


KubeCon | CloudNativeCon

North America 2021

RESILIENCE
REALIZED

Keeping Up with the CVEs!

How to Find a Needle in a Haystack?

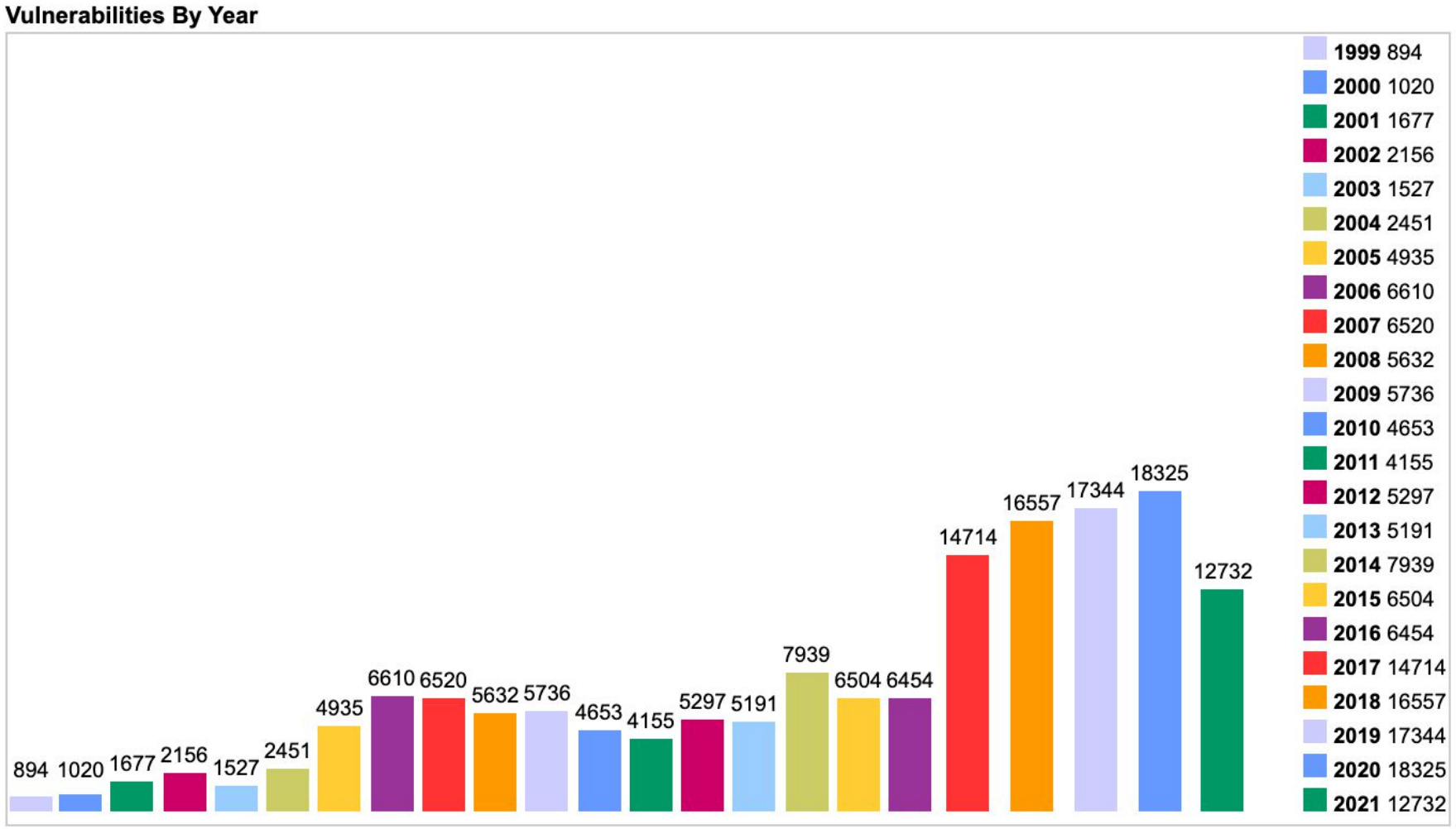
Pushkar Joglekar, *Sr. Security Engineer, VMware*

About Me

- Pushkar Joglekar (he/him)
- Nashik -> Pune -> S.F. Bay Area
- Speaks Marathi, Hindi and English
- Sr. Security Engineer @ VMware Tanzu
- Previously Security Engineer @ Visa (End User)
- Co-wrote “The Kubernetes Book” with Nigel Poulton
- Leads Kubernetes SIG Security Tooling Sub-Project
- CNCF Security TAG Contributor
- Find more about me: https://pushkarj.github.io/
- More active on Twitter *than* LinkedIn

