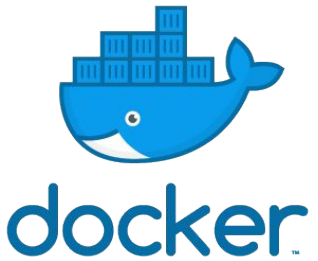
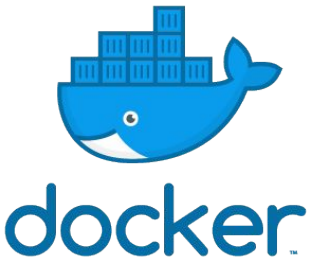
Securing Telemetry & Tracing with SPIFFE & Envoy

Sabree Blackmon
@HeavyPackets

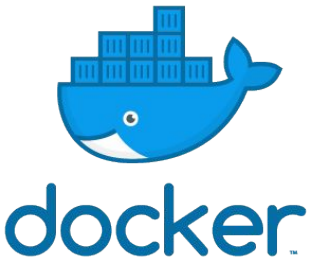
Our apps are generating a literal shit ton of data



Operational data, such as performance metric and infrastructure or systems logs

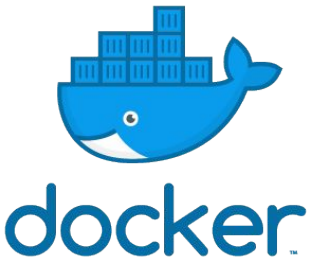


Some of this data is *auditable*, such as application access logs

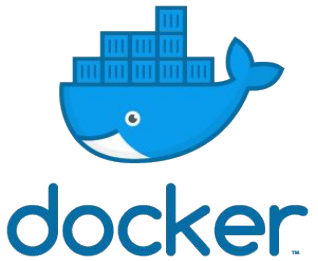



Charity Majors
@mipsytipsy
Follow
let's talk about two kinds of data that systems tend to generate: auditable and operational.
* auditable data: transaction logs, replication logs, billing/finance events etc
* operational data: telemetry, metrics and context that describe each request and system component
1:06 AM - 22 Aug 2018
77 Retweets 251 Likes
5
77
251

Engineers, but also auditors, InfoSec teams and BI folks, leverage this data for a diverse set of concerns



But, what happens when this observability data leaks?




Scott Helme
@Scott_Helme
Follow
When logging causes security incidents;
What we learned from GitHub and Twitter
Report URI
When logging causes security incidents; What we learned from GitHub and T...
You may have seen the news recently about Twitter and GitHub having accidentally logged sensitive information like passwords in their applications or server logs. Lo...
scotthelme.co.uk
10:08 AM - 25 May 2018

Common Weakness Enumeration

A Community-Developed List of Software Weakness Types

CWE and SANS Institute TOP 25 MOST DANGEROUS SOFTWARE ERRORS

CWE-532: Information Exposure Through Log Files

Weakness ID: 532
Abstraction: Variant
Structure: Simple

Status: Incomplete

Presentation Filter: Basic

Description

Information written to log files can be of a sensitive nature and give valuable guidance to an attacker or expose sensitive user information.

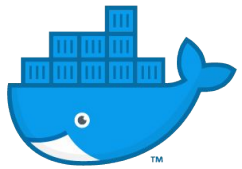
Extended Description

While logging all information may be helpful during development stages, it is important that logging levels be set appropriately before a product ships so that sensitive user data and system information are not accidentally exposed to potential attackers.

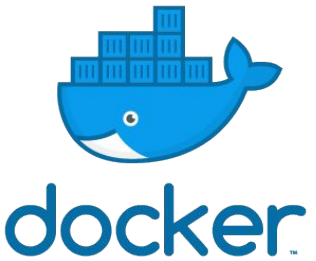
Different log files may be produced and stored for:

- Server log files (e.g. server.log). This can give information on whatever application left the file. Usually this can give full path names and system information, and sometimes usernames and passwords.
- log files that are used for debugging

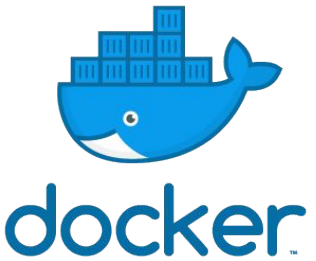
Relationships



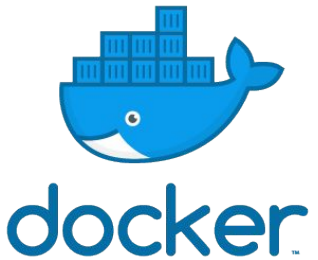
We have a great set of open-source observability tools within the CNCF...



