Improving Resilience Through Service Mesh Metrics

DevSecOps Days Pittsburgh 2021

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM21-0567

This talk will cover an introduction to…

- Resilience
- Metrics
- Service Mesh
- Kubernetes automation
- Prometheus
- Basic scripting concepts

We have a lot of territory to cover.

Resilience

“an ability to recover from or adjust easily to misfortune or change” –Merriam-Webster

- Resilience is also about survival
- Recovery is variable
- Adjustment is variable
- “Easily” is the measure of how resilient something is

Resilience is not easy.

warhistoryonline.com– B-17 bombers that miraculously made it home

Metrics

“a standard of measurement” –Merriam-Webster

Metrics are how we measure progress.

Things we can measure in DevSecOps: Disk IOPS, Network throughput, IO Latency, CPU Utilization, Memory Utilization, requests per second, response latency, authorizations/failed auths/s, bad requests/s, uptime, MTTR, commits per user, deploys per day, bugs discovered/fixed, security vulns discovered/fixed, protocol overhead, and an unlimited number of additional derivative metrics.

We can measure anything, but what should you be measuring and why and for whom?

What metrics are useful for measuring and improving resilience?

Service Mesh

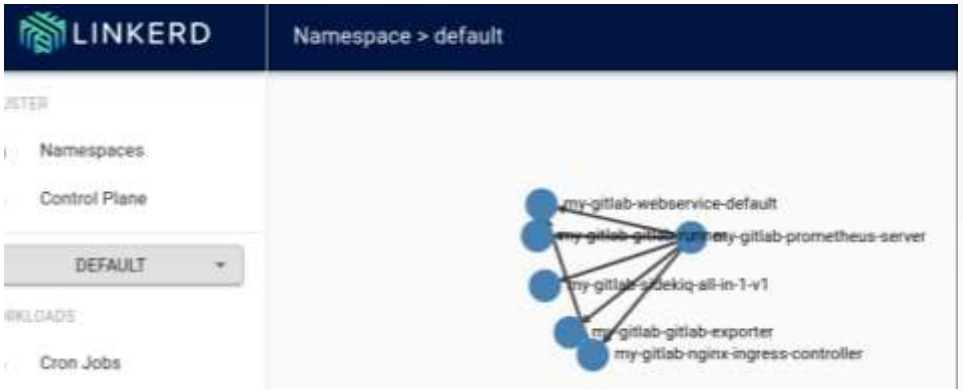
"... makes running services easier and safer by giving you runtime debugging, observability, reliability, and security…" –LinkerD

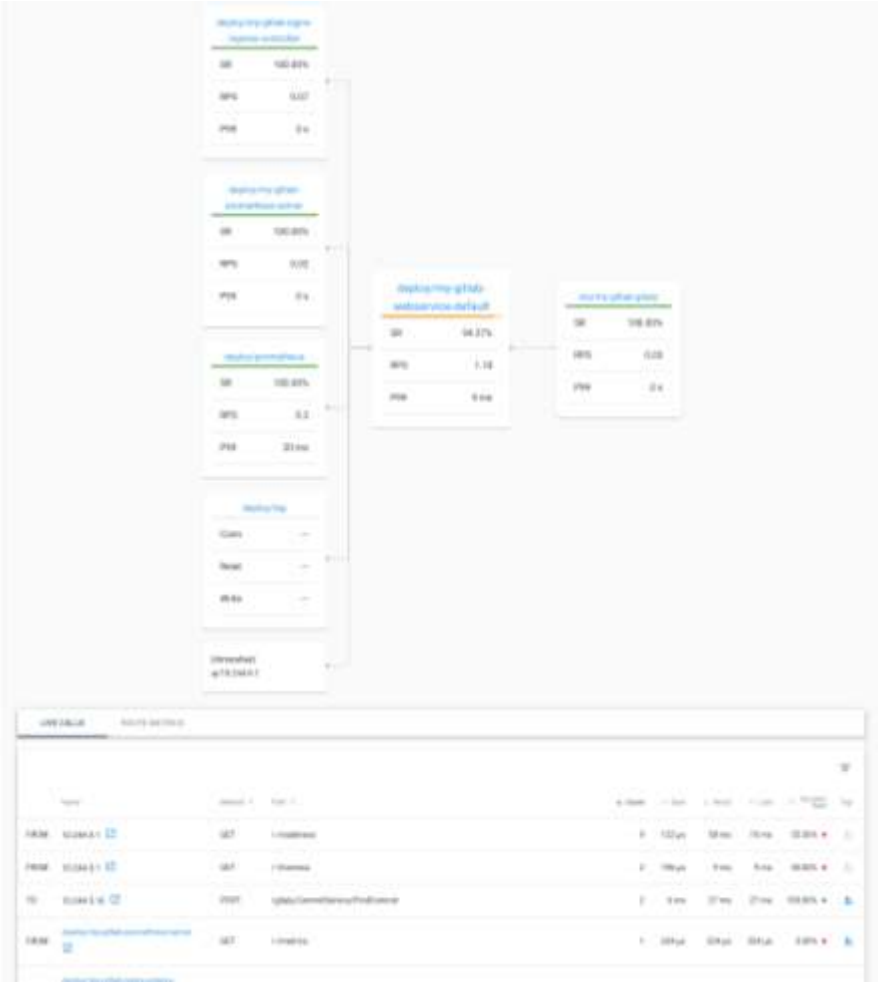
- The service mesh exists at the interface between co-operating services, providing all of the above properties to your existing applications. For us, they provide a lot useful data. (for creating metrics)
- Technology enablers:
  - Kubernetes' "pod" concept
  - Transparent, inject-able proxies that modify communications pathways at the edge of the pod