@PuDiJoglekar

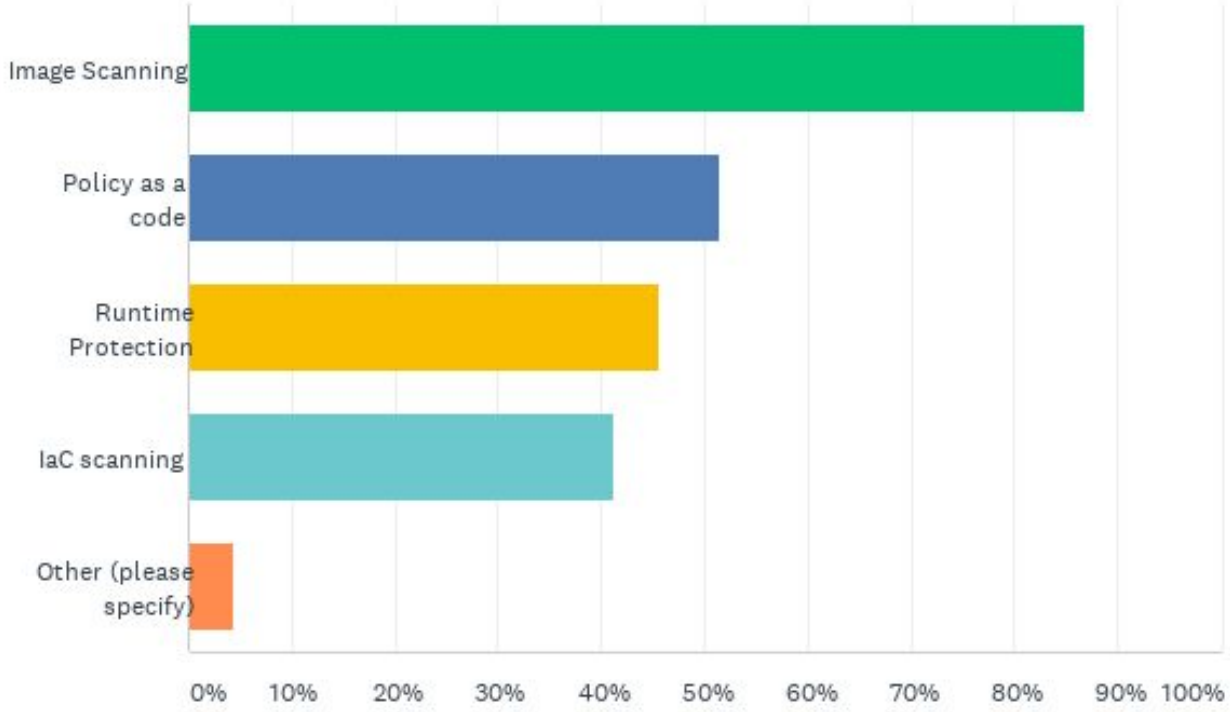
What's the deal with the CVEs?



Ref: https://www.cvedetails.com/browse-by-date.php

Cloud native security survey 2021

Q7 What types of tools are you currently using to protect your cloud native applications? (Select all that apply)

