

Hacking Monitoring for Fun and Profit

CNCF

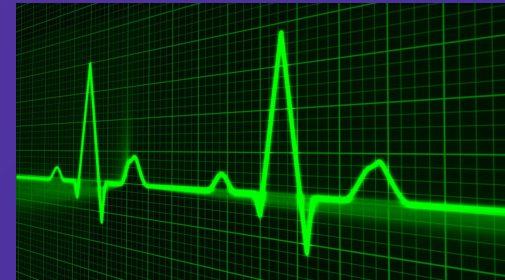
Omer Levi Hevroni | AppSec Engineer @ Snyk | @omerlh

Agenda

Threat Modeling



Monitoring

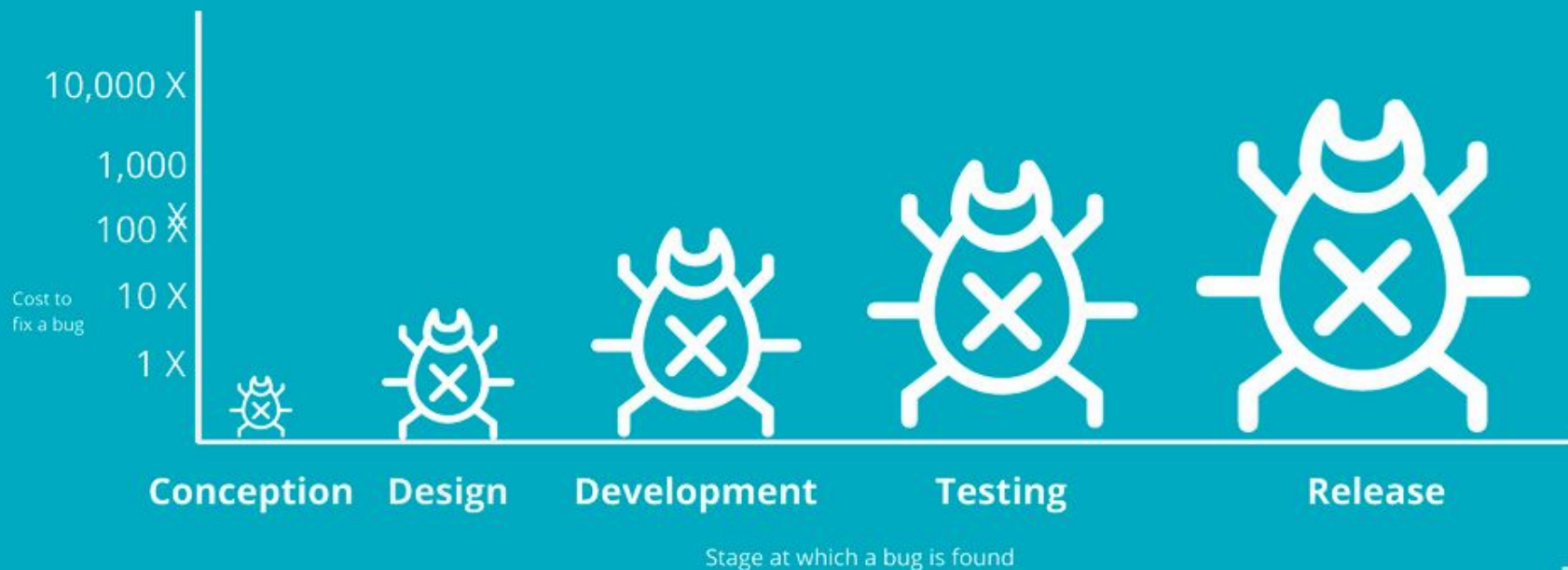


What we are going to discuss

- What is Threat Modeling?
- What is Monitoring?
- How hackers can leverage our monitoring systems?
 - AKA Threat modeling our monitoring systems

A. Threat Modeling

Resolving bugs early and often reduces associated costs



Threat Modeling Manifesto

- What are we building?
- What can go wrong?
- What are we doing about it?
- Did we do a good job?