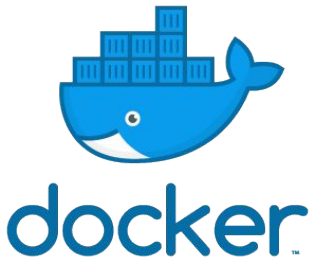
...but there isn't much guidance on how to protect this data in transit



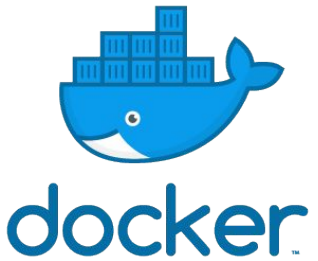
This data often needs to be secured *at rest* *too*



What are some types of observational data?

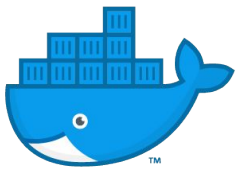


Time-series Metrics



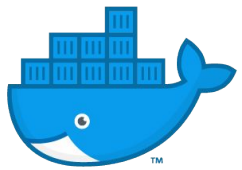
Time-series metrics (Telemetry)

- Often used for expose runtime performance stats such as memory, CPU load, disk IO, and service request/response metrics
  - This information can be useful in determining how apps behave in response to API events, time of day, etc



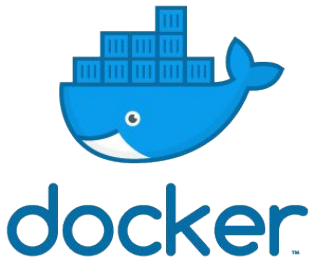
Time-series metrics (Telemetry)

- But can be used generically for any kind of metric your applications care about
  - In our example app, we use time-series data for application access/security statistics

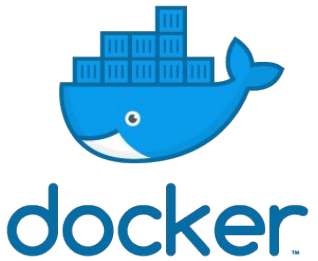


Text Logging

(structured or unstructured)

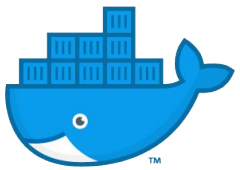


Logs give engineers context specific data about events within an application

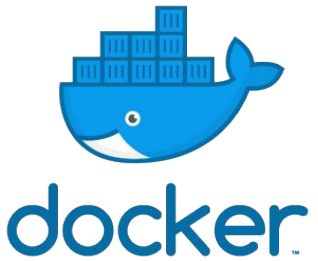


Logging

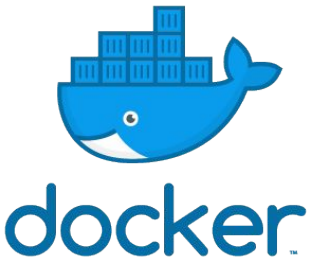
- Some of this logging is relevant for app debugging purposes
- Some of this logging is needed to audit ACLs or security events, data reading/writing, etc



Tracing

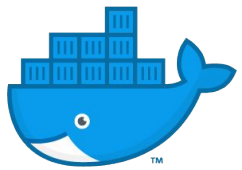


Often seen as an extension to logging, it *really* isn't



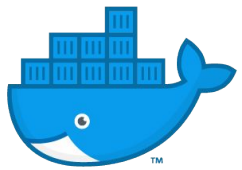
Tracing

- Bring together app events across a distributed call stack but also tie in infrastructure specific data into each trace
  - Proxies (like Envoy) can contribute data to the trace
  - Things like IP addresses or host information are exposed

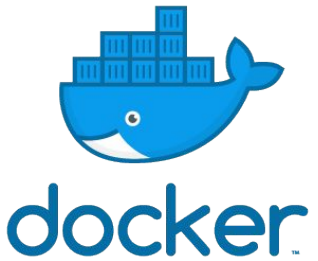


Tracing

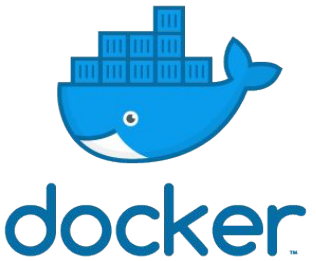
- Important app context specific information *can* be added to traces
  - BUT, there is no implied data sanitation or transport safety across *trace baggage*



