From Pain to Gain: Vulnerability Management Developers won’t hate

DevSecOps Conf42 2024

About Me

Goals for this talk

Review Conventional Vulnerability Management

Examine advancements in tooling and how they can augment our processes

Look at automation opportunities

Discuss vulnerability analytics and report

Hopefully leave listeners with ideas

So what? What is the gain?

Time.

Vulnerability Management Refresher

Key Terms

Vulnerability (Vuln): A security flaw, glitch, or weakness found in software code that could be exploited by an attacker (threat source).

Exploit: A method or piece of code that takes advantage of vulnerabilities

CVSS: Common Vulnerability Scoring System, a method used to supply a qualitative measure of severity. *CVSS is not a measure of risk

EPSS: Exploit Prediction Scoring System, a data-driven effort for estimating the likelihood that a vulnerability will be exploited in the wild

CVE: Common Vulnerabilities and Exposure, a list of publicly disclosed vulnerabilities. When saying CVE typically its in reference to a specific vulnerability, e.g. CVE-2021-44889

Where exactly are the vulns?

Packages

Libraries

Binaries

Dependencies

Images (e.g. AMI, Docker, OCI)

Codebases

Vendors

Work Stations

Servers

Direct and Transitive Dependency Vulnerabilities

- **Direct:** The vulnerable dependency is explicitly imported by the maintainer's codebase
- **Transitive:** The vulnerable dependency is implicitly imported by being imported by a direct dependency or further sub dependency

Direct Dependency Example

```
{
  "name": "mergeConfigs",
  "version": "1.0.0",
  "main": "mergeConfigs.js",
  "dependencies": {
    "lodash": "4.6.0"
  }
}
```

NAME	INSTALLED	FIXED-IN	TYPE	VULNERABILITY	SEVERITY
lodash	4.6.0	4.17.12	npm	GHSA-jf85-cpcp-j695	Critical
lodash	4.6.0	4.17.19	npm	GHSA-p6mc-m468-83gw	High
lodash	4.6.0	4.17.11	npm	GHSA-4xc9-xhrj-v574	High
lodash	4.6.0	4.17.21	npm	GHSA-35jh-r3h4-6jhm	High
lodash	4.6.0	4.17.11	npm	GHSA-x5rq-j2xg-h7qm	Medium
lodash	4.6.0	4.17.5	npm	GHSA-fvqr-27wr-82fm	Medium
lodash	4.6.0	4.17.21	npm	GHSA-29mw-wpgm-hmr9	Medium

Transitive Dependency Example

```
{
  "name": "transient-demo",
  "version": "1.0.0",
  "dependencies": {
    "handlebars": "4.0.0"
  }
}
```

NAME	INSTALLED	FIXED-IN	TYPE	VULNERABILITY	SEVERITY
handlebars	4.0.0	4.3.0	npm	GHSA-w457-6q6x-cgp9	Critical
handlebars	4.0.0	4.7.7	npm	GHSA-f2jv-r9rf-7988	Critical
handlebars	4.0.0	4.7.7	npm	GHSA-765h-qjxv-5f44	Critical
handlebars	4.0.0	4.0.14	npm	GHSA-q42p-pg8m-cqh6	High
handlebars	4.0.0	4.5.3	npm	GHSA-q2c6-c6pm-g3gh	High
handlebars	4.0.0	4.5.3	npm	GHSA-g9r4-xpmj-mj65	High
handlebars	4.0.0	4.4.5	npm	GHSA-62gr-4qp9-h98f	High
handlebars	4.0.0	4.5.3	npm	GHSA-3cqr-58rm-57f8	High
handlebars	4.0.0	4.5.2	npm	GHSA-2cf5-4w76-r9qv	High
handlebars	4.0.0	4.4.5	npm	GHSA-f52g-6jhx-586p	Medium
minimist	0.0.10	0.2.4	npm	GHSA-xvch-5gv4-984h	Critical
minimist	0.0.10	0.2.1	npm	GHSA-vh95-rmgr-6w4m	Medium
uglify-js	2.4.24	2.6.0	npm	GHSA-c9f4-xj24-8jqx	High