Threat Modeling Manifesto

Systematic Approach

Achieve thoroughness and reproducibility by applying security and privacy knowledge in a structured manner.

Informed Creativity

Allow for creativity by including both craft and science.

Varied Viewpoints

Assemble a diverse team with appropriate subject matter experts and cross-functional collaboration.

Useful Toolkit

Support your approach with tools that allow you to increase your productivity, enhance your workflows, enable repeatability and provide measurability.

Theory into Practice

Use successfully field-tested techniques aligned to local needs, and that are informed by the latest thinking on the benefits and limits of those techniques.

Threat Modeling Manifesto

These anti-patterns inhibit threat modeling:

Hero Threat Modeler

Threat modeling does not depend on one's innate ability or unique mindset; everyone can and should do it.

Admiration for the Problem

Go beyond just analyzing the problem; reach for practical and relevant solutions.

Tendency to Overfocus

Do not lose sight of the big picture, as parts of a model may be interdependent. Avoid exaggerating attention on adversaries, assets, or techniques.

Perfect Representation

It is better to create multiple threat modeling representations because there is no single ideal view, and additional representations may illuminate different problems.

B. Monitoring



**Detect issues before customers
experience them**



O'REILLY®

A detailed black and white illustration of a lizard, possibly a monitor lizard, is positioned horizontally across the top half of the cover. The lizard's body is textured with scales and spots, and its long tail curves downwards. It appears to be resting on a blue rectangular area that contains the title text.

Site Reliability Engineering

HOW GOOGLE RUNS PRODUCTION SYSTEMS

Edited by Betsy Beyer, Chris Jones,
Jennifer Petoff & Niall Murphy

**Black Box
Monitoring**

**White Box
Monitoring**



Let's Dive In!

C. Threat Modeling our Monitoring!

White Box Monitoring



Threat Modeling

- What are we building?
- What can go wrong?
- What are we doing about it?
- Did we do a good job?

What are we building?

“

Monitoring based on metrics exposed by the internals of the system, including logs, interfaces like the Java Virtual Machine Profiling Interface, or an HTTP handler that emits internal statistics.

”

White-box Monitoring

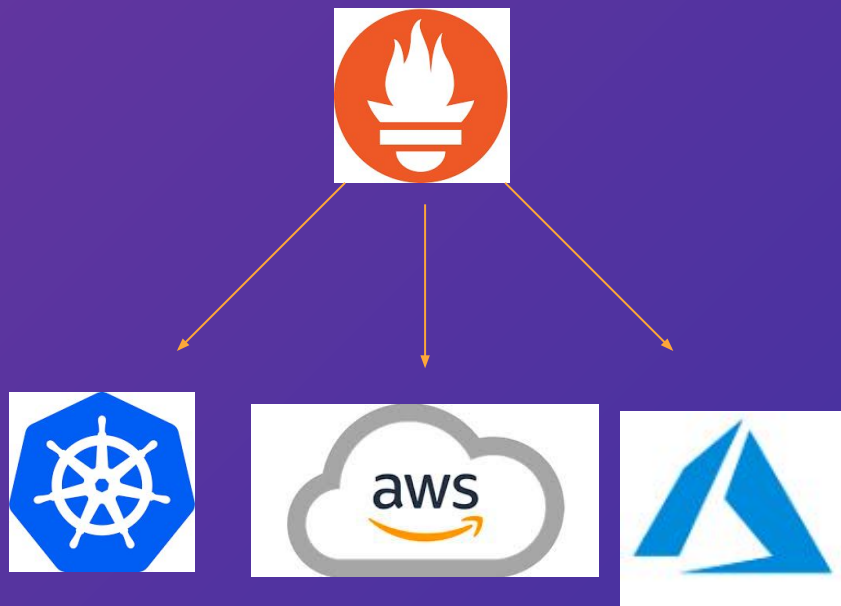
Prometheus



- Monitoring system
- Time-Series database
- Alerting system
- Auto-discovery

<https://prometheus.io/>

Prometheus Scraping Model



Exposing Metrics

```
go_gc_duration_seconds{quantile="0"} 0.0001939  
go_gc_duration_seconds{quantile="0.25"} 0.000289999  
go_gc_duration_seconds{quantile="0.5"} 0.000342298  
go_gc_duration_seconds{quantile="0.75"} 0.000465998  
go_gc_duration_seconds{quantile="1"} 0.307486594
```

Metric's Name

Labels

Value



What Can go wrong?