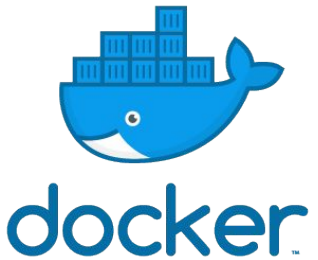
TL;DR



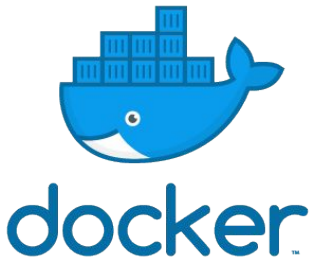
These tools are all great for debugging and observability purposes...



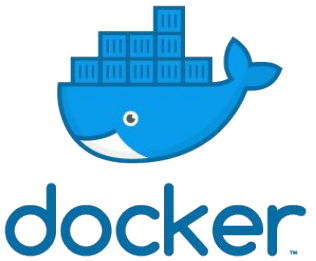
...but there are important AppSec & InfoSec concerns around *all* of this data



Bad actors can gain valuable introspection into your how applications behave



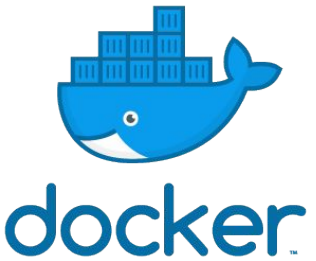
The cost of info leaks increases the more *useful* we make our data to engineers



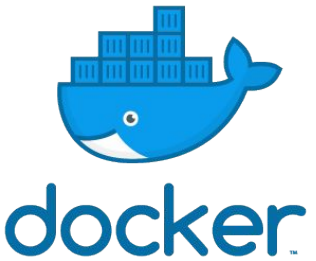
Case Study

Example App: A Simple Secrets Server

Simple Rust HTTP API in front of etcd, in which you can store text “secrets”



Simple username & password login, token based API access

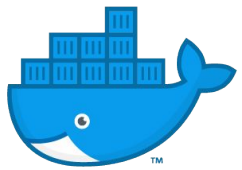


simple-secrets

Metrics (Prometheus) endpoint

Provides a metrics endpoint, that Prometheus scrapes (pulls), that exposes application access statistics