Service mesh proxies perform useful transformations on your data, including encryption, service instance routing, HTTP trace logging, and many others.

Service Mesh with Kubernetes

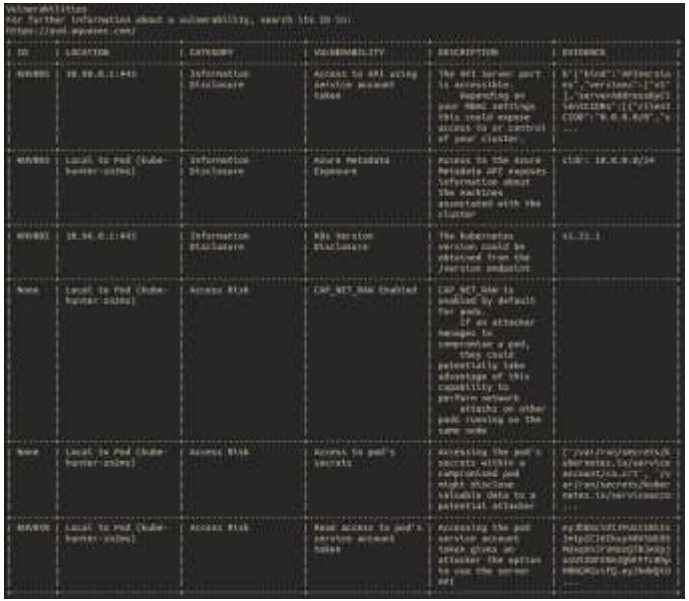
- Install your application services as normal
- Install and configure your mesh
- “Inject” the mesh into your application
- Gain insights into your operations





Where does the data come from?

- For the purposes of this discussion, the data comes from the individual services and service mesh proxies, and is reported to a central logging service, called Prometheus.
- Novel metrics can be created from data sources that are not normally collected by the mesh.
  - Application Service logs
  - Security/Access logs
  - Git Commit logs
  - Many others



"Kube Hunter" – Kubernetes' threat report

The data comes from wherever it needs to come from, in order to generate useful metrics.

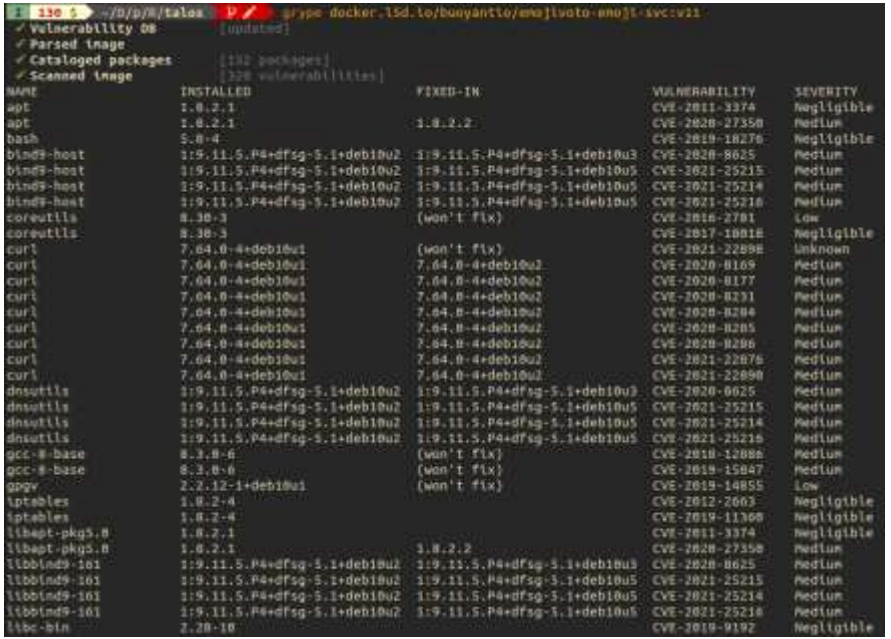
Plenty of data – How to use it?