Information Disclosure

JS index.js

```
1  const express = require('express');
2  const server = express();
3  const promClient = require('prom-client')
4  const register = promClient.register;
5
6  server.get('/metrics', (req, res) => {
7    res.set('Content-Type', register.contentType);
8    res.end(register.metrics());
9  });
10
11  promClient.collectDefaultMetrics();
12
13  server.listen(3000);
```

Some Interesting Metrics

```
# HELP nodejs_version_info Node.js version info.  
# TYPE nodejs_version_info gauge  
nodejs_version_info{version="v11.10.0",major="11",minor="10",patch="0"} 1
```

```
# HELP nodejs_eventloop_lag_seconds Lag of event loop in seconds.  
# TYPE nodejs_eventloop_lag_seconds gauge  
nodejs_eventloop_lag_seconds 0.013998085 1556457528024
```

Vulnerability DB

Detailed information and remediation guidance for known vulnerabilities.

Find out if you have vulnerabilities that put you at risk

Test your code

nodejs

Search

any

cocoapods

Composer

Go

Linux

Maven



npm

NuGet

pip

RubyGems

Report a new vulnerability

VULNERABILITY	AFFECTS	TYPE	PUBLISHED
H  Improper Input Validation	nodejs <10.19.0-r0	alpine:3.9	21 Jul, 2020
H  Improper Input Validation	nodejs <12.15.0-r0	alpine:3.12	21 Jul, 2020
H  Improper Input Validation	nodejs <12.15.0-r0	alpine:3.11	21 Jul, 2020
H  Improper Input Validation	nodejs <10.19.0-r0	alpine:3.10	21 Jul, 2020
H  HTTP Request Smuggling	nodejs <10.19.0-r0	alpine:3.9	21 Jul, 2020
H  HTTP Request Smuggling	nodejs <12.15.0-r0	alpine:3.12	21 Jul, 2020
H  HTTP Request Smuggling	nodejs <12.15.0-r0	alpine:3.11	21 Jul, 2020
H  HTTP Request Smuggling	nodejs <10.19.0-r0	alpine:3.10	21 Jul, 2020
H  HTTP Request Smuggling	nodejs <10.19.0~dfsg-1	debian:unstable	08 Feb, 2020

What about this metric?

```
# HELP requests_by_client count the requests by source client (using x-api-client header)
# TYPE requests_by_client counter
requests_by_client{ip="movie-api"} 1
requests_by_client{ip="subscription-api"} 1
requests_by_client{ip="authorization-api"} 1
```

The code behind it

```
const gauge = new promClient.Counter({
  name: 'requests_by_client',
  help: 'count the requests by source client (using x-api-client header)',
  labelNames: ['ip']
});

server.get('/hey', (req, res) => {
  gauge.labels(req.headers['x-api-client'] || req.connection.remoteAddress).inc();

  res.send('hey');
});
```



Vulnerability DB

Detailed information and remediation guidance for known vulnerabilities.

