mailing list is moderated by staff and messages must be approved before distribution to the public list, the opportunity still exists to distribute vulnerabilities to the list. This could occur inadvertently by an authorized user sending the wrong command to the list system, but it also introduces opportunities for disclosure due to misconfiguration of the mailing list, the mailing list host system, or even deliberate abuse by an insider. Open OnDemand can easily address this by changing their reporting processes such that initial reports are to be directed to Open OnDemand staff rather than a moderated public mailing list. Specific recommendations are provided in Section 4.3.1. of this report.

A patch to open source software will necessarily disclose technical details of the flaw being patched. As Open OnDemand is deployed at a wide variety of research institutions, such details of a vulnerability in Open OnDemand are valuable to attackers targeting research infrastructure. Research operations using Open OnDemand could benefit from advance notice through a non-public channel dedicated to security communications with this key set of stakeholders so that they can be prepared to apply necessary security patches as soon as possible. The Open OnDemand team currently does not have a formal capability to provide this advance notice, although the necessary community interaction mechanisms to do so, such as the Open OnDemand Discourse web forum, are available.

Open OnDemand would be well served by formally assigning responsibilities for vulnerability handling and ensuring that individuals given these responsibilities understand the relevant procedures. Both leadership and developers have responsibilities in this regard.[28] These are described in Section 3.3.2. and recommendations are described in 4.3.2. of this report.

Organizations involved in vulnerability research such as Google Project Zero[29] often establish policies and procedures both to ensure that developers have sufficient opportunity to address vulnerabilities and that users of software are made aware of issues which may affect them even if the developer fails to sufficiently address identified issues. This collection of practices is often referred to as responsible vulnerability disclosure and attempts to balance the needs of users

[28] Leadership responsibilities are discussed in additional detail in module 5 of the Introduction to Software Security series developed by Elisa Heymann, Barton P. Miller and Loren Kohnfelder. Module 5.7 in particular details responsibilities for managers. https://research.cs.wisc.edu/mist/SoftwareSecurityCourse/

[29] https://googleprojectzero.blogspot.com/p/about-project-zero.html

with developers in regard to vulnerabilities.[30] [31] Software development projects such as Open OnDemand should anticipate this need when establishing procedures for handling vulnerability reports.

3.3.2. Roles and responsibilities

In many cases, a single developer will be fully capable of handling a vulnerability report from receipt to remediation and reporting. In some cases however, a vulnerability may be complex, remediating it may require changes which would have significant impact on other software components, or it may affect in some way the operations of users. Having a single developer assigned this responsibility introduces risks of having insufficient personnel resources, availability, and attention to address reported vulnerabilities. Responsibility to fix the vulnerability should not be assigned only to a single individual with no fallback or accountability mechanism to ensure that remediation is completed. Open OnDemand will need to ensure a degree of flexibility both in personnel responsible for remediation and in roles.

Individuals with key roles will need to be in regular communication from the time a vulnerability report is received until the vulnerability is remediated. During the remediation process, leadership is responsible for ensuring that a capable developer is assigned to remediating a vulnerability and has sufficient resources to do so. Leadership is also responsible for assigning additional resources if required and ensuring that personnel assigned to the task are held accountable for its completion. In addition to producing the necessary fixes to remediate vulnerabilities, developers are responsible for initial triage, communication with the reporter of the vulnerability to confirm key details, and ensuring that leadership is aware of any additional resources which would be required to address the vulnerability report.

3.3.3. Procedures

Consistent procedures for changes, whether necessary to remediate vulnerabilities or for any other reason, help to ensure that changes made under time pressure do not introduce additional issues. During this engagement, Trusted CI and Open OnDemand developed

[30] Weulen Kranenbarg, M., Holt, T.J. & van der Ham, J. Don't shoot the messenger! A criminological and computer science perspective on coordinated vulnerability disclosure. Crime Sci 7, 16 (2018). https://doi.org/10.1186/s40163-018-0090-8

[31] H. Cavusoglu, H. Cavusoglu and S. Raghunathan, "Efficiency of Vulnerability Disclosure Mechanisms to Disseminate Vulnerability Knowledge," in IEEE Transactions on Software Engineering, vol. 33, no. 3, pp. 171-185, March 2007, doi: 10.1109/TSE.2007.26.

checklists to help developers to assess code changes for security impact more quickly and consistently. Checklists for assessing code changes can be found in Appendix F.

Although policies and procedures should be developed and codified to ensure consistency, they cannot be treated as static artifacts. Revision is an integral part of any policy's lifecycle.[32] Likewise, operational procedures and other less formal norms should be subject to revision where appropriate. Techniques, norms, and operational realities will change over time. Even well developed and current policies and procedures will have gaps which were not anticipated by their authors. Procedures will need to be iterated upon in order to account for this.

3.4. Procedural Improvements & Effectively Utilizing Assessment Outputs

3.4.1. GitHub pull request templates to capture important context by default

GitHub provides a means to set templates for pull requests.[33] This ensures that contributors will see what should be included in a pull request before submitting. This feature can be used not only to set expectations for what contributors should provide for descriptions but can also convey expected security norms for contributors. These norms may include security checklists, cautionary statements, or other guidance which helps contributors to the Open OnDemand project understand what is expected of them when submitting a pull request and what characteristics may cause their changes to be rejected.[34] [35]

3.4.2. Identify sensitive processes and catalog them in a software bill of materials

Identification of sensitive processes, as in step 4 of the FPVA process, allows Open OnDemand to prioritize which software repositories and components are most important to analyze for vulnerabilities. In addition, code changes to these should receive additional scrutiny to ensure that they are not introducing new vulnerabilities to the software. Trusted CI and Open OnDemand have produced a draft of a repository inventory for use as a software "bill of

[32] The policy lifecycle is discussed in more detail in Trusted CI's Framework Implementation Guide for Research Cyberinfrastructure Operators, Must 9: Policy. https://zenodo.org/record/4562447#.YDlDhuhKiUk
[33] GitHub provides descriptive instructions for creating these templates. https://docs.github.com/en/communities/using-templates-to-encourage-useful-issues-and-pull-requests/creating-a-pull-request-template-for-your-repository
[34] Established businesses built upon open source products such as Elastic provide such guidance to their communities as well. These may provide valuable examples when understanding what should go into such a template. https://www.elastic.co/guide/en/kibana/current/pr-review.html
[35] GitHub pull request templates are also being recommended in other domains as ways to incorporate security checklists into developer workflows. https://jemurai.com/2019/08/15/github-pr-checklist/

materials" which identifies repositories relevant to the project and includes provisions for repository owners, contact information, sensitive software components, and other key information.

Although it would be preferable to automate the production of the bill of materials and produce it in a format which could be readily queried by those deploying or integrating Open OnDemand software into their own operations, the resources required to implement this automation will be significantly higher. In addition, the recently released White House Executive Order on Improving the Nation's Cybersecurity[36] instructed the National Institute of Standards and Technology (NIST) to produce additional guidance regarding software supply chain security, including guidance specifically for software bills of materials. It is possible that the eventual guidance from NIST will substantially differ from the template created during this engagement and a significant effort would need to be undertaken to adhere to their guidance. Because of this, an exhaustive effort to produce a comprehensive software bill of materials may prove counterproductive until this future NIST guidance is published and can be reviewed for applicability. We believe that the template we have produced represents a reasonable starting point which balances effort requirements and utility given these limitations.

3.4.3. Consistent change processes

During this engagement, Trusted CI and Open OnDemand developed checklists for use when assessing code changes for security issues. These can be found in Appendix F. The practices incorporated into these checklists will have the most positive impact if they are followed consistently. Failure to perform these checks during high pressure situations such as when addressing a vulnerability or making an important change introduces additional risk that something will be missed and may ultimately lead to additional vulnerabilities being introduced.

4. Recommendations

4.1. Dependency Analysis Tools

Status: Open OnDemand currently uses Dependabot to search for vulnerabilities in the project's dependencies. An analysis of Dependabot's results revealed that the number of meaningful

[36] Relevant details are in section 4.e. of the May 12, 2021 Executive Order on Improving the Nation's Cybersecurity: https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/

vulnerabilities found is significantly smaller than the number reported by Snyk and OWASP Dependency-Check.

Recommendation: While Dependabot's GitHub integration is highly convenient, the relative sparsity of results compared to other tools represent a significant risk to the overall system. We recommend using Snyk and OWASP Dependency-Check in place of Dependabot. We also recommend using the command-line integration of these tools where possible, rather than the website interfaces, due to differences in the results reported, as detailed in Section 3.1.3.

4.2. Static Analysis Tools

Status: Open OnDemand currently does not use any static analysis tools for JavaScript or Ruby.

4.2.1. Recommendation for JavaScript

Recommendation: For JavaScript, we recommend using SonarQube, as it provides concise reports, has flexible configuration options, feature-full GUI experience, and performs a more in-depth analysis than just simple pattern matching. We would discourage the use of Semgrep and Codacy as they report the highest number of issues of all the tools we assessed. We note that their results are primarily false positives. Moreover, the overwhelming number of results can be frustrating and confusing to resolve.

4.2.2. Recommendation for Ruby

Recommendation: For Ruby we recommend using a combination of tools: Brakeman, DawnScanner, and DeepSource as no single tool reports all the probable security issues when applied to the Open OnDemand code. We do not recommend Semgrep even though it reports two probable security issues, because more than 80% of its results are false positives (shown in Table C7). The overwhelming number of false positives can be frustrating and confusing to resolve.