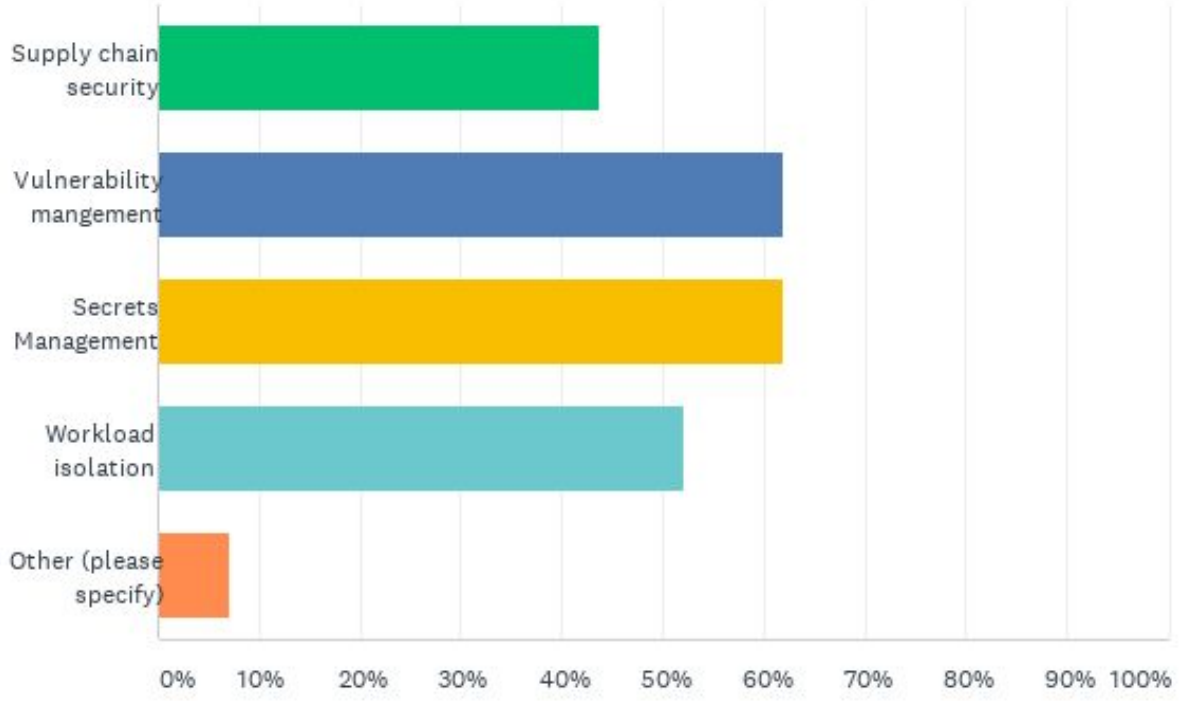


Ref: https://github.com/cncf/surveys/tree/master/security

Cloud native security survey 2021

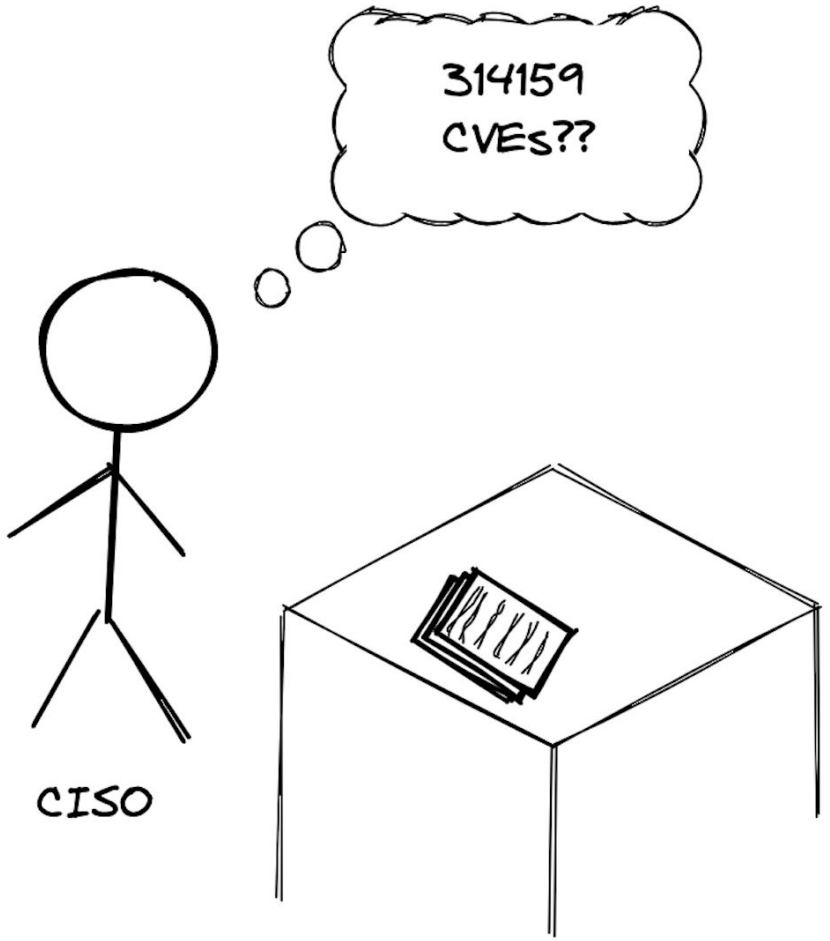
KubeCon CloudNativeCon North America 2021

Q4 What is the topmost cloud native security concern for you right now? (Select all that apply)