Find out if you have vulnerabilities that put you at risk

Test your code

🔍 prometheus

Search

any cocoapods Composer Go Linux Maven npm NuGet pip RubyGems

Report a new vulnerability

VULNERABILITY	AFFECTS	TYPE	PUBLISHED
L Cross-site Scripting (XSS)	prometheus *	ubuntu:18.10	07 Feb, 2019
L Cross-site Scripting (XSS)	prometheus *	ubuntu:18.04	07 Feb, 2019
M Cross-site Scripting (XSS)	prometheus <2.7.1+ds-1	debian:unstable	07 Feb, 2019
M Cross-site Scripting (XSS)	prometheus <2.7.1+ds-1	debian:11	07 Feb, 2019
M Cross-site Scripting (XSS)	prometheus <2.7.1+ds-1	debian:10	07 Feb, 2019
H ELSA-2020-5827	prometheus <0:2.13.1-1.0.3.e17	oracle:7	31 Aug, 2020
H Server-Side Request Forgery (SSRF)	golang-github-prometheus-node_exporter <0.18.1-1.6.2	sles:12.5	17 Sep, 2020
H Server-Side Request Forgery (SSRF)	golang-github-prometheus-node_exporter <0.18.1-1.6.2	sles:12.4	17 Sep, 2020
H Server-Side Request Forgery (SSRF)	golang-github-prometheus-node_exporter <0.18.1-1.6.2	sles:12.3	17 Sep, 2020



Closed

Permission for secrets and configmaps - Kube-Prometheus #2064

jakirpatel opened this issue on Oct 29, 2018 · 10 comments

**squat** closed this on Nov 5, 2018**omerlh** commented on Dec 23, 2019

I think it worth discussing it again - permission to read all the secrets in the entire cluster is very dangerous, and makes the operator a very sensitive workload. Maybe now we can revise this decision? If the permission is only needed for Prometheus, can we have the operator watch for only specific namespaces (where Prometheus is installed) for secrets, and not the entire cluster? Also, why does the operator need permissions to delete services/endpoints?

**brancz** commented on Jan 7

Contributor



Agreed and it's already tracked in [#2186](#)

**omerlh** commented on Jan 7

Thanks, I'll follow this thread. For now, I changed the rule manually to make it work - I can maybe open a PR on kube-prometheus if this will be helpful



Michael Thomas (gnarlygoat)

170

Reputation

-

Rank

6.36

Signal

93rd

Percentile

1

#764731

Publicly accessible Grafana install allows pivoting to Prometheus datasource

Share:

State ● Resolved (Closed)Severity ■ High (7 ~ 8.9)Disclosed **May 14, 2020 8:11pm +0300**

Participants

Reported To [U.S. Dept Of Defense](#)Visibility **Disclosed (Full)**Reported at **December 26, 2019 2:27pm +0200**Weakness **Information Disclosure**[Collapse](#)

TIMELINE



gnarlygoat submitted a report to [U.S. Dept Of Defense](#).

Dec 26th (11 months ago)

Summary:

A publicly accessible Grafana install exposes semi sensitive Dashboards. This also exposes the Prometheus proxied datasources which allow direct queries to a Prometheus instance which reveals sensitive data and opens the instance up to potential DoS via crafted requests.

Description:

TOTAL RESULTS

5,777

TOP COUNTRIES



United States	2,921
Germany	505
United Kingdom	375
Singapore	357
Netherlands	236

TOP SERVICES

HTTPS	2,940
HTTP	2,032
8086	160
10250	89
9090	51

TOP ORGANIZATIONS

Choopa, LLC	1,634
Digital Ocean	1,209
Amazon.com	977
Linode	397
Hetzner Online GmbH	183

TOP OPERATING SYSTEMS

Windows 6.1	3
-------------	---

TOP PRODUCTS

nginx	652
InfluxDB	162
Kubernetes	80

New Service: Keep track of what you have connected to the Internet. Check out [Shodan Monitor](#)

301 Moved Permanently

135.181.96.40
static.40.96.181.135.clients.your-server.de
Hetzner Online GmbH
Added on 2020-11-15 07:11:40 GMT
Germany

HTTP/1.1 301 Moved Permanently
Date: Sun, 15 Nov 2020 07:11:40 GMT
Content-Type: text/html
Content-Length: 162
Connection: keep-alive
Location: https://135.181.96.40/
X-Frame-Options: sameorigin
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Server: Prometheus
Pre-Cogni...

135.181.96.40

static.40.96.181.135.clients.your-server.de
Hetzner Online GmbH
Added on 2020-11-15 07:11:42 GMT
Germany

SSL Certificate

Issued By:
|- Common Name: **Let's Encrypt**
Authority X3
|- Organization: **Let's Encrypt**
Issued To:
|- Common Name: **amoslaki.fi**

Supported SSL Versions

TLSv1.2, TLSv1.3

HTTP/1.1 301 Moved Permanently
Date: Sun, 15 Nov 2020 07:11:42 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Redirect-By: WordPress
Location: https://amoslaki.fi/
X-Frame-Options: sameorigin
X-Content-Type-Options: nosniff
X-XSS-Protection...