4.3. Addressing Gaps in Vulnerability Disclosure Processes

4.3.1. Community interactions and expectations

Status: Open OnDemand doesn’t provide a dedicated workflow for reporting vulnerabilities.

4.3.1.a. Aligning with responsible disclosure practices.

Recommendation: We recommend that Open OnDemand adopt a vulnerability reporting process which supports responsible disclosure practices for security researchers. Open OnDemand should establish clear communication lines between reporters and those responsible for vulnerability remediation and establish and communicate both expected remediation timelines to the community and that these expected timelines are not applicable in all cases. We recommend that this expected remediation period be set initially at two weeks.

4.3.1.b. Communications

Recommendation: We also recommend that Open OnDemand provide a complete description of the Open OnDemand project's expectations for vulnerability reporting be on the Open OnDemand website, in documentation, and directly on the relevant security policy pages in github repositories. We have produced sample text which can be reused for this purpose, found in Appendix E.

4.3.1.c. Confidentiality

Recommendation: To improve handling of sensitive security communications, Trusted CI recommends that the Open OnDemand team establish a separate security mailing list for vulnerability reporting and other security concerns. The recipients of these messages should include, at minimum, one person from Ohio Supercomputer Center's leadership and one software developer from the Open OnDemand team.

In addition to a dedicated communication channel for reports to be received, Open OnDemand may need a method to distribute sensitive vulnerability information to security and operational staff at facilities using their software prior to public release of a patch. As Open OnDemand operates a web forum for users, it may be appropriate for Open OnDemand to set up an access controlled section of this forum for this purpose.

4.3.1.d. Report assignment

Recommendation: When a vulnerability report is received it should be assigned to a developer and that individual should be responsible for remediation and response, including communications with the reporter of the issue. We recommend that vulnerability reports are given high priority and investigated immediately by the developer assigned to the reported issue.

4.3.1.e. Report handling consistency

Recommendation: Open OnDemand should expect that team members will occasionally receive reports and relevant security information directly. When this occurs, the team member should forward the report and all relevant information to the security reporting mailing list. From that point, the same reporting and remediation procedures should be followed.

4.3.2. Roles and responsibilities

Status: Open OnDemand doesn't have formal roles and responsibilities for vulnerability remediation.

Recommendation: We recommend that Open OnDemand identify, at minimum, two individuals who can be primarily responsible for managing vulnerability reports. One of these individuals should be a member of the Ohio Supercomputer Center's team with sufficient authority to assign resources if necessary, and one should be a developer with dedicated effort allocation assigned to the Open OnDemand project. Individuals identified should be recipients of the mailing list as described in Section 3.3.1. and responsible for triage and remediation of vulnerability reports. Responsibilities for these individuals should reflect the needs described in Section 3.3.2. of this report.

4.3.3. Procedures

Status: Open OnDemand lacks a formal vulnerability disclosure procedure.

Recommendation: We recommend that Open OnDemand adopt responsible disclosure procedures[37] as described in Section 3.3.1. We recommend that the procedures codified in the checklists developed during this engagement be followed consistently, including when assessing changes intended to remediate vulnerabilities or address other high priority issues.

In addition to the creation of checklists, OOD should refine their procedures over time by reviewing and comparing them to guides such as the OWASP Vulnerability Management Guide. [38] The guide documents the detection, reporting, and remediation cycles of vulnerability management.

---

[37] https://cheatsheetseries.owasp.org/cheatsheets/Vulnerability_Disclosure_Cheat_Sheet.html#responsible-or-coordinated-disclosure

[38] https://owasp.org/www-project-vulnerability-management-guide/

4.4. Procedural Improvements & Effectively Utilizing Assessment Outputs

4.4.1. Develop a GitHub pull request template to capture important context

Status: Open OnDemand could use pull request templates to convey expected security norms to contributors.

Recommendation: We recommend that Open OnDemand develop pull request templates for Github as an additional means to set expectations about what information should be included in pull requests. These templates should incorporate relevant security guidance for contributors as discussed in Section 3.4.1.

4.4.2. Identify sensitive processes and catalog them in a software bill of materials

Status: Open OnDemand does not have a comprehensive inventory of software components.

Recommendation: We recommend that Open OnDemand fill out the template bill of materials, prioritizing the core software components, and continue to update it to ensure that it can be used to quickly identify when a change implicates a sensitive software component. Open OnDemand should expect to evolve this document over time and may find that a spreadsheet doesn't provide necessary functionality and need to adopt a different format. As this topic is actively being explored by NIST, it may also be worth exploring alignment with future NIST recommendations for developing a software bill of materials. Open OnDemand may find it appropriate in the future to incorporate improvements such as automating the inclusion of relevant information from manifest files or incorporating relevant outputs from build tools to reduce the workload required to keep this information up to date.

4.4.3. Consistent change processes

Status: During this engagement we developed checklists which describe security procedures. In order to ensure that they are effective for their intended purpose they should be followed consistently.

Recommendation: Trusted CI recommends the checklists discussed in Section 3.4.3. (Appendix F) be incorporated into formal documentation, incorporated into relevant pull request templates as discussed in Section 3.4.1, and training on their use be incorporated into onboarding procedures for new software developers assigned to the project. We recommend that Open OnDemand adhere to these consistently regardless of whether the source of a

proposed change is a member of the project, an external contributor, if the change is addressing an urgent issue, or other situations which may arise.

5. Conclusion

On behalf of Trusted CI, the engagement team would like to thank the Open OnDemand project team for its openness to the engagement process and hard work during the engagement period. We are hopeful that the progress made in developing Open OnDemand expertise with conducting vulnerability assessments using the FPVA methodology and related security-specific checklists enable OOD to better maintain the security of their software and to identify and mitigate future vulnerabilities. As the OOD team gains more experience conducting assessments using the FVPA methodology, the Trusted CI team remains available to provide additional guidance, support, and expertise.

Appendix A: Lines of Code Analysis at version v1.8.19

Language	files	blank	comment	code
JavaScript	525	10632	16273	64403
Ruby	550	6645	7981	28212
SVG	46	31	36	16139
CSS	39	456	304	11652
Markdown	94	3355	0	8585
JSON	74	0	0	6547
reStructuredText	129	5010	5146	5946
ERB	131	443	24	4648
YAML	191	476	1155	4607
Bourne Shell	80	371	929	1678
XML	7	124	0	1551
HTML	29	134	55	1168
Sass	25	221	117	1079
Go	2	62	24	927
Puppet	17	104	334	915
Python	15	203	275	850
CoffeeScript	7	65	53	476
Lua	10	97	144	382
Freemarker Template	6	32	0	364
make	4	49	5	263
DOS Batch	1	36	2	243
diff	8	24	122	231
Handlebars	9	13	5	212
Bourne Again Shell	20	68	47	194
Dockerfile	9	38	13	152
C	4	27	35	94
Mustache	4	14	0	69
R	1	25	8	67
INI	1	1	0	38
TeX	1	1	0	36
Expect	1	2	10	9
CUDA	1	0	0	6
MATLAB	1	0	2	2
Windows Resource File	1	0	0	2
SUM:	2,043	28,759	33,099	161,747

Appendix B: Comparison of Dependency Tools

Reported Severity Legend	
High	The tool reported this vulnerability as a high-risk
Medium	The tool reported this vulnerability as a medium-risk
Low	The tool reported this vulnerability as a low-risk
White (no highlight)	The tool did not provide a severity for this vulnerability

Results from dependency tools when applied to the Open OnDemand code at version v1.8.19

	Package Name	Dependabot	Snyk (CLI)	OWASP Dependency-Check	Depfu
1	actioncable		CWE-200		
2	activerecord	CVE-2021-22880	CVE-2021-22880		CVE-2021-22880
3	bootstrap				CVE-2018-14041
4	braces			CWE-400	
				NPM-786	
5	clean-css		CWE-185		
6	cloudcmd		CWE-79		
7	color-string		CWE-400		
8	debug		CVE-2017-16137	CWE-400	
				NPM-534	
9	diff			NPM-1631	
10	engine.io		CVE-2020-36048		
11	glob-parent		CVE-2020-28469		
12	growl			CWE-78	
				NPM-146	
13	handlebars		CVE-2021-23369	CWE-471	
				NPM-755	
14	hosted-git-info		CVE-2021-23362		
15	http-auth		CWE-294		
16	is-svg	CVE-2021-28092	CVE-2021-28092		

17	jquery			CVE-2007-2379	
				CVE-2012-6708	
				CVE-2015-9251	
				CVE-2019-11358	
				CVE-2020-11022	
				CVE-2020-11023	
18	lodash		CVE-2020-28500	CVE-2018-16487	
			CVE-2021-23337	CVE-2018-3721	
			CWE-400	CVE-2019-1010266	
				CVE-2019-10744	
				CVE-2020-28500	
				CVE-2020-28500	
				CVE-2020-8203	
				CVE-2020-8203	
				CVE-2021-23337	
				CWE-506	
				CWE-770	
				NPM-1065	
				NPM-1523	
				NPM-577	
				NPM-782	
19	lodash.template		CVE-2021-23337		
20	markdown-it		CWE-400		
21	minimist			CWE-94	
				NPM-1179	
22	ms		CWE-400		
23	negotiator		CVE-2016-10539		
24	node-sass		CVE-2018-11694		
			CVE-2018-11697		
			CVE-2018-11698		
			CVE-2018-19797		
			CVE-2018-19827		
			CVE-2018-19839		

			CVE-2018-20190		
			CVE-2018-20821		
			CVE-2018-20822		
			CVE-2019-18797		
			CVE-2019-18798		
			CVE-2019-18799		
			CVE-2019-6283		
			CVE-2019-6284		
			CVE-2019-6286		
			CVE-2020-24025		
25	parsejson		CVE-2017-16113	CWE-400	
				NPM-528	
26	ponse		CWE-22		
27	postcss		CVE-2021-23368		
28	rails		CVE-2021-22880		
29	socket.io	CVE-2020-28481	CVE-2020-28481	CVE-2020-28481	CVE-2020-28481
				NPM-1609	
30	socket.io-parser		CVE-2020-36049		
31	ssri		CVE-2021-27290		
32	ws		CVE-2016-10542	NPM-550	
			CWE-330		
			CWE-400		
33	xmlhttprequest-ssl		CVE-2020-28502		

Appendix C: Comparison of Static Analysis Tools

Classification Legend	
Probable	The reported issue appears to be a probable issue, though was confirmed without a running instance
Unable to Determine	The reported issue was not easily evaluated, and would warrant further investigation
False Positive	It was determined by a manual review of the code that the issue reported is a false positive

Subset of the results from static analysis tools for JavaScript when applied to the formatted Open Ondemand code at commit 4541d6b.