Ref: https://github.com/cncf/surveys/tree/master/security

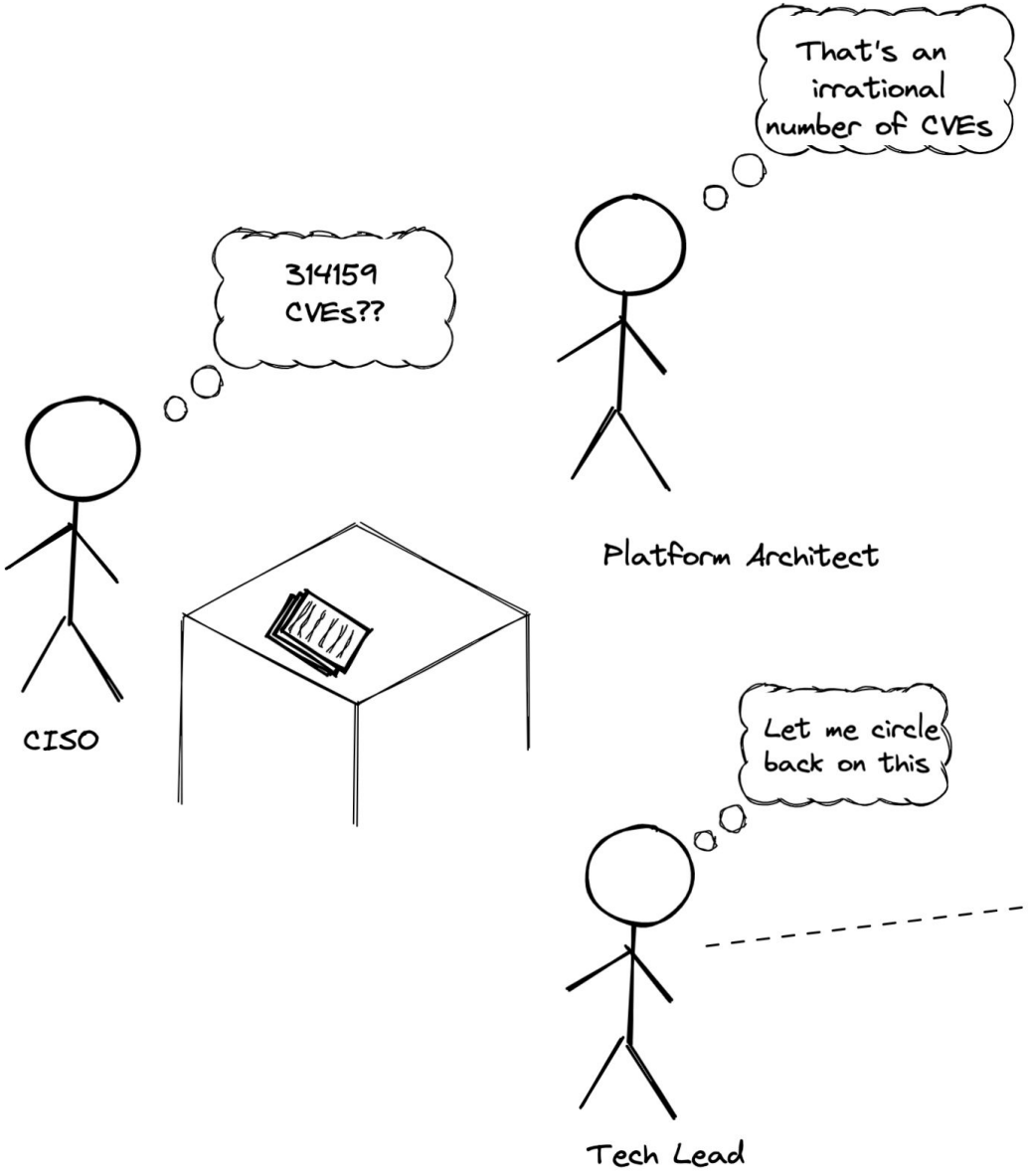
How does this impact real world?