- transient-demo (Root Library)
  - handlebars-4.0.0.tgz
    - optimist-0.6.1.tgz
      - ❌ minimist-0.0.10.tgz (Vulnerable Library)

Docker Image/Binary/Package Example

```
FROM alpine:3.14

RUN apk add --no-cache \
    chromium=93.0.4577.82-r0 \ # Vulnerable Binary/Package
    nodejs \ # Installed Packages Should be Version Pinned!
    npm

RUN npm install selenium-webdriver@3.3.0 # Vulnerable Dependency/Library/Package
```

More Vulnerability Management FAQ

Where do vulnerabilities come from?: Security researchers discover vulnerabilities and disclose them. Then a CVE Numbering Authority (CNA) will issue a CVE number (id)

How do we find the vulnerabilities?: They can be found using a vulnerability scanner.

What if we cannot patch our vulnerabilities?: An exception may be granted in cases where there is an inability to patch within defined SLA's.

How often should we be scanning?: Always be scanning, scan the dependencies, scan the container images and artifacts. Scan it before commit, during the CI pipeline and in production. Scanning early on helps proactively plan around vulnerabilities before they are found later on.

Traditional Vulnerability Management Programs

- Set up Scanning server and infrastructure
- Install agents on every machine
- Schedule scans
- Receive large volume of results
- Triage
- Contact Remediators
- Schedule Maintenance Window
- Change control board? Release process?
- Vette updates in test lab?

Severity	SLA
Critical	15
High	30
Medium	60
Low	90

To dev-team@madeupcompany.com Cc Bcc

Remediate ASAP!

Hey team,

Our scanner picked up vulnerabilities in your project, can you find time in your current sprint to take care of these?

	A	B	C	D	E	F
1	NAME	INSTALLED	FIXED-IN	TYPE	VULNERABILITY	SEVERITY
2	@npmcli/arborist	2.6.2	2.8.2	npm	GHSA-gmw6-94gg-2rc2	High
3	@npmcli/arborist	2.6.2	2.8.2	npm	GHSA-2h3h-q99f-3fhc	High
4	ansi-regex	3.0.0	3.0.1	npm	GHSA-93q8-gq69-wqmw	High
5	ansi-regex	5.0.0	5.0.1	npm	GHSA-93q8-gq69-wqmw	High
6	avahi-libs	0.8-r5		apk	CVE-2023-38473	Medium
7	avahi-libs	0.8-r5		apk	CVE-2023-38472	Medium
8	avahi-libs	0.8-r5		apk	CVE-2023-38471	Medium
9	avahi-libs	0.8-r5		apk	CVE-2023-38470	Medium
10	avahi-libs	0.8-r5		apk	CVE-2023-38469	Medium
11	busybox	1.33.1-r8		apk	CVE-2022-48174	Critical
12	c-ares	1.17.2-r0		apk	CVE-2024-25629	Medium
13	chromium	93.0.4577.82-r0		apk	CVE-2024-7971	Critical
14	chromium	93.0.4577.82-r0		apk	CVE-2024-7024	Critical
15	chromium	93.0.4577.82-r0		apk	CVE-2024-5274	Critical
16	chromium	93.0.4577.82-r0		apk	CVE-2024-4949	Critical
17	chromium	93.0.4577.82-r0		apk	CVE-2024-4947	Critical
18	chromium	93.0.4577.82-r0		apk	CVE-2024-4671	Critical

The attachment has the full list.