Issue Type	File	Reported Severity
Code Injection	apps/file-editor/public/ace/1.2.6/ext-old_ie.js:278	Medium
	apps/file-editor/public/ace/1.2.6/worker-coffee.js:16598	Medium
	apps/file-editor/public/ace/1.2.6/worker-coffee.js:16606	Medium
	apps/file-editor/public/ace/1.2.6/worker-css.js:1680	Medium
	apps/file-editor/public/ace/1.2.6/worker-javascript.js:9556	Medium
Fingerprinting	apps/shell/app.js:51	Low
Insecure HTTP Protocol	apps/file-editor/public/ace/1.2.6/worker-html.js:3626	Low
	apps/shell/public/javascripts/hterm_all_1.85.js:16458	Low
Weak Cryptography	apps/myjobs/app/assets/javascripts/joyride.js:677	Medium
	apps/shell/public/javascripts/hterm_all_1.85.js:1833	Medium
cross-origin communication verification	apps/dashboard/public/noVNC-1.1.0/app/error-handler.js:65	Critical
	apps/dashboard/public/noVNC-1.1.0/app/error-handler.js:68	Critical
	apps/dashboard/public/noVNC-1.1.0/core/util/events.js:109	Critical
	apps/dashboard/public/noVNC-1.1.0/core/util/events.js:110	Critical
	apps/dashboard/public/noVNC-1.1.0/legacy/app.js:13832	Critical
	apps/dashboard/public/noVNC-1.1.0/legacy/app.js:13833	Critical
	apps/file-editor/public/ace/1.2.6/worker-html.js:10107	Critical
	apps/file-editor/public/ace/1.2.6/worker-html.js:10125	Critical

Table C1. Results from SonarQube. Showing 18 of 18 security issues at code commit 4541d6b

Issue Type	File	Reported Severity
Detected non-literal in eval	apps/file-editor/public/ace/1.2.6/worker-coffee.js:16598	WARNING
	apps/file-editor/public/ace/1.2.6/worker-coffee.js:16606	WARNING
	apps/file-editor/public/ace/1.2.6/worker-css.js:1680	WARNING
	apps/file-editor/public/ace/1.2.6/worker-javascript.js:9556	WARNING
	apps/file-editor/public/ace/1.2.6/ext-old_ie.js:279	WARNING
	apps/files/lib/cloudcmd/lib/client/edit.js:139	WARNING
	apps/files/lib/cloudcmd/modules/menu/menu-io.js:170	WARNING
	apps/files/lib/cloudcmd/modules/menu/menu-io.js:179	WARNING
	apps/files/lib/cloudcmd/modules/menu/menu-io.js:180	WARNING
	apps/files/lib/cloudcmd/modules/menu/menu-io.js:236	WARNING
	apps/files/lib/cloudcmd/modules/menu/menu-io.js:245	WARNING
	apps/files/lib/cloudcmd/modules/execon/lib/exec.js:106	WARNING
	apps/files/lib/cloudcmd/modules/execon/lib/exec.js:119	WARNING
	apps/files/lib/cloudcmd/modules/execon/lib/exec.js:164	WARNING
	apps/files/lib/cloudcmd/modules/execon/lib/exec.js:174	WARNING
	apps/files/lib/cloudcmd/modules/execon/lib/exec.js:65	WARNING
	apps/files/lib/cloudcmd/modules/execon/lib/exec.js:66	WARNING
	apps/files/lib/cloudcmd/lib/client/storage.js:37	WARNING
	apps/files/lib/cloudcmd/lib/client/storage.js:51	WARNING
	apps/files/lib/cloudcmd/lib/client/storage.js:70	WARNING
	apps/files/lib/cloudcmd/lib/client/storage.js:80	WARNING
	apps/files/lib/cloudcmd/lib/client/storage.js:90	WARNING
Detected non-literal in require	apps/files/lib/cloudcmd/test/lib/cloudfunc.js:7	WARNING
	apps/files/lib/cloudcmd/test/lib/cloudfunc.js:8	WARNING
	apps/files/lib/cloudcmd/lib/server/route.js:20	WARNING
	apps/files/lib/cloudcmd/lib/server/route.js:21	WARNING
	apps/files/lib/cloudcmd/lib/server/route.js:22	WARNING
	apps/files/lib/cloudcmd/lib/server/route.js:31	WARNING
	apps/files/lib/cloudcmd/lib/server/config.js:8	WARNING

	apps/files/lib/cloudcmd/lib/server/config.js:9	WARNING
	apps/files/lib/cloudcmd/lib/cloudcmd.js:10	WARNING
	apps/files/lib/cloudcmd/lib/cloudcmd.js:11	WARNING
	apps/files/lib/cloudcmd/lib/cloudcmd.js:12	WARNING
	apps/files/lib/cloudcmd/lib/cloudcmd.js:13	WARNING
	apps/files/lib/cloudcmd/lib/cloudcmd.js:14	WARNING
	apps/files/lib/cloudcmd/lib/cloudcmd.js:8	WARNING
	apps/files/lib/cloudcmd/lib/cloudcmd.js:9	WARNING
	apps/file-editor/public/ace/1.2.6/worker-coffee.js:160	WARNING
	apps/file-editor/public/ace/1.2.6/worker-css.js:160	WARNING
	apps/file-editor/public/ace/1.2.6/worker-html.js:160	WARNING
	apps/file-editor/public/ace/1.2.6/worker-javascript.js:160	WARNING
	apps/file-editor/public/ace/1.2.6/worker-json.js:160	WARNING
	apps/file-editor/public/ace/1.2.6/worker-lua.js:160	WARNING
	apps/file-editor/public/ace/1.2.6/worker-php.js:160	WARNING
	apps/file-editor/public/ace/1.2.6/worker-xml.js:160	WARNING
	apps/files/lib/cloudcmd/bin/cloudcmd.js:10	WARNING
	apps/files/lib/cloudcmd/bin/cloudcmd.js:11	WARNING
	apps/files/lib/cloudcmd/bin/cloudcmd.js:163	WARNING

Table C2. Results from Semgrep. Showing 48 of 48 security issues at code commit 4541d6b

Issue Type	File	Reported Severity
raw-html-concat	apps/file-editor/public/ace/1.2.6/ace.js:15378	WARNING
	apps/dashboard/public/noVNC-1.1.0/app/webutil.js:94	WARNING
	apps/dashboard/public/noVNC-1.1.0/legacy/app.js:10089	WARNING
	apps/file-editor/public/ace/1.2.6/mode-html_ruby.js:2208	WARNING
	apps/files/lib/cloudcmd/modules/smalltalk/src/smalltalk.js:35	WARNING
	apps/files/lib/cloudcmd/lib/client/files.js:79	WARNING
	apps/shell/public/javascripts/hterm_all_1.85.js:2227	WARNING
	apps/dashboard/public/noVNC-1.1.0/core/rfb.js:789	WARNING
	apps/dashboard/public/noVNC-1.1.0/app/ui.js:830	WARNING
	apps/files/lib/cloudcmd/modules/fancybox/source/jquery.fancybox.js:984	WARNING
jquery-insecure-selector	apps/activejobs/app/assets/javascripts/application.js:207	WARNING
	apps/files/lib/cloudcmd/lib/client/jstree/jstree.js:8308	WARNING
	apps/files/lib/cloudcmd/lib/client/polyfill.js:54	WARNING
	apps/dashboard/app/assets/javascripts/application.js:73	WARNING
insecure-document-method	apps/myjobs/app/assets/javascripts/modernizr.mq.js:37	WARNING
insecure-innerhtml	apps/file-editor/public/ace/1.2.6/ace.js:11538	WARNING
insufficient-postmessage-origin-validation	apps/dashboard/public/noVNC-1.1.0/core/input/keyboard.js:358	WARNING

Table C3. Results from Semgrep with an alternate ruleset. Showing 17 of 887 security issues at code commit 4541d6b

Issue Type	File	Reported Severity
Generic Object Injection Sink	apps/file-editor/public/ace/1.2.6/worker-xml.js:225	None Provided
	apps/files/lib/cloudcmd/lib/client/config.js:233	None Provided
	apps/files/lib/findit2/index.js:57	None Provided
	apps/file-editor/public/ace/1.2.6/mode-r.js:111	None Provided
	apps/file-editor/public/ace/1.2.6/mode-clojure.js:156	None Provided
	apps/file-editor/public/ace/1.2.6/mode-makefile.js:304	None Provided
	apps/file-editor/public/ace/1.2.6/mode-logiql.js:121	None Provided
	apps/file-editor/public/ace/1.2.6/mode-pascal.js:187	None Provided
Found non-literal in RegExp Constructor	apps/file-editor/public/ace/1.2.6/mode-abc.js:139	None Provided
	apps/file-editor/public/ace/1.2.6/mode-applescript.js:89	None Provided
	apps/file-editor/public/ace/1.2.6/mode-gobstones.js:525	None Provided
	apps/file-editor/public/ace/1.2.6/mode-live_script.js:552	None Provided
Unsafe Regular Expression	apps/file-editor/public/ace/1.2.6/mode-gobstones.js:362	None Provided
	apps/file-editor/public/ace/1.2.6/mode-rust.js:136	None Provided
Variable Assigned to Object Injection Sink	apps/file-editor/public/ace/1.2.6/ext-language_tools.js:263	None Provided
Function Call Object Injection Sink	apps/shell/public/javascripts/ood_shell.1.js:28	None Provided
Found non-literal argument in require	apps/files/lib/cloudcmd/lib/server.js:8	None Provided

Table C4. Results from Codacy. Showing 17 of 883 security issues at code commit 4541d6b

All results from static analysis tools for Ruby when applied to the Open Ondemand code at commit 4541d6b.