- # of valid logins
- # of invalid logins
- # secrets set
- # of sets accessed

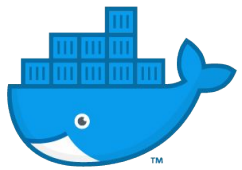


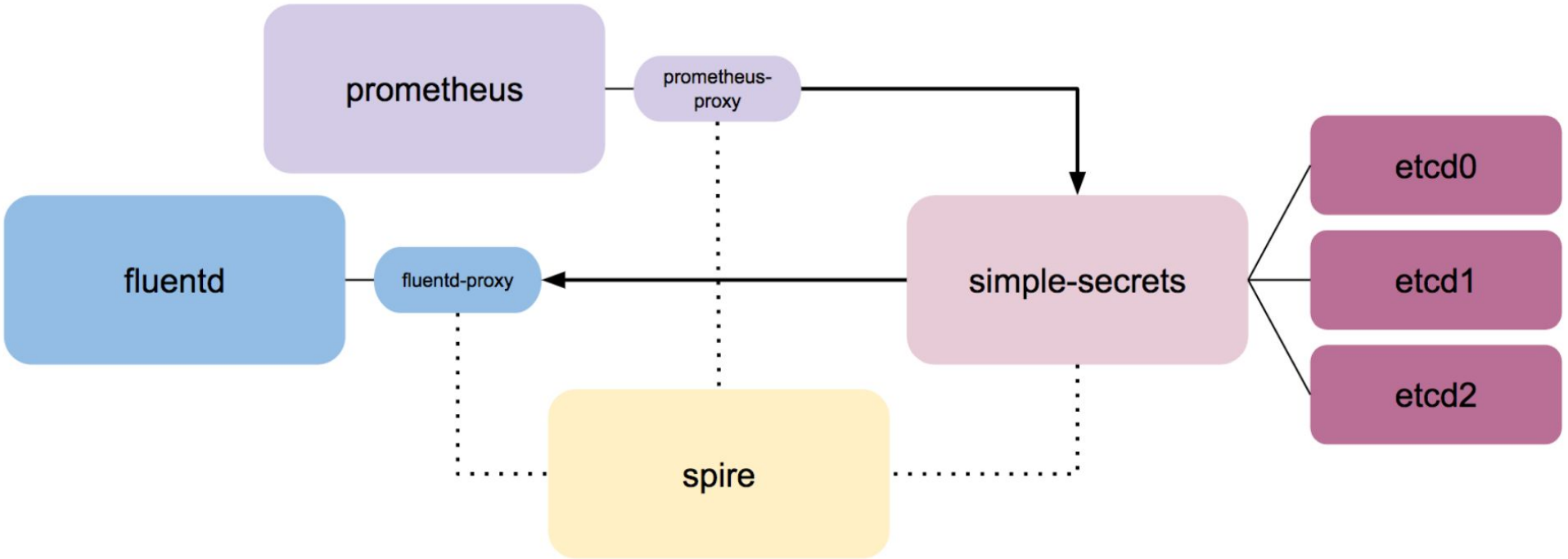
simple-secrets

Logging (Fluentd)

Pushes auditable events to Fluentd:

- Logs access events and system errors in detail
  - As such, valid and invalid user identities are exposed



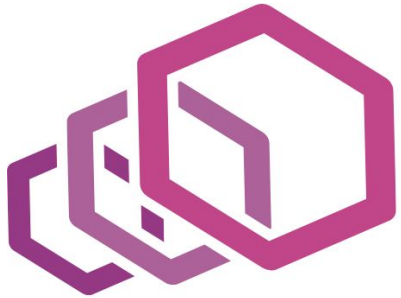

prometheus
prometheus-proxy
fluentd
fluentd-proxy
simple-secrets
etcd0
etcd1
etcd2
spire

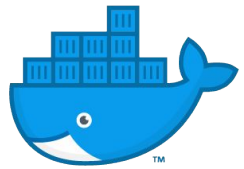
docker

simple-secrets

Envoy & SPIFFE

- SPIFFE SVIDs (x509 certificates) are securely issued through a process called *attestation* with a SPIRE agent
  - SPIFFE identity are tied to a specific host system
  - Regularly rotated automatically

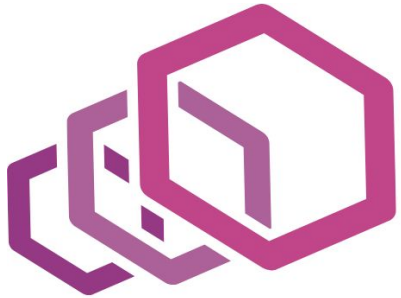


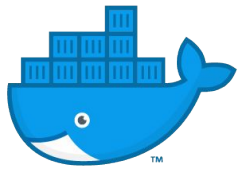


simple-secrets

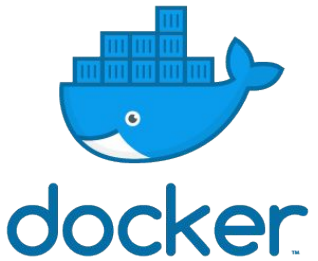
Envoy & SPIFFE

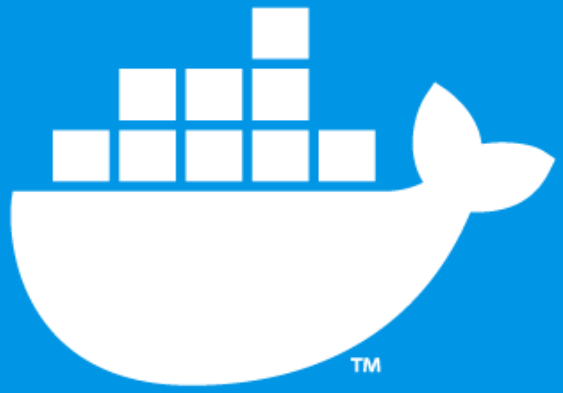
- These SVIDs are used by Envoy to establish TLS tunnels between the app and Prometheus + Fluentd
  - The app knows where data is going to, and the tooling knows what app instance is providing the telemetry





https://github.com/heavypackets/simple-secrets





THANK YOU :)