Challenges faced by Conventional Programs

- Resource constraints (e.g. staff shortages)
- Very flat approach to severity and SLA's
- Documenting known issues
- Response
  - If another Log4j or Polyfill incident occurs, how quickly can you respond?
- Sheer number of vulnerabilities is increasing
  - More vulnerabilities, more fatigue experienced by maintainers
  - More time spent on vulnerabilities, more time spent on triage, less on features
  -
- Tooling
  - Maintenance burden of scanning systems
  - Lack of features or integrations

Revamping Vulnerability Management

DevOps + Vulnerability Management Programs

- Real-time and event driven Vulnerability detection workflows
- Numerous automated patching tools
  - Integrate right onto the codebase
  - Patch your servers without requiring a reboot
  - Instance replacement
  - Rolling updates/Blue green
- Everyone's got a scanning feature
- CVSS + EPPS + More Data && Info = New SLA equation
- Capable analytics solutions for better reporting and tracking
- More integrations than ever

alaiuppa 12:14 PM

Hey, our scanner saw your project had some vulnerabilities, can you find some time this sprint to review this Excel sheet and take care of these? Thanks!

findings.png

	A	B	C	D	E	F
1	NAME	INSTALLED	FIXED-IN	TYPE	VULNERABILITY	SEVERITY
2	@npmcli/arborist	2.6.2	2.8.2	npm	GHSA-gmw6-94gg-2rc2	High
3	@npmcli/arborist	2.6.2	2.8.2	npm	GHSA-2h3h-q99f-3fhc	High
4	ansi-regex	3.0.0	3.0.1	npm	GHSA-93q8-gq69-wqmw	High
5	ansi-regex	5.0.0	5.0.1	npm	GHSA-93q8-gq69-wqmw	High
6	avahi-libs	0.8-r5		apk	CVE-2023-38473	Medium
7	avahi-libs	0.8-r5		apk	CVE-2023-38472	Medium
8	avahi-libs	0.8-r5		apk	CVE-2023-38471	Medium
9	avahi-libs	0.8-r5		apk	CVE-2023-38470	Medium
10	avahi-libs	0.8-r5		apk	CVE-2023-38469	Medium
11	busybox	1.33.1-r8		apk	CVE-2022-48174	Critical
12	c-ares	1.17.2-r0		apk	CVE-2024-25629	Medium
13	chromium	93.0.4577.82-r0		apk	CVE-2024-7971	Critical
14	chromium	93.0.4577.82-r0		apk	CVE-2024-7024	Critical
15	chromium	93.0.4577.82-r0		apk	CVE-2024-5274	Critical
16	chromium	93.0.4577.82-r0		apk	CVE-2024-4949	Critical
17	chromium	93.0.4577.82-r0		apk	CVE-2024-4947	Critical
18	chromium	93.0.4577.82-r0		apk	CVE-2024-4671	Critical

Junior Analyst Slack's a maintainer asking for some remediation, *Circa 2018*

Vex

Vulnerability Exploitability eXchange (VEX) a document that acts as a form of security advisory indicating whether or not a product is affected by a known vulnerability or vulnerabilities

Using Vex software authors can communicate to their users that an otherwise vulnerable component has no security implications in their product.

Vex documents allow “turning off” security alerts scanner alerts of vulnerabilities known not to affect the product*

If you scanner respects Vex documents, this is in early adoption

Code Reachability

A process that determines if a vulnerability in code, libraries, or containers can be exploited in a given environment.

- Reachable
  - The vulnerable function is called in a manner that can be interacted with
- Conditionally Reachable
  - The vulnerable function is called in a manner that can be interacted with if certain criteria are met
- Always Reachable
  - Just importing the vulnerable package makes you susceptible
- Unreachable
  - The vulnerable code is not present or not able to be interacted with outside of the programs runtime

Some vendors are able to automate this!

Code Reachability Example

```
{
  "name": "code-reachability-demo",
  "version": "1.0.0",
  "main": "index.js",
  "dependencies": {
    "lodash": "4.6.0"
  }
}
```

NAME	INSTALLED	FIXED-IN	TYPE	VULNERABILITY	SEVERITY
lodash	4.6.0	4.17.12	npm	GHSA-jf85-cpcp-j695	Critical
lodash	4.6.0	4.17.19	npm	GHSA-p6mc-m468-83gw	High
lodash	4.6.0	4.17.11	npm	GHSA-4xc9-xhrj-v574	High
lodash	4.6.0	4.17.21	npm	GHSA-35jh-r3h4-6jhm	High
lodash	4.6.0	4.17.11	npm	GHSA-x5rq-j2xg-h7qm	Medium
lodash	4.6.0	4.17.5	npm	GHSA-fvqr-27wr-82fm	Medium
lodash	4.6.0	4.17.21	npm	GHSA-29mw-wpgm-hmr9	Medium

Can you spot which is vulnerable?