Issue Type	File	Reported Severity
Command Injection	apps/dashboard/app/controllers/products_controller.rb:119	None Provided
	apps/dashboard/app/controllers/products_controller.rb:84	None Provided
	apps/dashboard/app/models/batch_connect/session.rb:230	None Provided
	apps/myjobs/app/models/filesystem.rb:73	None Provided
	nginx_stage/lib/nginx_stage/socket_file.rb:48	None Provided
	nginx_stage/lib/nginx_stage/user.rb:70	None Provided
	ood-portal-generator/lib/ood_portal_generator/application.rb:126	None Provided
File Access	apps/dashboard/app/controllers/products_controller.rb:116	None Provided
	apps/myjobs/app/controllers/templates_controller.rb:37	None Provided
	apps/myjobs/app/controllers/templates_controller.rb:86	None Provided
Mass Assignment	apps/dashboard/app/controllers/permissions_controller.rb:56	None Provided
	apps/dashboard/app/controllers/products_controller.rb:143	None Provided
	apps/myjobs/app/controllers/workflows_controller.rb:219	None Provided
RCE	apps/activejobs/config/initializers/cookies_serializer.rb:5	Warning
SQL Injection	apps/dashboard/app/controllers/permissions_controller.rb:14	None Provided
	apps/dashboard/app/controllers/permissions_controller.rb:23	None Provided
	apps/dashboard/app/controllers/permissions_controller.rb:42	None Provided
	apps/dashboard/app/controllers/permissions_controller.rb:6	None Provided
	apps/dashboard/app/controllers/products_controller.rb:113	None Provided
	apps/dashboard/app/controllers/products_controller.rb:20	None Provided
	apps/dashboard/app/controllers/products_controller.rb:39	None Provided
Template Injection	apps/dashboard/app/controllers/products_controller.rb:63	None Provided
	apps/dashboard/app/models/motd_formatter_markdown_erb.rb:7	Warning
	apps/dashboard/app/models/motd_formatter_plaintext_erb.rb:7	Warning

Table C5. Results from Brakeman. Showing 24 of 24 security issues at code commit 4541d6b

Issue Type	File	Reported Severity
Security Header	apps/file-editor	Warning
	apps/activejobs	Info
	apps/dashboard	Info
	apps/myjobs	Info
Session Management	apps/activejobs/config/initializers/session_store.rb	Info
	apps/dashboard/config/initializers/session_store.rb	Info
	apps/file-editor/config/initializers/session_store.rb	Info
	apps/myjobs/config/initializers/session_store.rb	Info

Table C6. Results from DawnScanner. Showing 8 of 8 security issues at code commit 4541d6b

Issue Type	File	Reported Severity
content_tag()	apps/dashboard/app/helpers/application_helper.rb:47	Warning
	apps/dashboard/app/helpers/batch_connect/session_contexts_helper.rb:26,30...	Warning
	apps/dashboard/app/views/apps/_app.html.erb:20	Warning
	apps/dashboard/app/views/dashboard/_motd_rss.html.erb:6,8	Warning
	apps/dashboard/app/views/layouts/nav/_group.html.erb:5,6,19	Warning
	apps/dashboard/app/views/products/_form_git.html.erb:3	Warning
	apps/dashboard/app/views/products/_form_reset_git.html.erb:18	Warning
	apps/myjobs/app/views/layouts/application.html.erb:46,47,70	Warning
	apps/myjobs/app/views/workflows/index.html.erb:90,93	Warning
html-safe	apps/activejobs/app/views/jobs/_extended_panel.html.erb:50,51,52	Warning
	apps/activejobs/app/views/jobs/index.html.erb:10,55,56	Warning
	apps/activejobs/app/views/layouts/application.html.erb:39,41	Warning
	apps/dashboard/app/views/batch_connect/session_contexts/new.html.erb:76	Warning
	apps/dashboard/app/views/batch_connect/sessions/connections/_custom.html.erb:1	Warning
	apps/dashboard/app/views/dashboard/_motd_markdown.html.erb:3	Warning
	apps/dashboard/app/views/dashboard/_motd_rss.html.erb:8	Warning
	apps/dashboard/app/views/layouts/application.html.erb:28	Warning
	apps/dashboard/app/views/products/_form_git.html.erb:2,3	Warning
	apps/dashboard/app/views/products/_form_manifest.html.erb:10	Warning
	apps/dashboard/app/views/products/_form_reset_git.html.erb:18	Warning
	apps/dashboard/app/views/products/_product.html.erb:18	Warning
	apps/dashboard/app/views/products/show.html.erb:151	Warning
	apps/file-editor/app/views/pages/index.html.erb:20,23	Warning
	apps/myjobs/app/views/layouts/application.html.erb:34,36,51	Warning
	apps/myjobs/app/views/workflows/_form.html.erb:26	Warning
Link To	apps/dashboard/app/views/permissions/_permission.html.erb:3	Warning

	apps/dashboard/app/views/permissions/index.html.erb:3,6	Warning
	apps/dashboard/app/views/permissions/new.html.erb:7	Warning
	apps/dashboard/app/views/products/_breadcrumbs.html.erb:12	Warning
	apps/dashboard/app/views/products/_new_choose_method.html.erb:14	Warning
	apps/dashboard/app/views/products/_product.html.erb:8,47	Warning
	apps/dashboard/app/views/products/edit.html.erb:2,24	Warning
	apps/dashboard/app/views/products/new_from_git_remote.html.erb:2,17	Warning
	apps/dashboard/app/views/products/show.html.erb:66,73,74,96,103	Warning
	apps/myjobs/app/views/workflows/show.html.erb:18,31	Warning
Manual Template Creation	apps/dashboard/app/models/announcement.rb:43	Warning
	apps/dashboard/app/models/batch_connect/app.rb:306	Warning
	apps/dashboard/app/models/batch_connect/session.rb:62,526	Warning
	nginx_stage/lib/nginx_stage/generator.rb:136	Warning
	ood-portal-generator/lib/ood_portal_generator/view.rb:128	Warning
Mass Assignment	apps/myjobs/app/controllers/templates_controller.rb:14,99	Warning
MD5	apps/dashboard/app/models/batch_connect/session.rb:508	Warning
Raw Markdown	apps/dashboard/app/views…ts/application.html.erb:61	Warning
	apps/dashboard/app/views/dashboard/_motd_osc.html.erb:8	Warning
RCE	apps/activejobs/config/initializers/cookies_serializer.rb:5	Warning
Ruby Eval	nginx_stage/lib/nginx_stage/generator.rb:93	Warning
	nginx_stage/spec/generators/pun_config_generator_spec.rb:43	Warning
	nginx_stage/spec/generators/pun_config_generator_spec.rb:59,69,75,86	Warning
SHA-1	ood-portal-generator/lib/ood_portal_generator/view.rb:98	Warning
Template Injection	apps/dashboard/app/models/motd_formatter_markdown_erb.rb:7	Warning
	apps/dashboard/app/models/motd_formatter_plaintext_erb.rb:7	Warning
Unsafe Deserialization	ood-portal-generator/spec/application_spec.rb:57	Warning
var in script	apps/activejobs/app/views/layouts/application.html.erb:10	Warning
	apps/file-editor/app/views/pages/index.html.erb:42,43,46,47	Warning

	apps/myjobs/app/views/layouts/application.html.erb:11,61	Warning
var-href	apps/activejobs/app/views/jobs/_job_details_node_view.html.erb:15	Warning
	apps/activejobs/app/views/layouts/application.html.erb:39,41	Warning
	apps/dashboard/app/views/dashboard/_motd_rss.html.erb:7	Warning
	apps/dashboard/app/views/errors/internal_server_error.html.erb:5	Warning
	apps/dashboard/app/views/layouts/application.html.erb:26,28	Warning
	apps/dashboard/app/views/layouts/nav/_all_apps.html.erb:2	Warning
	apps/dashboard/app/views/layouts/nav/_sessions.html.erb:2	Warning
	apps/dashboard/app/views/products/_form_icon.html.erb:52,76	Warning
	apps/myjobs/app/views/layouts/application.html.erb:34,36	Warning
	apps/myjobs/app/views/workflows/index.html.erb:16,19,21	Warning
	apps/myjobs/app/views/workflows/new.html.erb:7	Warning

Table C7. Results from Semgrep. Showing 64 of 64 security issues at code commit 4541d6b

Issue Type	File	Reported Severity
Kernel#open	apps/dashboard/app/models/balance.rb:15	Serious
	apps/dashboard/app/models/motd_file.rb:63	Serious
	apps/dashboard/app/models/quota.rb:20	Serious
YAML.load	apps/dashboard/app/apps/manifest.rb	Warning
	ood-portal-generator/spec/application_spec.rb	Warning

Table C8. Results from DawnScanner. Showing 5 of 5 security issues at code commit 4541d6b

Appendix D: Architectural and Resource Diagrams

The OnDemand team identified eight ‘flows’ that constitute the functions OnDemand can perform. For each flow, one diagram was created and is shown in the following sections. The eight flows are:

- Authentication Flow
- Dashboard Flow
- Dev App Flow
- Linux Host Adapter Flow
- NGINX Cron Job Flow
- Shared App Flow
- Shell Session Flow
- VNC App Flow

Authentication Flow

The authentication flow shows the series of requests and responses that are involved in authenticating a user through an identity provider. It is assumed in all other diagrams that the user is already authenticated, otherwise this flow would occur before further interactions with the webserver.