You need useful user stories.

“As a security professional, I need to know how many failed authorizations we are seeing each day, so that I know when we are facing a specific kind of threat actor.”

“As a developer, I need to know what vulnerabilities exist in the dependencies my code relies on, so that we deliver safe software.”

“As an automation engineer, I need to know when the system is experiencing abnormal loads, so I can decide whether or not to scale the infrastructure.”



Anchore’s Grype tool, reporting known package vulnerabilities in a running container instance.

Automation with Kubernetes

Containers ⊆ Pods ⊆ Services ⊆ Deployments ⊆ Charts ⊆ Applications

- Kubernetes is known for adding the concepts of Pods, Services, and Deployments on top of Containers
  - Containers are the building blocks of the Application
  - Pods are the smallest deployable unit in Kubernetes (unit of organization of containers)
  - Services abstract features of your containers, such as the Service IP or endpoint
  - Deployments allow abstract reasoning about the runtime state of Services for the purposes of scaling
  - Helm Charts (and related technologies) allow you to reason about your applications, as if they were a set of packages that work together in a specific configuration
  - Applications are made up of potentially numerous Charts, configured to provide a full user experience or capability

Jobs are the Kubernetes janitor

The basic unit of cluster work is the Job

When the cluster needs to run some meta-task that isn't part of the application, it deploys a Job.

As containers are the building blocks of the Application/Cluster, Jobs are built on top of Containers

Jobs can be built for any purpose. We will use them for collecting and reporting metrics.

```yaml
---
apiVersion: batch/v1
kind: Job
metadata:
  name: kube-hunter
spec:
  template:
    spec:
      containers:
        - name: kube-hunter
          image: aquasec/kube-hunter
          command: ["kube-hunter"]
          args: ["--pod"]
      restartPolicy: Never
  backoffLimit: 4
```

You had one job.

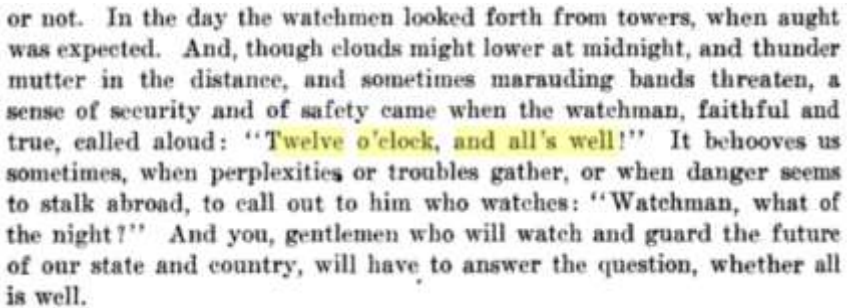
```
kubectl --kubeconfig ./kubeconfig describe jobs kube-hunter
Name:           kube-hunter
Namespace:      default
Selector:       controller-uid=9ddbac51-a225-4eab-b8a5-0ce48cb7d900
Labels:         controller-uid=9ddbac51-a225-4eab-b8a5-0ce48cb7d900
                job-name=kube-hunter
Annotations:    <none>
Parallelism:    1
Completions:    1
Start Time:     Tue, 15 Jun 2021 02:34:54 +0000
Pods Statuses:  1 Running / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  controller-uid=9ddbac51-a225-4eab-b8a5-0ce48cb7d900
           job-name=kube-hunter
  Containers:
   kube-hunter:
    Image:      aquasec/kube-hunter
    Port:       <none>
    Host Port:  <none>
    Command:
      kube-hunter
    Args:
      --pod
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Events:
  Type    Reason           Age   From                    Message
  ----    ------           ----  ----                    -------
  Normal  InjectionSkipped  19s   linkerd-proxy-injector  Linkerd sidecar proxy injection skipped
  Normal  SuccessfulCreate  19s   job-controller          Created pod: kube-hunter-zn2ms
```

This Job produced the Kube-Hunter logs from slide 7.

Jobs can be repeated (Cron Jobs)

```yaml
apiVersion: batch/v1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "0 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: hello
            image: busybox
            imagePullPolicy: IfNotPresent
            command:
            - /bin/sh
            - -c
            - date +"%H:%M"; echo "O'clock and all is well."
          restartPolicy: OnFailure
```

A simple cron job illustrates the point nicely. We can use the same approach to collect and report metrics



or not. In the day the watchmen looked forth from towers, when aught was expected. And, though clouds might lower at midnight, and thunder mutter in the distance, and sometimes marauding bands threaten, a sense of security and of safety came when the watchman, faithful and true, called aloud: "Twelve o'clock, and all's well!" It behooves us sometimes, when perplexities or troubles gather, or when danger seems to stalk abroad, to call out to him who watches: "Watchman, what of the night?" And you, gentlemen who will watch and guard the future of our state and country, will have to answer the question, whether all is well.