Enterprise Customer

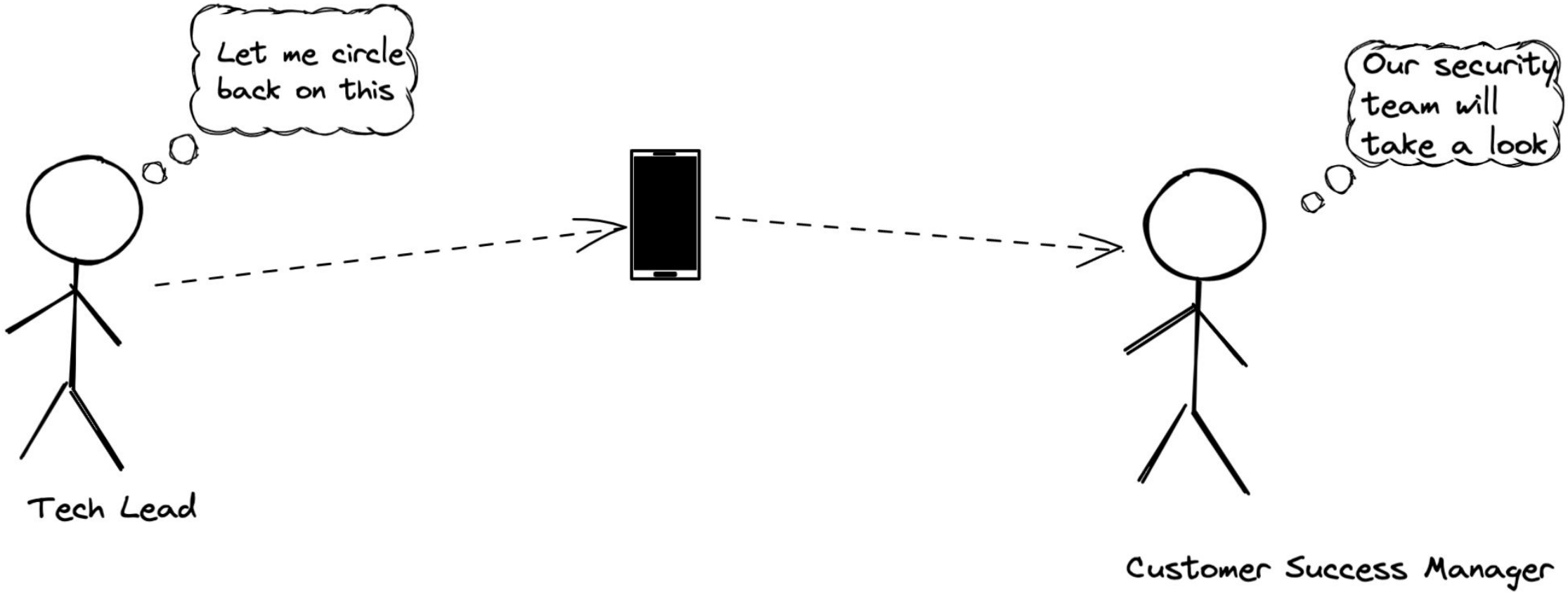
This is a fictional account

How does this impact real world?



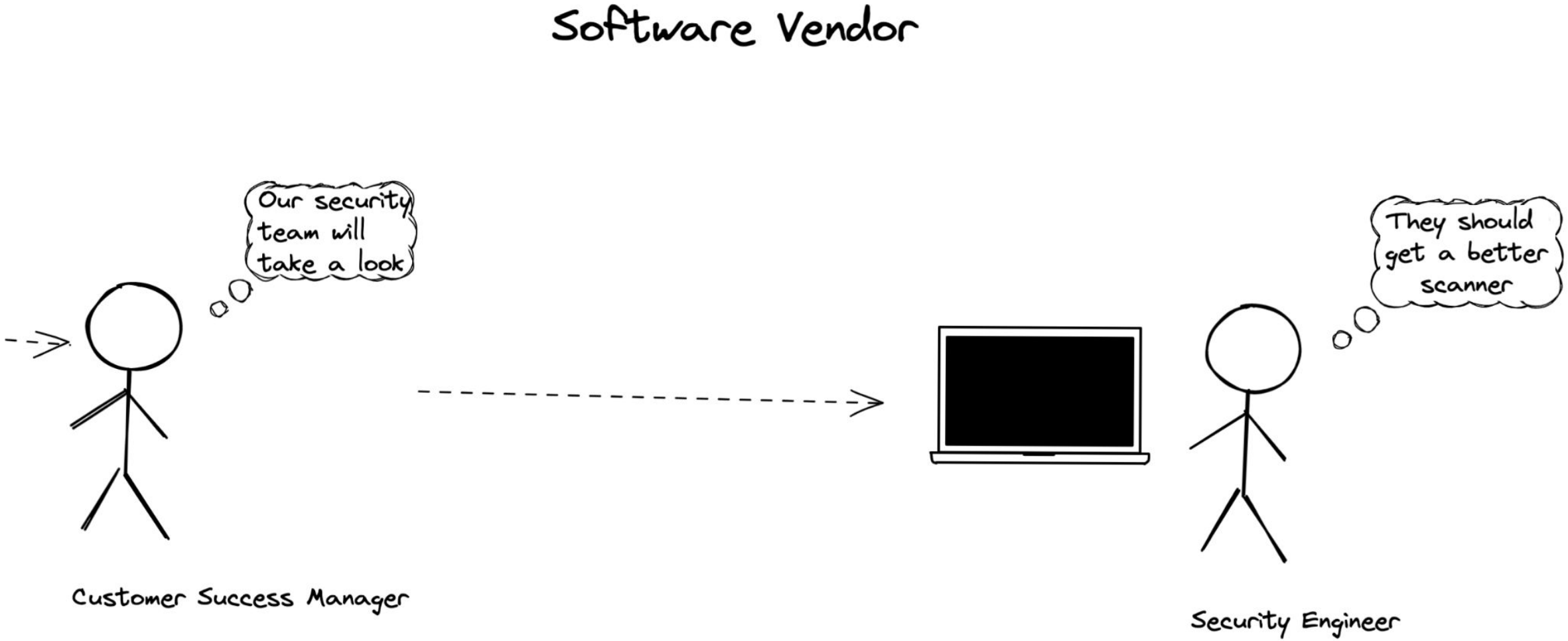
This is a fictional account

How does this impact real world?