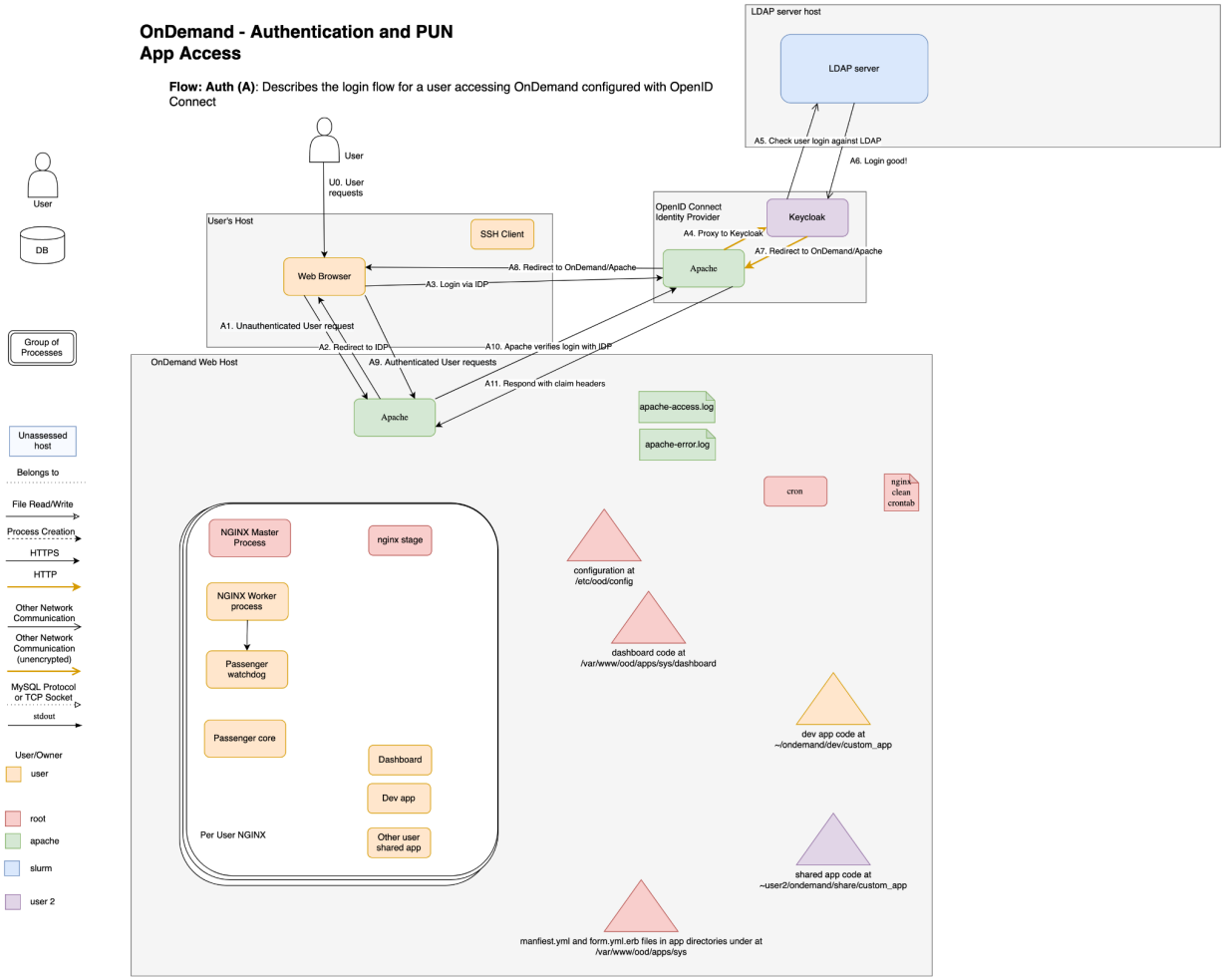

Figure D1. Authentication Flow for Open OnDemand version 1.8.19

Dashboard App Flow

The dashboard app flow shows the interactions that occur between Open OnDemand processes when a user connects to the OnDemand dashboard after logging in.

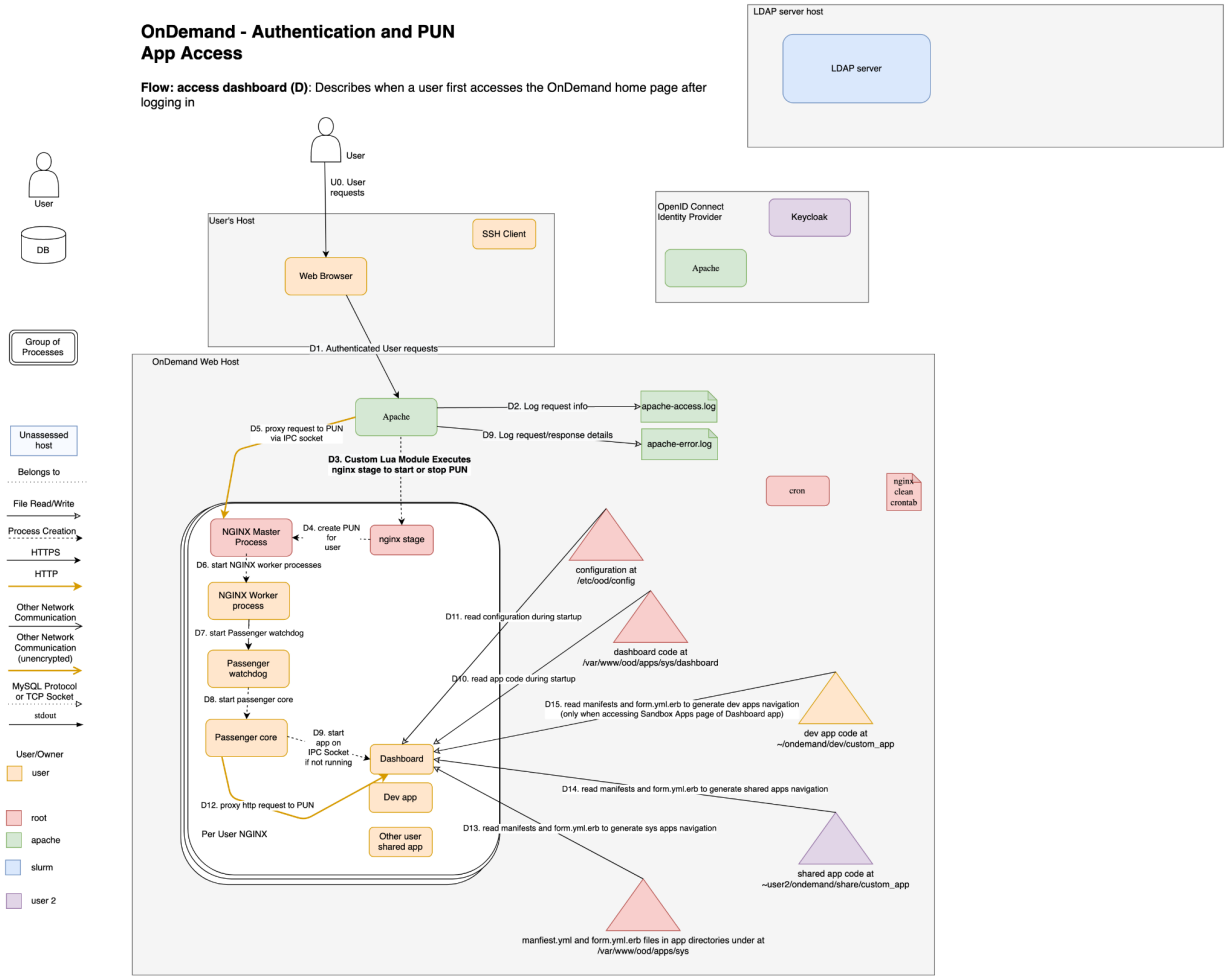

Figure D2. Dashboard App Flow for Open OnDemand version 1.8.19

Dev App Flow

Users can be authorized to be OnDemand developers, in which case they can run custom apps, stored in their home directory, using OnDemand. The dev app flow shows the interactions that occur when such an app is launched and used.