47.101.174.209

Hangzhou Alibaba Advertising Co.,Ltd.
Added on 2020-11-15 07:15:21 GMT
China

devops

SSL Certificate

Issued By:
|- Common Name: **10.212.1.218-ca@1542901833**
Issued To:
|- Common Name: **10.212.1.218@1542901833**

Supported SSL Versions

TLSv1.2

Kubernetes:

Node #1:
Name: **ams-x-transcode-uat-54bbdb964-fzwtq**
Container #1:
Name: **ams-x-transcode**
Image: **10.212.1.235/opgcloud/ams-x-transcode:2.0.16.20200103**
Node #2:
Name: **exporter-node-cluster-monitoring-4kgH5**
Container #1:
Name: **exporter-node**
Ima...

What Can We Do About It?

Block Access (nginx ingress)

<https://www.edureka.com/community/19277/access-some-specific-paths-while-using-kubernetes-ingress?show=19278#a19278>

```
! ingress.yaml
```

```
1  apiVersion: extensions/v1beta1
2  kind: Ingress
3  metadata:
4    name: block-metrics
5    annotations:
6      nginx.ingress.kubernetes.io/configuration-snippet: |
7        server_tokens off;
8        location /metrics {
9          deny all;
10         return 403;
11       }
12
13  spec:
14    rules:
15      - host: bar.foo.com
16        http:
17          paths:
18            - backend:
19              serviceName: my-service
20              servicePort: 80
```

Prometheus Metrics Limit

```
- job_name: 'kubernetes-services'  
  sample_limit: 2000  
  metrics_path: /probe  
  params:  
    module: [http_2xx]  
  
  kubernetes_sd_configs:  
    - role: service
```

Other Mitigations

- **Check for known vulns**
- **Review manifests files before deployment**
- **Authentication/VPN**
 - Grafana has built-in authentication
 - Use products like oauth proxy for all the rest

Did we do a good job?

**Black Box
Monitoring**

**White Box
Monitoring**

Threat Modeling

- What are we building?
- What can go wrong?
- What are we doing about it?
- Did we do a good job?



Movie Streaming

What are we building?

“

Testing externally visible behavior as a user would see it.

”

Black-box Monitoring

Our Monitoring System





What Can go wrong?

STRIDE

S – Spoofing

T – Tampering

R – Repudiation

I – Information Disclosure

D – Denial of Service

E – Elevation of Privileges

Spoofing

```
function isAuthorized(user)
{
  if (user.Name == "test-bot"){
    return true
  }
}
```


Repudiation



Denial of Service



What are we doing about it?

Potential Mitigations

- Least Privilege
- Block access
- Tracing
- Limit to test data only

Did we do a good job?

Wrapping Up

Key Takeaway

- Monitoring is just code
- Careful when exposed to the internet
- Conduct threat model for monitoring



One line of text slide

Questions?

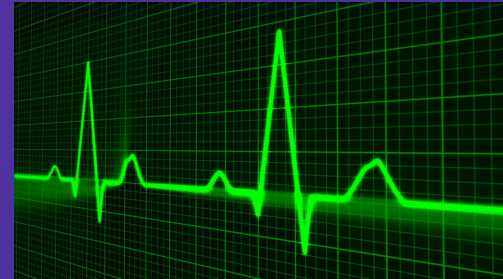


Agenda

Threat Modeling



Monitoring



Snyk Infrastructure as Code

Developer-focused configuration security

snyk

Product ▾

Pricing

Docs ▾

Learn ▾

Company ▾

SnykCon ^{NEW}

Log in

BOOK A DEMO

QUICK START

Snyk Infrastructure as Code

Put cloud native configuration security in the hands of developers

QUICK START FOR FREE



Find and fix security issues in Terraform and Kubernetes code



Thanks for listening

Sign up for free at snyk.io/signup