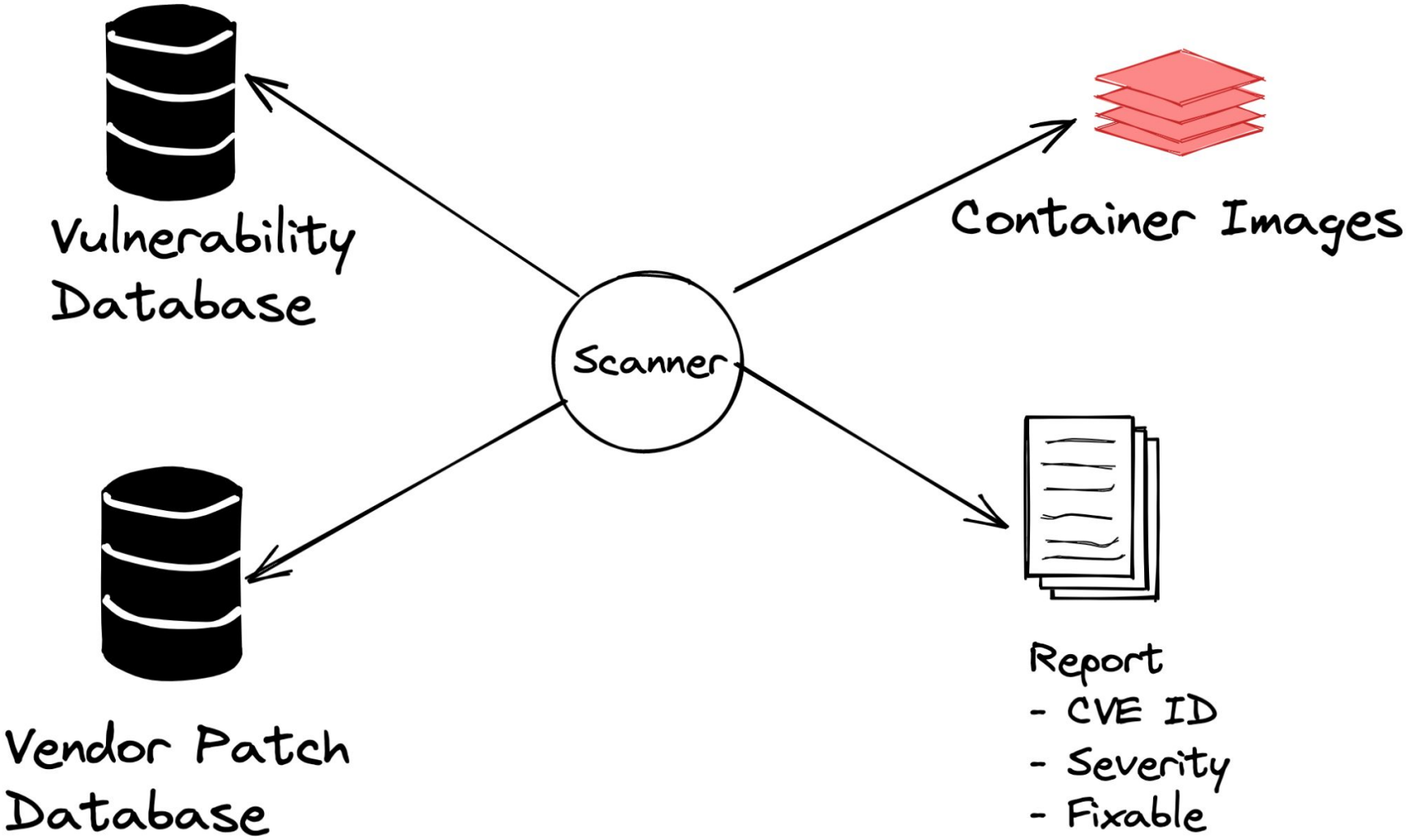
This is a fictional account

How does this impact real world?