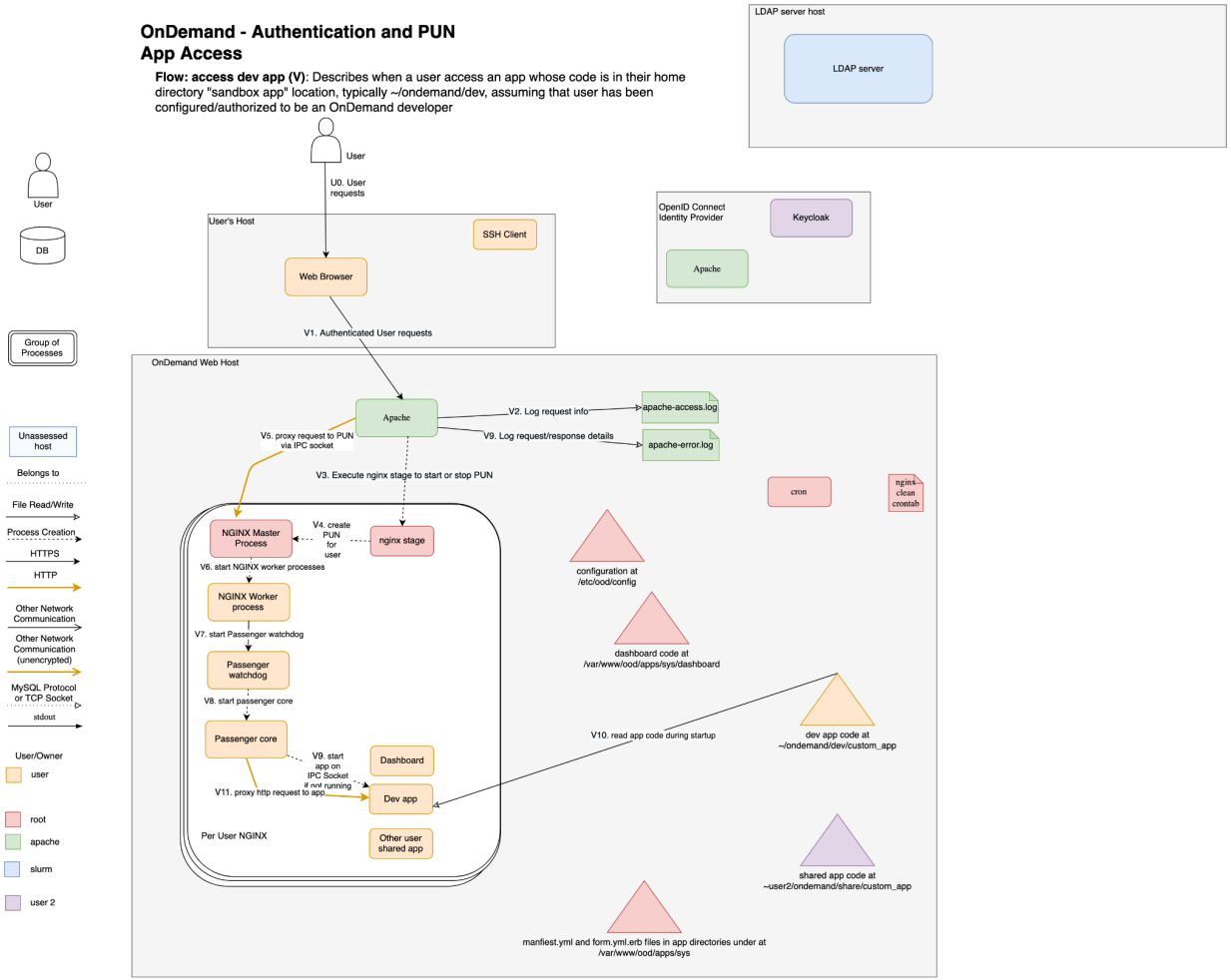

Figure D3. Dev App Flow Flow for Open OnDemand version 1.8.19

Linux Host Adapter Flow

The linux host adapter allows hosts that aren't configured with a scheduler to be used with On Demand. Rather than submitting jobs to the scheduler, OnDemand sets up an ssh connection to the server and starts the job itself in a tmux session. The Linux Host Adapter Flow shows the interactions that occur during its operation.

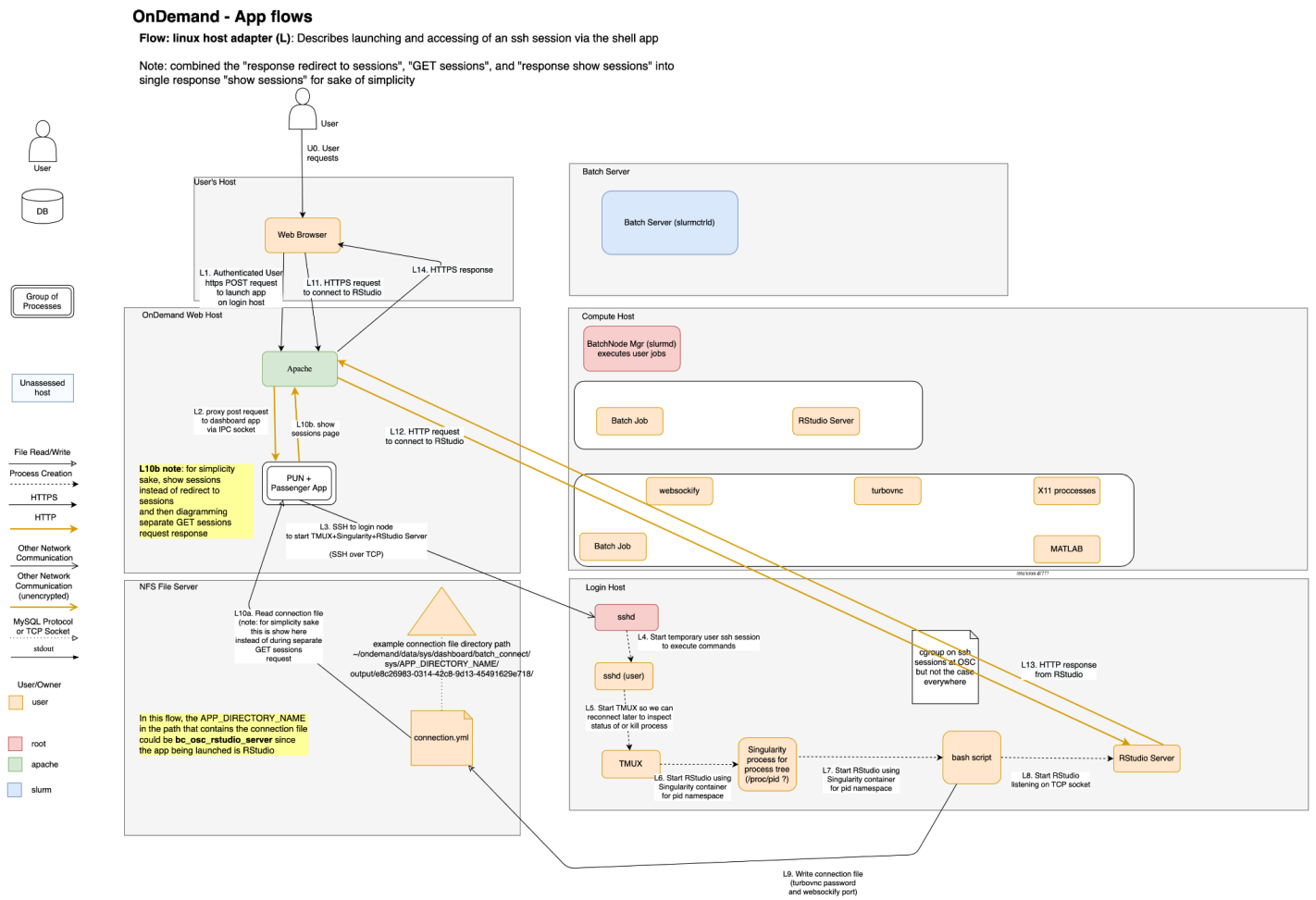

Figure D4. Linux Host Adapter Flow for Open OnDemand version 1.8.19

NGINX Cron Job Flow

The NGINX cron job runs periodically to kill old and inactive nginx instances. The NGINX cron job flow describes its interaction with the OnDemand instance's processes.