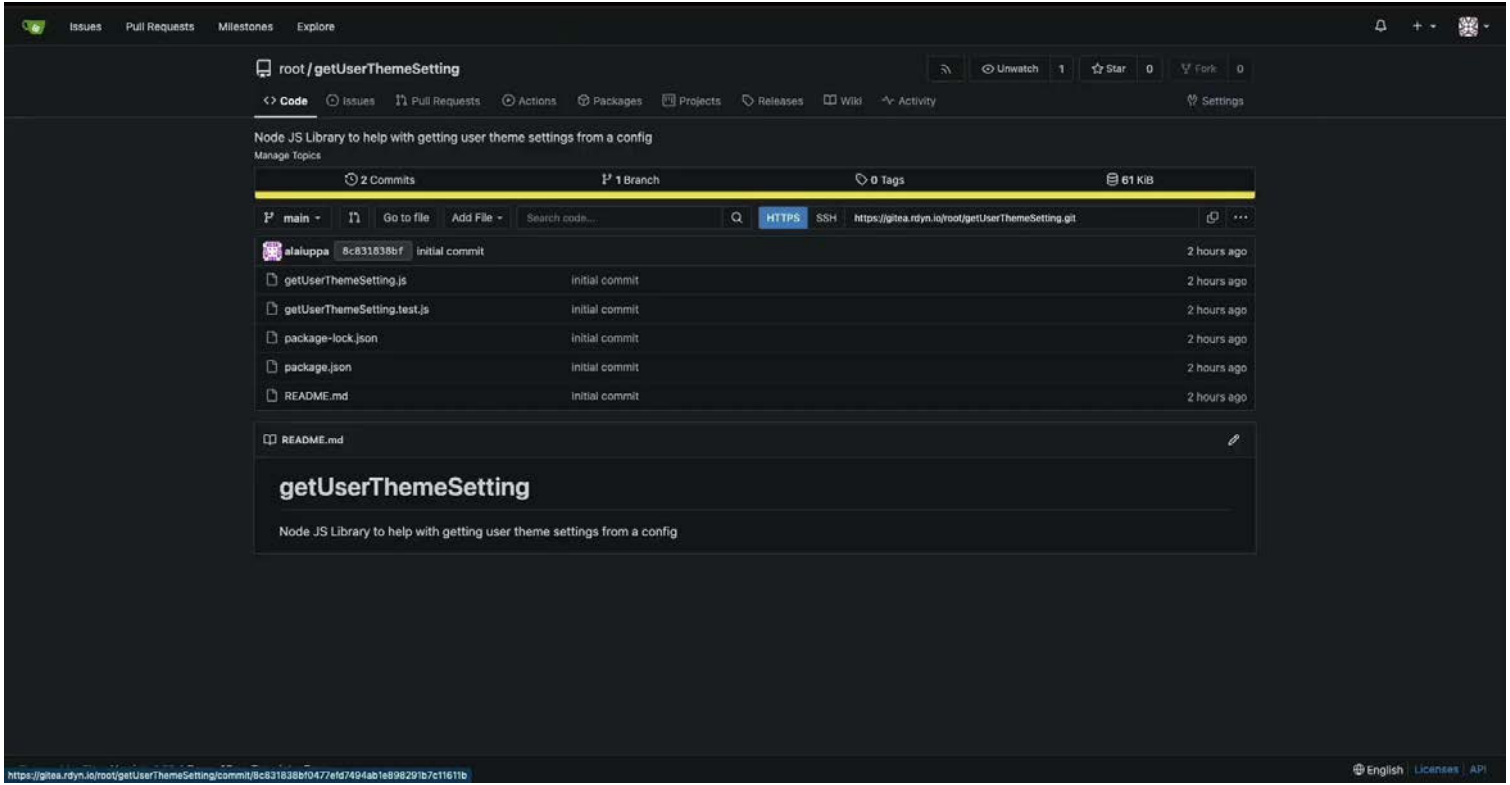
CVE-2018-16487

A prototype pollution vulnerability was found in lodash <4.17.11 where the **functions merge**, **mergeWith**, and **defaultsDeep** can be tricked into adding or modifying properties of Object.prototype.

```
const lodash = require('lodash');

// mergeConfigs returns merged result of defaultConfig and userConfig
function mergeConfigs(userConfig) {
    const defaultConfig = { safe: true };
    return lodash.merge({}, defaultConfig, userConfig); // Direct use of lodash.merge
}

module.exports = mergeConfigs;
```

```
const lodash = require('lodash');

// getUserThemeSetting returns the color of a theme in a userConfig
function getUserThemeSetting(userConfig) {
    const defaultConfig = { theme: { color: 'blue' } };
    // Using lodash.get to retrieve the color setting
    return lodash.get(userConfig, 'theme.color', defaultConfig.theme.color);
}

module.exports = getUserThemeSetting;
```

Vex Demo


Issues
Pull Requests
Milestones
Explore
root / getUserThemeSetting
Unwatch 1
Star 0
Fork 0
Code
Issues
Pull Requests
Actions
Packages
Projects
Releases
Wiki
Activity
Settings
Node JS Library to help with getting user theme settings from a config
Manage Topics
2 Commits
1 Branch
0 Tags
61 KiB
main
Go to file
Add File
Search code...
HTTPS
SSH
https://gitea.rdyn.io/root/getUserThemeSetting.git
alaiuppa
8c831838bf
initial commit