This is a fictional account

Basic Vulnerability scanner



Demo
Analyzing CVE-2019-9923

Demo
Analyzing CVE-2018-6829

Demo
Analyzing CVE-2021-3711

Zero Trust != Zero CVEs

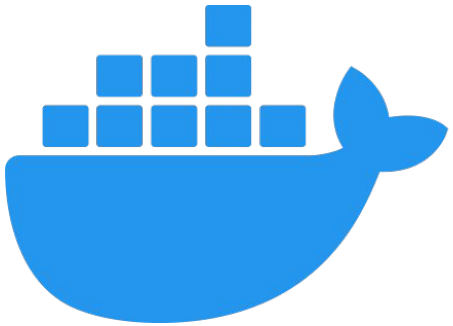
- Signal to Noise Ratio
  - False positives + negatives
  - Tied to severity than risk
- Impact assessment takes time, effort and expertise
- Risk v/s Reward of updating images
- Too many unneeded packages

Zero Trust == Secure in spite of CVEs

- Knowing the impact of a vulnerability
- Knowing the threat model and risk posture
- Being quick to Detect, Isolate and Fix
- Being able to do all of this in parallel

This is not just an end user problem

kubernetes



docker®

Choosing what to focus on

- Signal to Noise Ratio
  - False positives + negatives
  - Tied to severity than risk
- Impact assessment takes time, effort and expertise
- **Risk v/s Reward of updating images**
- **Too many unneeded packages**

How many images are part of Kubernetes release?

Shoutout to **@Puerco**

```
curl -L https://sbom.k8s.io/v1.22.1/release
| grep 'PackageName: k8s.gcr.io/'
| awk '{print $2}'
```

How are these images maintained?

- kube-apiserver
- kube-scheduler
- kube-controller-manager
- kube-proxy
- conformance (skipping)

  & etcd

Let’s go back to 2020 - Disclaimer

- Many community contributors made this happen!
- Focus is on container images only
- Future is even more exciting!

Let’s go back to 2020 - KEPs

Shoutout to **@yuvenma**

Motivation

Rebasing the k8s images to distroless/static can make the images thinner, safer and less vulnerable.

Meanwhile, it will drastically reduce churn on the total number of k8s images versions. Due to the fact that many images are based on debian base and a vulnerability in debian base (a couple times a month) will result in rebuilding every image, changing the image from debian base to distroless/static can reduce the total number of k8s image versions.

What's more, it reduces the burden of managing and maintaining multiple k8s images from the security (e.g. CVE), compatibility and build process concerns.

Ref: https://github.com/kubernetes/enhancements/tree/master/keps/sig-release/1729-rebase-images-to-distroless

But first what is distroless?

Distroless: Base & Static

Statically compiled applications (Go) that do not require libc can use the `gcr.io/distroless/static` image, which contains:

- ca-certificates
- A /etc/passwd entry for a root user
- A /tmp directory
- tzdata

Most other applications (and Go apps that require libc/cgo) should start with `gcr.io/distroless/base`, which contains all of the packages in `gcr.io/distroless/static`, and

- glibc
- libssl
- openssl

Ref: https://github.com/GoogleContainerTools/distroless/tree/main/base

Security v/s Usability

Enter go-runner

Before

```
"/bin/sh",
  "-c",
  "exec /usr/local/bin/kube-apiserver </snip> 1>>/var/log/kube-apiserver.log
2>&1"
```

After

```
"/go-runner", "--log-file=/var/log/kube-apiserver.log", "--also-stdout=false",
"--redirect-stderr=true",
  "/usr/local/bin/kube-apiserver",
  </snip>
```

Ref: https://github.com/kubernetes/kubernetes/pull/90804

Enter bash-static

Switch to static bash and distroless image for etcd #91171

Merged **k8s-ci-robot** merged 2 commits into `kubernetes:master` from `dims:switch-etcd-to-bash-static` on May 19, 2020

Conversation 39 | Commits 2 | Checks 0 | Files changed 14

dims commented on May 16, 2020 • edited — Member

What type of PR is this?
/kind feature

What this PR does / why we need it:
This is similar to #90674 where we switched apiserver/scheduler to distroless. In this case etcd image has scripts, the idea is to use the `bash-static` utility (instead of the FULL debian-base image) inside the distroless image and cleanup a bunch of things to get this to work:

Kube Proxy


giphy.com
#kimsconvenience
Everything is perfect the way it is!

Kube Proxy

TL;DR kube-proxy needed IPtables

So it had to continue to use Debian upstream image.

As a result it is now the image that is most often updated :-)

What did we learn from this?