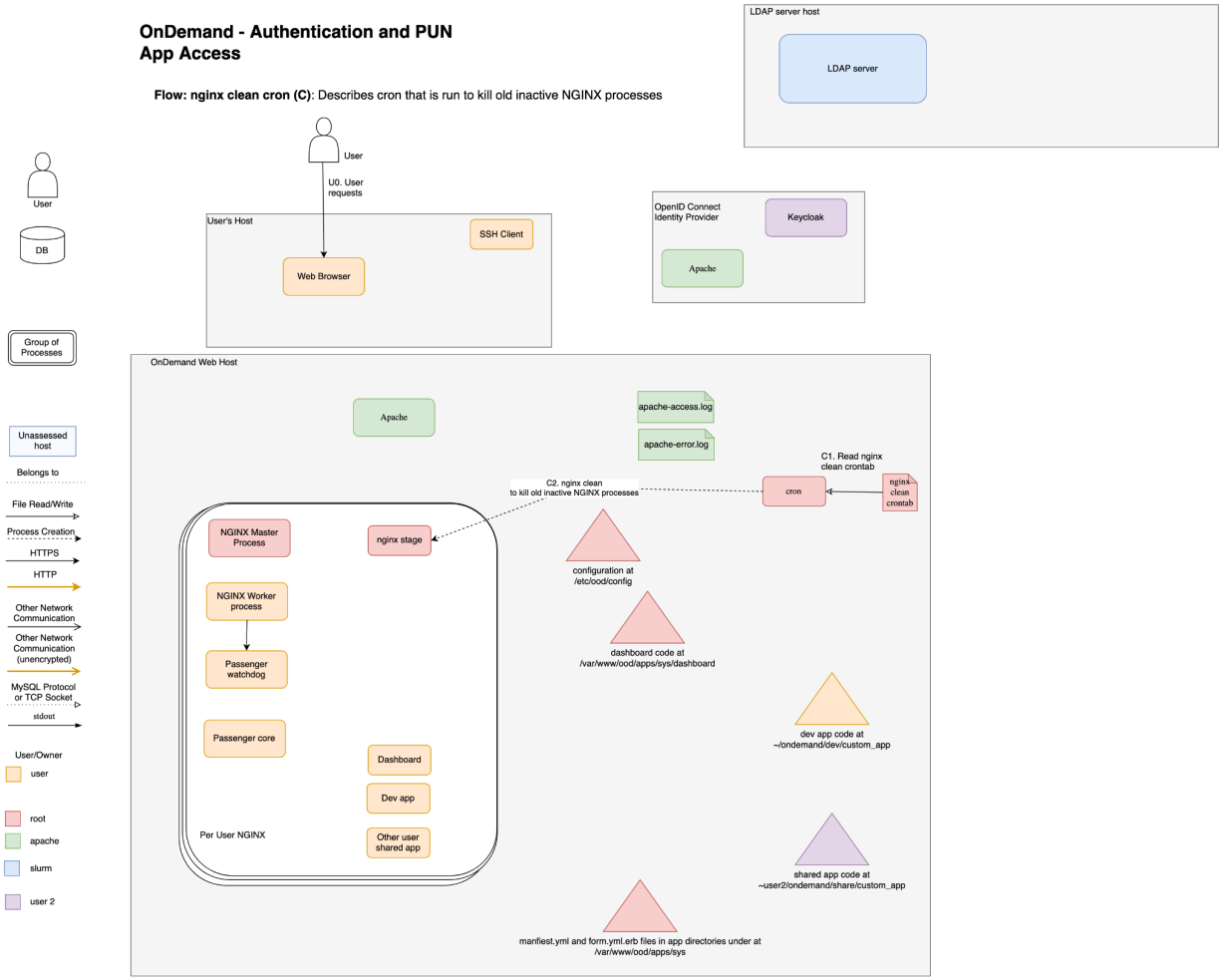

Figure D5. NGINX Cron Job Flow for Open OnDemand version 1.8.19

Shared App Flow

The operation of shared apps is similar to that of dev apps. It allows users to share code and/or binaries with other users of the OnDemand instance. The Shared App Flow shows its interactions with the OnDemand processes and resources.

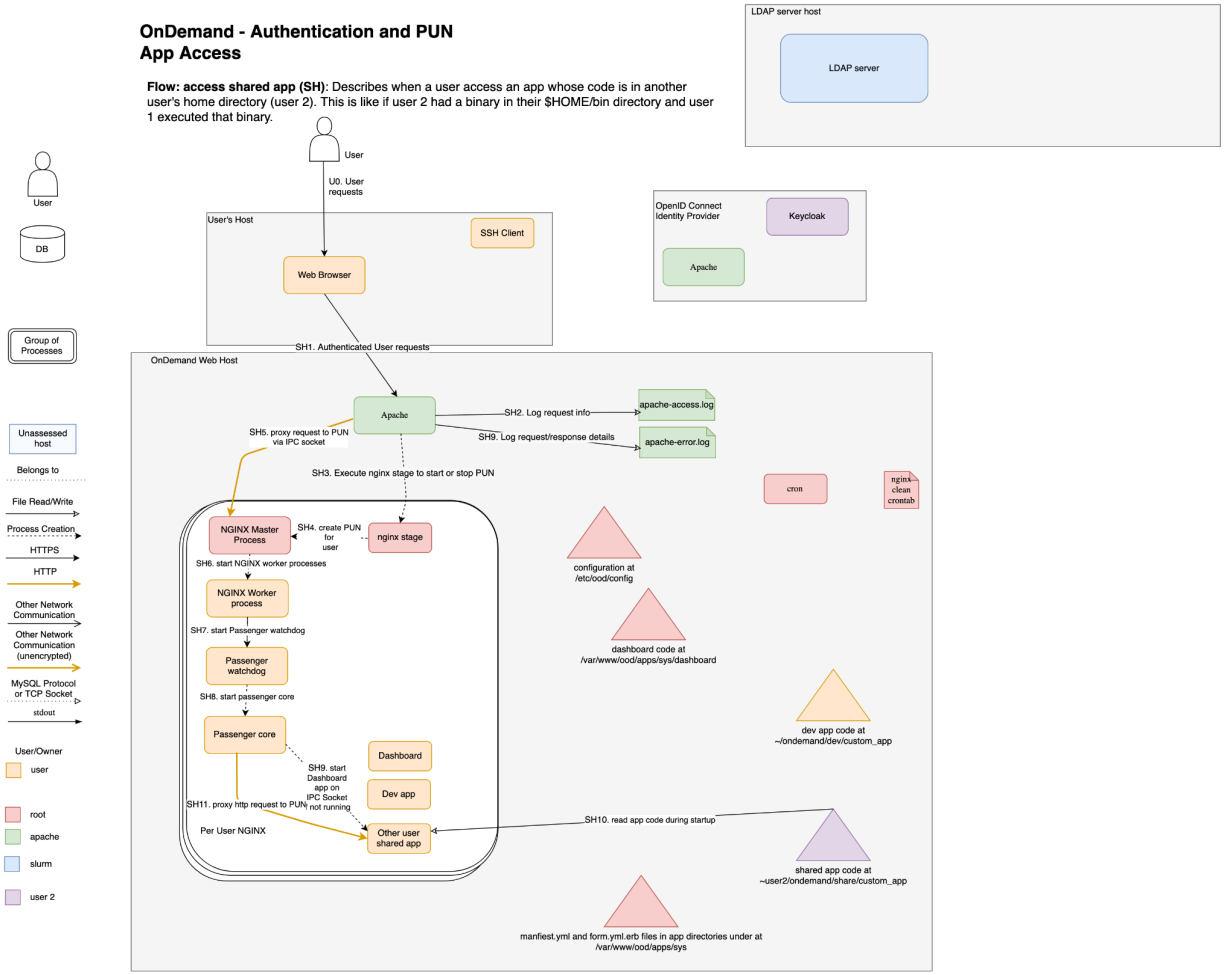

Figure D6. Shared App Flow for Open OnDemand version 1.8.19

Shell Session Flow

OnDemand allows a user to open a shell session from their browser on either a compute node or a login node. The Shell Session Flow shows the interactions that occur when the shell session is setup and run.