2 hours ago
getUserThemeSetting.js
initial commit
2 hours ago
getUserThemeSetting.test.js
initial commit
2 hours ago
package-lock.json
initial commit
2 hours ago
package.json
initial commit
2 hours ago
README.md
initial commit
2 hours ago
README.md
getUserThemeSetting
Node JS Library to help with getting user theme settings from a config
English
Licenses
API

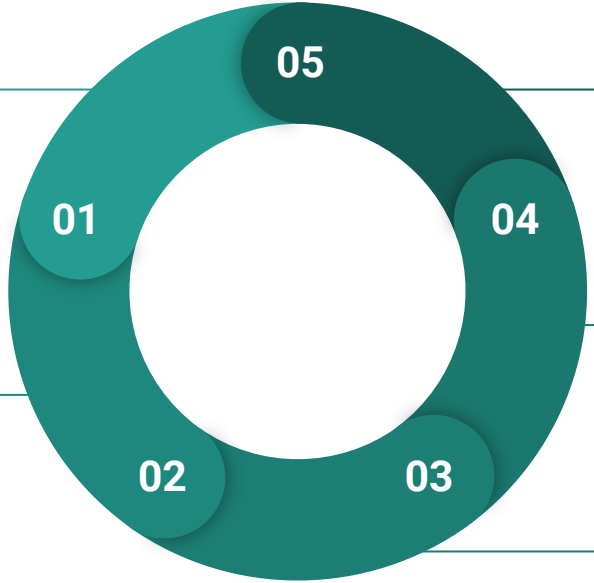
Rethinking Production Vuln Management

Scanner reports findings

Your scanner of choice returns a series of unfiltered findings.

Exceptions applied

The downstream consumer searches for Exceptions to remove those vulnerability findings from ticketing



Ticket

Using everything prior to remove vulnerabilities and adjust the SLA, the developers now have a better remediation priority.

Exposure Check

Is this an internal or external service? Is it a job that talks to nothing? Can additional tags or labels be used to consider mitigating/compensating controls?

Vex and Code Reachability

Deeply automate triage and introspection by assessing if package maintainers have added Vex documents, or running automated code reachability analysis

Redefining SLA’s

Automating intake and triage of vulnerabilities has it’s limit, mainly the amount of information available. Factoring in things like EPSS, Code Reachability, Network exposure and other aspects can make a noticeable difference.