Creating custom automations

```
~/D/p/R/talos  grype docker.l5d.io/buoyantio/emojivoto-emoji-svc:v11 -o json | jq '.matches[].vulnerability | select(.severity == "High")' > results.json
✓ Vulnerability DB        [no update available]
✓ Parsed image
✓ Cataloged packages      [152 packages]
✓ Scanned image           [320 vulnerabilities]
```

JSON Data is extremely common. Get used to working with it.

- A popular data source for basic metrics reporting for cluster applications today is Prometheus. (Next slide)
- Prometheus provides a number of clients for connecting to the service, reporting and querying data.
- Clients are available for various languages and can be used to create various forms of metrics and automation jobs.


```
{
  "id": "CVE-2021-3520",
  "dataSource": "https://security-tracker.debian.org/tracker/CVE-2021-3520",
  "namespace": "debian:10",
  "severity": "High",
  "urls": [
    "https://security-tracker.debian.org/tracker/CVE-2021-3520"
  ],
  "cvss": [],
  "fix": {
    "versions": [
      "1.8.3-1+deb10u1"
    ],
    "state": "fixed"
  },
  "advisories": [
    {
      "id": "DSA-4919-1",
      "link": "https://security-tracker.debian.org/tracker/DSA-4919-1"
    }
  ]
}
~/D/p/R/talos  cat results.json | jq '.id' | count
14
```


Creating a metric to count current “High” CVEs.

ConfigMaps, Custom Containers, and Scripts

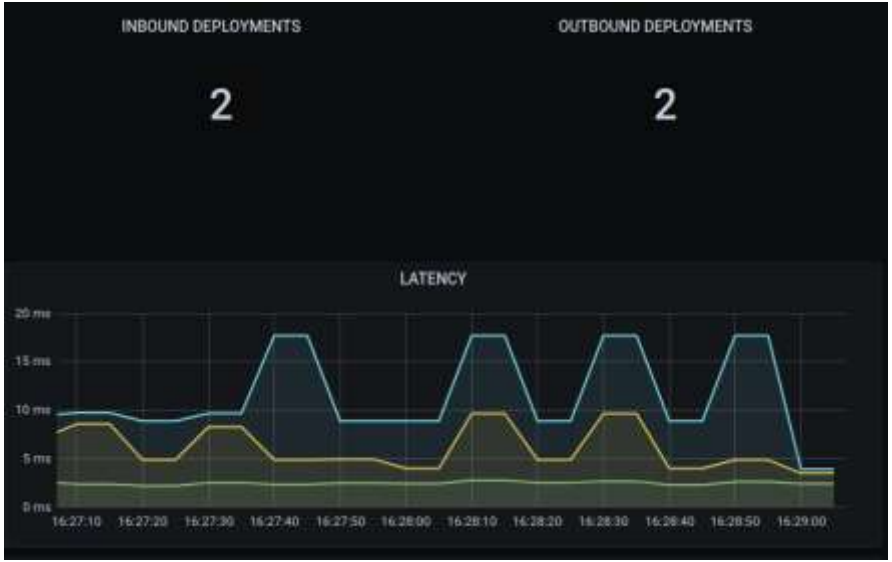
- ConfigMaps are a useful place to store simple scripts that can be executed from an off-the-shelf container.
- Custom containers provide ultimate flexibility for running your jobs.
  - Implement a Prometheus client metric in the language of your choice
  - Place it in a container or a ConfigMap
  - Create a job to run it
  - Create a Cron job to schedule it
  - Report results and/or push log files to your collection point.

```python
from prometheus_client import start_http_server, Summary
import random
import time

# Create a metric to track time spent and requests made.
REQUEST_TIME = Summary('request_processing_seconds', 'Time spent processing request')

# Decorate function with metric.
@REQUEST_TIME.time()
def process_request(t):
    """A dummy function that takes some time."""
    time.sleep(t)

if __name__ == '__main__':
    # Start up the server to expose the metrics.
    start_http_server(8000)
    # Generate some requests.
    while True:
        process_request(random.random())
```

A sample script, a starting point for developing a custom metric.

Displaying Metrics

- User engagement is crucial in developing a visual form for your data that gets the users the information they need, in a format that has meaning and facilitates decisions.
- Many tools to choose from for displaying information.
- The “dashboard” is a common tool.



You could visualize your metrics like this, but does it convey the necessary message to your user?

Never lose sight of the target for the metrics and the automations you create.

The end