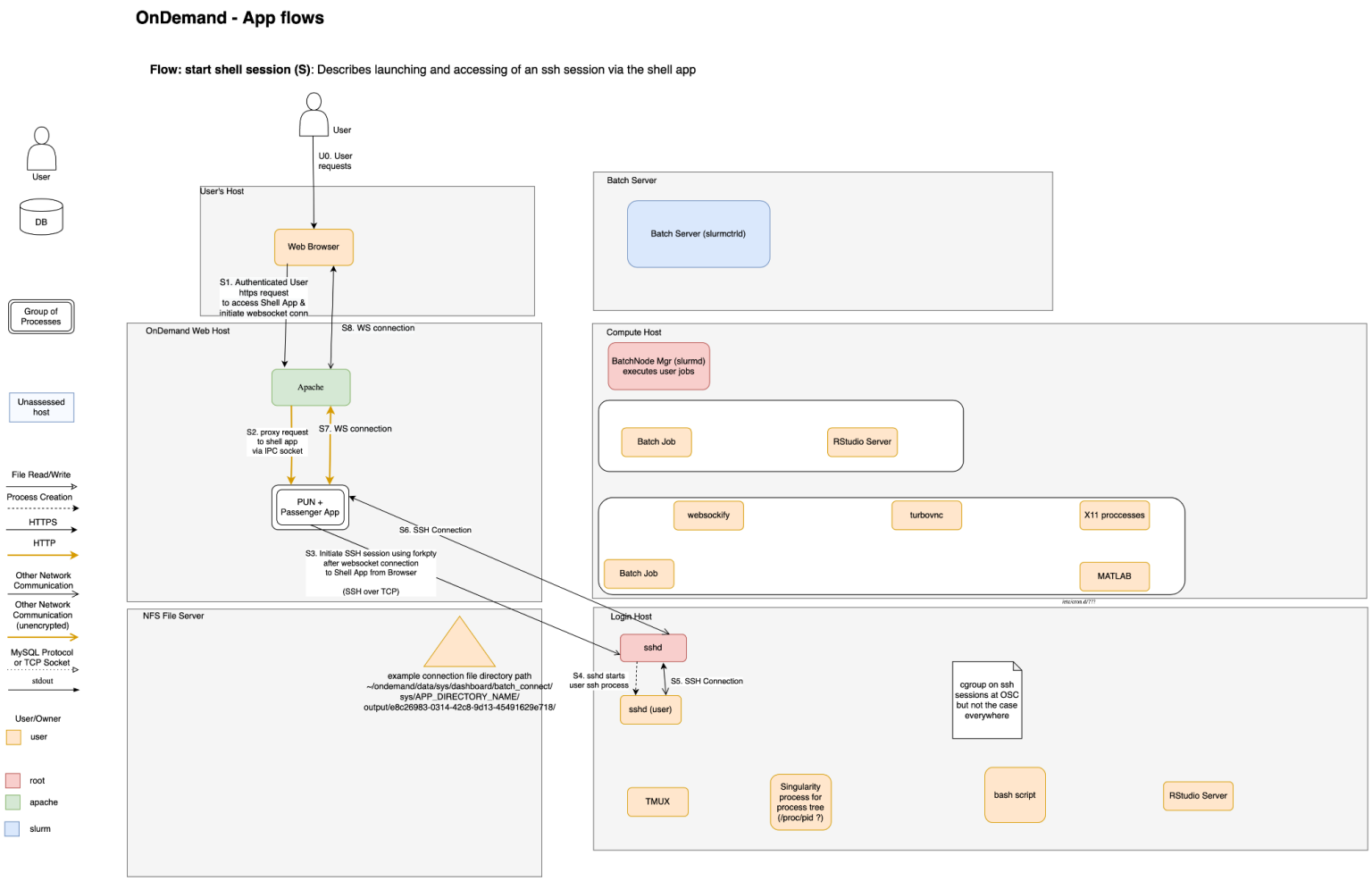

Figure D7. Shell Session Flow for Open OnDemand version 1.8.19

VNC App Flow

OnDemand allows users to run traditional desktop programs, such as IDEs or MATLAB, on a computer host by setting up a VNC connection between their browser and the OnDemand web host and proxying this connection to the compute node where the program is running. The VNC App Flow shows the interactions involved in setting up and maintaining this connection.

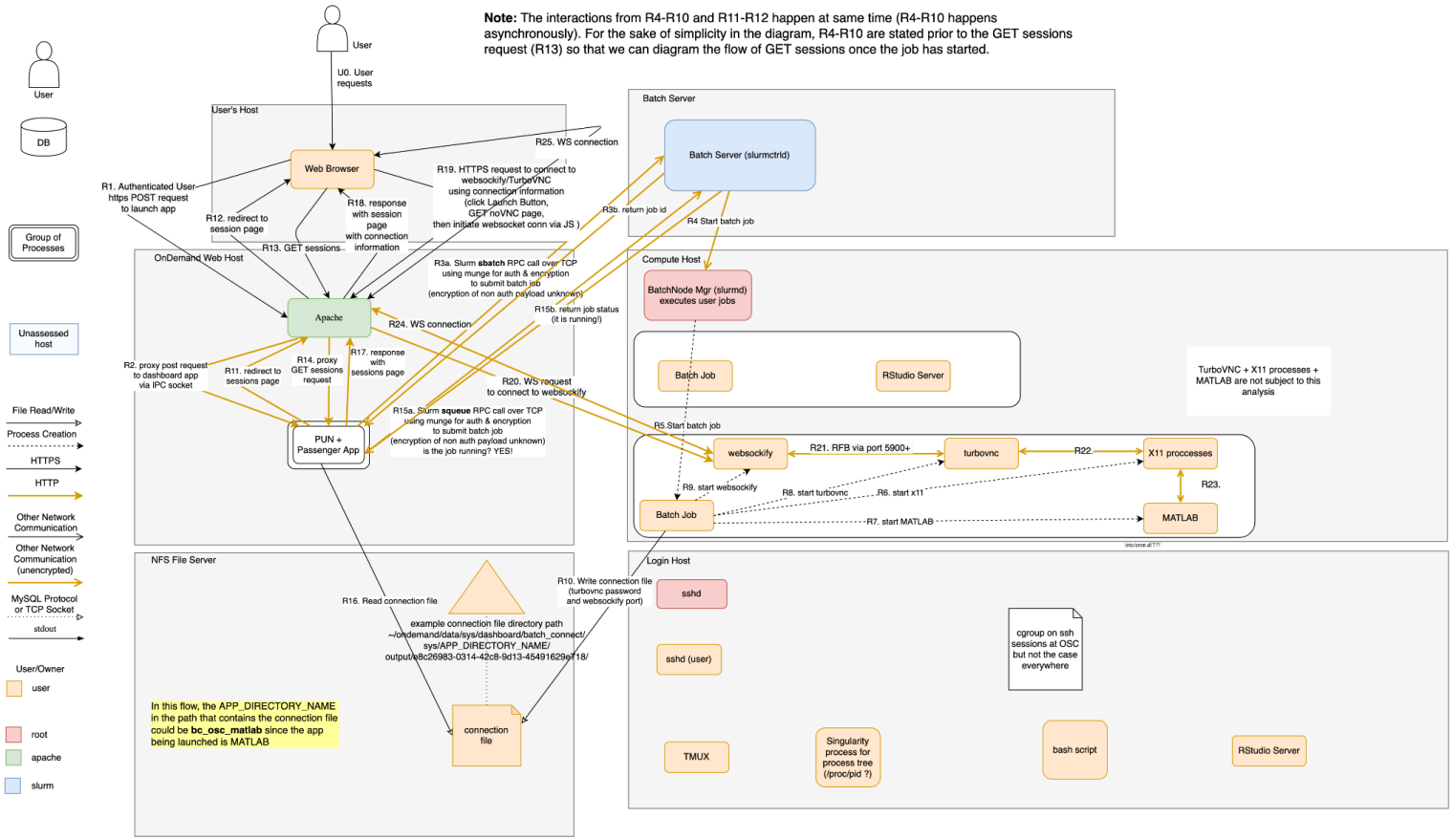

Figure D8. VNC App Flow for Open OnDemand version 1.8.1

Appendix E: Vulnerability Reporting Process Sample Text

Trusted CI and Open OnDemand developed this text during this engagement to describe vulnerability reporting expectations for the Open OnDemand user community.

The Open OnDemand project supports responsible vulnerability disclosure. We request that you do not report suspected security issues directly on github or through other public means such as the Open OnDemand discourse forum or public mailing lists. Vulnerabilities and security concerns should be reported by sending a message directly to the security reporting email address directly. If you wish to report a vulnerability, please send an email message to this address and include the following information:

1. *Any affected Open OnDemand version or versions identified*
2. *A description of the vulnerability or security concern*
3. *If applicable, any steps the Open OnDemand team can take to reproduce the behavior*
4. *Your preferred contact email address and means of address for clarification and follow-up emails*

The Open OnDemand team will send acknowledgement of reports to the provided email address within two business days of receipt and may request additional clarifying detail in further communications. Open OnDemand values reports of vulnerabilities and other security concerns. If a report is confirmed as a vulnerability, Open OnDemand will acknowledge the individual or organization which reported the vulnerability in announcements about the vulnerability unless otherwise requested to do otherwise.

The Open OnDemand team endeavors to remediate serious vulnerabilities and provide fixes for supported versions in under 14 days. For issues where this is not possible, Open OnDemand will distribute suggested steps which Open OnDemand users can take to mitigate risks until a patch is available through a message to the ood-users@lists.osc.edu mailing list and through an announcement on Open OnDemand's Discourse forum.[39] *In addition, a notification will be sent to alerts@trustedci.org for Trusted CI to distribute mitigation recommendations to the NSF CI community.*

[39] https://discourse.osc.edu/c/open-ondemand/announcements

Appendix F: Security Checklists

Pull request checklist - Security issues

1. Do you have adequate test coverage for the change? For example:
    a. Are expected states covered by tests?
    b. Do tests cover exceptional and unexpected states?
    c. Do tests account for nefarious input?
    d. Tests cover expected input to the interfaces.
2. Does this modify or add an attack vector? For example, reading user input from a web form or constructing database queries.
3. Are the default configurations reasonably secure? For example, are file permissions set correctly to prevent users from modifying files which are consumed by the software?
4. Does the code pass checks with Snyk, OWASP Dependency-Check, and static analysis tools? See release checklist for details.
5. Does this modify sensitive components? For example:
    a. Code which would run as a privileged user
    b. Code which handles authentication or authorization.

Release Tasks

Use automated scanning tools for security vulnerabilities and other flaws

1. Perform scans with Snyk, OWASP dependency-check, and static analysis tools
2. Result triage - verify scan results to eliminate false positives and either remediate issues or hold release to fix identified issues
    a. If you identify false positives, create a filter to eliminate the specific results from future scans
3. Make necessary changes to remediate any true positive results
4. Scan again. If tools don't identify anything new, move on to the next step.

Review and describe: What is fundamentally changing? Incorporate these observations into appropriate documentation when appropriate.

1. What are the security implications of those changes?
2. Are there new information assets,[40] new information flows, new open ports, new inputs/outputs, or other possible new avenues of attack?
3. Update existing security documentation including security diagrams, security procedures, FPVA model, dependency information, and sensitive components inventory as appropriate

Check configurations & documentation:

1. Are all new dependencies being checked by dependency tracking tools?
2. Are any other scanning tools configured properly so they scan any new components?
3. Are the changes in this version accurately represented in relevant security documentation?

When sensitive components are changed, check:

1. Are proper authorization controls implemented in new sections?
2. Are proper authentication controls being used for user access?
3. Is sensitive data encrypted if it would cross important boundaries? How are encryption keys handled for this communication?
4. Is there proper logging in important areas?

[40] Information assets are valuable, sensitive, and/or mission critical information and information systems. Software and software components are information assets.

5. Do logging mechanisms prevent exposure of sensitive information in logs?
6. Is data sanitization implemented for user input, cookies, SQL etc?
7. Are least privilege rules being followed? (don't use admin privileges when not needed)