Even slight additions of information can help adjust the SLA for remediation.

For example a HIGH vulnerability that is part of a *JOB*, and is only *Conditionally Reachable* may have its SLA extended due to the significantly lower likelihood of exploitation.

Automation Strategies

DevOps deployment strategies and tooling make deploying and testing remediations a breeze.

Things like:

- Deploying to lower environments
- CI pipelines that run tools like Renovate Bot & Automated tests
- Rolling deployments, instance refresh, blue green
- Having IaC that can recreate environments or substacks consistently
- Live kernel patching
- Scratch/Distroless containers

Renovate Bot

We released Renovate v39. Read the Release notes for major versions of Renovate to learn what's changed.

Renovate Docs

Search

renovatebot/renovate

39.43.0 17.9k 2.4k

Renovate Docs
Home
Reading List
Getting Started
Troubleshooting
Configuration
Mend-hosted Apps
Key concepts
Renovate Modules
Language Support
Deep Dives
Included Presets
All Other
About Us
Contributing to Renovate

MEND Renovate

Renovate documentation

Automated dependency updates. Multi-platform and multi-language.

Why use Renovate?

Automatic updates

Get pull requests to update your dependencies and lock files.

On your schedule

Reduce noise by scheduling when Renovate creates PRs.

Works out of the box

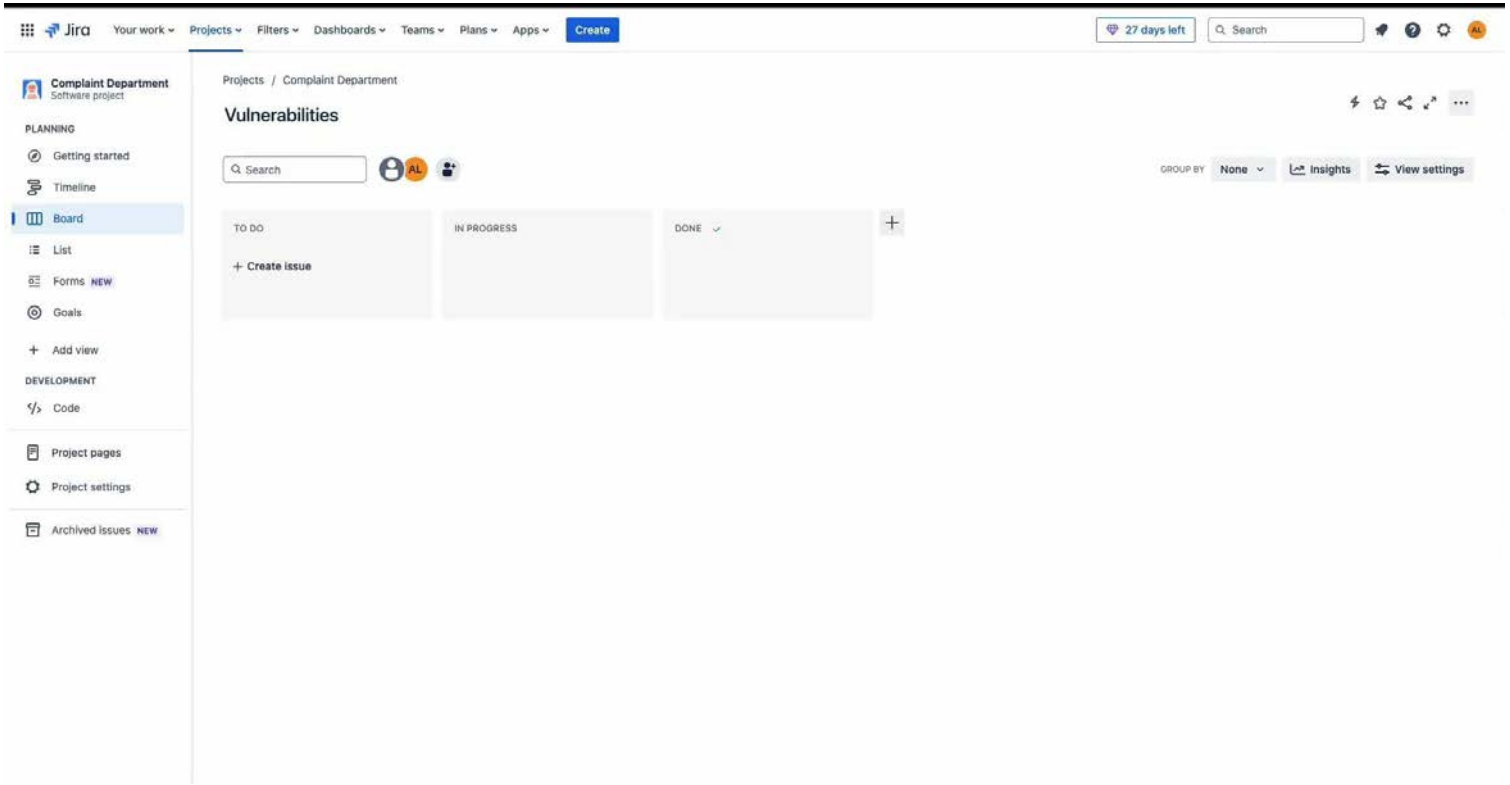
Renovate finds relevant package files automatically, including in monorepos.