Release train rarely looks like this

It can often looks like this

But an imperfect solution still has benefits


```
pjoglekar@pjoglekar-a01 demo % trivy image
gcr.io/google-containers/kube-apiserver:v1.16.9

gcr.io/google-containers/kube-apiserver:v1.16.9 (debian 9.12)
=============================================================
Total: 167 (UNKNOWN: 1, LOW: 93, MEDIUM: 26, HIGH: 28, CRITICAL: 19)
```

```
pjoglekar@pjoglekar-a01 demo % trivy image
k8s.gcr.io/kube-apiserver:v1.22.1

k8s.gcr.io/kube-apiserver:v1.22.1 (debian 10.10)
================================================
Total: 0 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 0, CRITICAL: 0)
```

Ref: Snippet from https://github.com/aquasecurity/trivy scan result

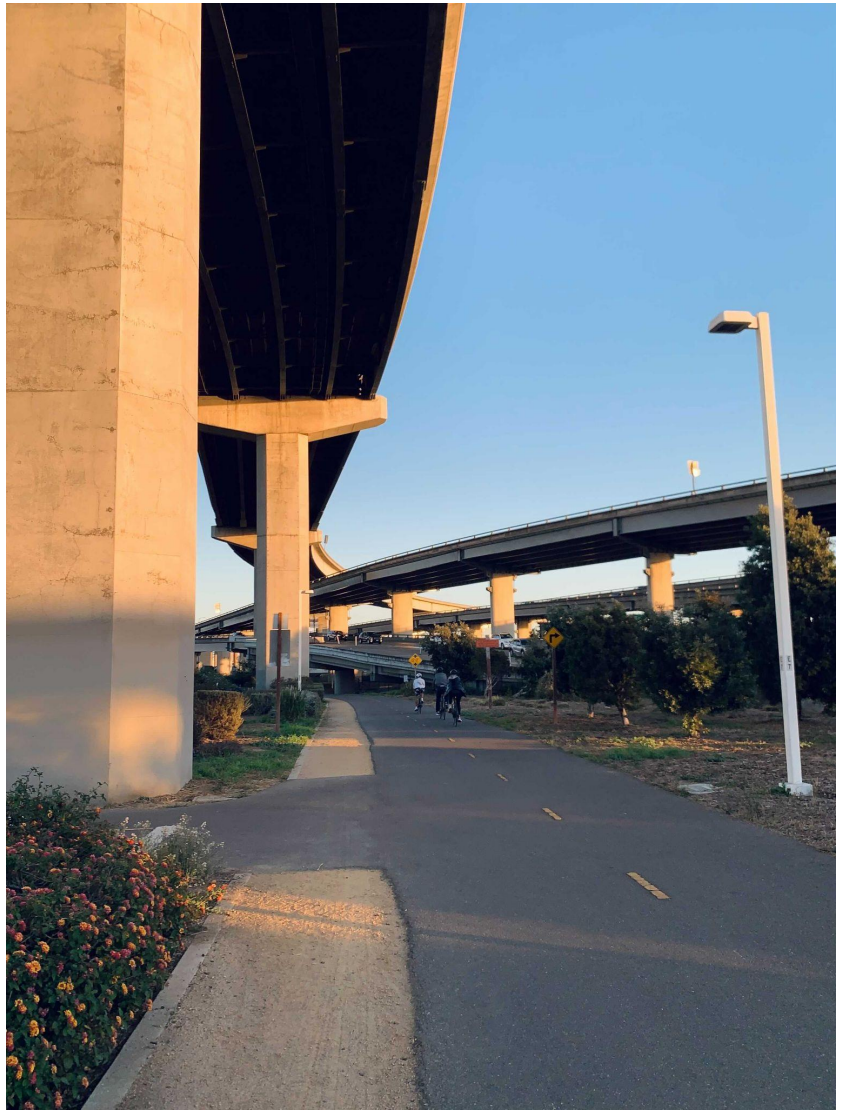
None of the vulnerable components were needed

Imagine the reduced churn in fixing these vulnerabilities

What to do when distroless is not an option?

Focus on fixable CVEs

What to do when distroless is not an option?



Focus on whether the vulnerability is in code execution path or not

What to do when distroless is not an option?

Focus on improving automation to rebuild, ship and test updated base images

What to do when distroless is not an option?

Create a list of images that need special care / frequent updates e.g. kube-proxy

What's next in Kubernetes?

- ➔ Defining a policy and process to triage ***known*** vulnerabilities
- ➔ Automated Periodic scans for ***known*** vulnerabilities:
  - ◆ In Images and
  - ◆ Build time dependencies
- ➔ Feedback or Suggestions welcome!!
  - ◆ #sig-security-tooling on Kubernetes Slack

Manage Vulnerabilities, by *Being* Vulnerable

- Identify base images and standardize them
- Learn limitations of vulnerability scanners
- Understand the efforts to remediate

... Most importantly model threats and assess risk

Questions?

Thank you!