How you like it

You can customize the bot's behavior with configuration files.

Table of contents

Event Driven Vulnerability Response Demo



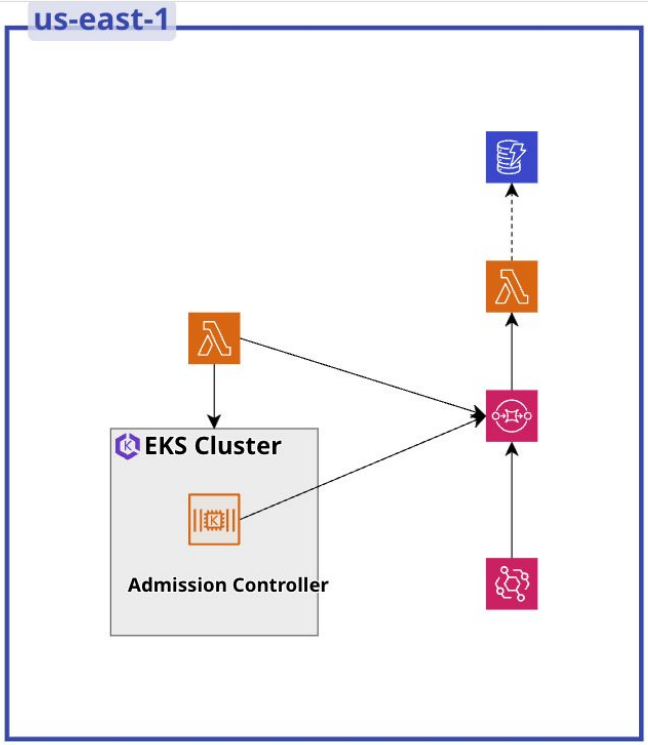
A word on Kubernetes

There is a tremendous amount of vendor solutions available to handle vulnerabilities within Kubernetes

You can also proactively gate keep by enforcing failing on <severity> in your CI pipeline, or just proactively scan your artifacts.

Ideas for constructing your own Kubernetes solution, you could create an “Admission Controller” which would give you the ability to do things like reject pods that dont have scanned images.

Or report what images come through and schedule them to be scanned, or create an external Lambda/service to do the same!



A brief word on Reporting and Analytics

Being able to quantify the amount of reduced risk and exposure that comes from remediating vulnerabilities will speak volumes about the program's effectiveness.

Aside from being more secure you can directly translate time saved to development labor hours saved.

Most vendors and tools have some semblance of reporting. Depending on how you structure your program, you may have additional options.

Vulnerability burn downs over time, breakdowns by team, vulnerabilities by resource type.

Every critical vulnerability may not be truly critical, it’s important we know which and as fast as possible.