Container Security Monitoring using Open Source

Madhu Akula, Appsecco

About Me

- Automation Ninja at Appsecco
- Expert in Security, Containers, Cloud and DevOps
- Speaker & Trainer at Defcon, Blackhat USA, Appsec EU, DevSecCon, All Day DevOps, c0c0n, null, etc.
- Discovered security vulnerabilities in Google, Microsoft, Yahoo, Adobe, LinkedIn, Ebay, AT&T, Blackberry, Cisco and many more
- Co-author of Security Automation with Ansible2
- Follow me (or) tweet to me at @madhuakula
- Never Ending Learner!

What we are going to learn

- Why do we need container security monitoring?
- Container security monitoring
- How to implement container security monitoring?
- Introducing Sysdig Falco
- Automated defence based on container security attacks - demo!
- Other use cases
- References and resources

Importance of container security monitoring

An application vulnerability Server-Side Request Forgery (SSRF) in one of the containers running in Kubernetes cluster allows attackers to access and gain control over the entire Shopify cluster and instances.

Read more about https://hackerone.com/reports/341876

Importance of container security monitoring

By pushing malicious images to a Docker Hub registry and pulling it from the victim's system, hackers were able to mine 544.74 Monero, which at time was equal to approximately $90,000

Read more about
https://kromtech.com/blog/security-center/cryptojacking-invades-cloud-how-modern-containerization-trend-is-exploited-by-attackers

Why use container security monitoring?

- It enables us to perform the practice of looking at security frequently and easily
- To make important decisions for troubleshooting, monitoring is the place to start
- Taking certain actions based on the monitoring alerts
- We can apply the feedback into the system to learn and make it better

Container security monitoring

- Collecting logs from containers to a centralised location
- Analysing logs to identify suspicious activity over a period of time
- Triggering alerts based on the thresholds and pattern matches
- It allows us to take actions based on the alert
- Helps us to apply feedback into knowledge bases and existing systems to make it better

How to do container security monitoring

- There are multiple tools and methods to perform container security monitoring:
  a. Collection
  b. Identification
  c. Actionable output

How to do container security monitoring

- Collection
  - Key thing is to collect the logs and events related to the containers and store them in a centralised place for analysis
- Identification
  - Creating attack patterns to identify attacks and alert them
- Actionable output
  - Triggering alerts based on the identified attack patterns
  - Human (or) playbook (or) defence system to take actions based on the alerts triggered

Simple and quick way to look at logs

The simplest way to look the container events and logs is using built-in commands

- docker events
- docker logs
- kubectl logs

docker events example

```
$ docker events
2018-10-15T00:28:07.748589607+05:30 container create 39a297e1fb0c484776338d07d44ad6391a287d816cb4a041f67b33731c58e48c (
image=alpine, name=wonderful_colden)
2018-10-15T00:28:07.750293594+05:30 container attach 39a297e1fb0c484776338d07d44ad6391a287d816cb4a041f67b33731c58e48c (
image=alpine, name=wonderful_colden)
2018-10-15T00:28:07.791296902+05:30 network connect a8544c935d78105cf9d803245f47f56100941806c51a57f390c3cb96d9841f4c (c
ontainer=39a297e1fb0c484776338d07d44ad6391a287d816cb4a041f67b33731c58e48c, name=bridge, type=bridge)
2018-10-15T00:28:08.184588799+05:30 container start 39a297e1fb0c484776338d07d44ad6391a287d816cb4a041f67b33731c58e48c (i
mage=alpine, name=wonderful_colden)
2018-10-15T00:28:08.185870420+05:30 container resize 39a297e1fb0c484776338d07d44ad6391a287d816cb4a041f67b33731c58e48c (
height=16, image=alpine, name=wonderful_colden, width=59)
2018-10-15T00:28:38.785913151+05:30 container resize 39a297e1fb0c484776338d07d44ad6391a287d816cb4a041f67b33731c58e48c (
height=16, image=alpine, name=wonderful_colden, width=119)
2018-10-15T00:28:42.346538950+05:30 container die 39a297e1fb0c484776338d07d44ad6391a287d816cb4a041f67b33731c58e48c (exi
tCode=0, image=alpine, name=wonderful_colden)
2018-10-15T00:28:42.554423533+05:30 network disconnect a8544c935d78105cf9d803245f47f56100941806c51a57f390c3cb96d9841f4c
 (container=39a297e1fb0c484776338d07d44ad6391a287d816cb4a041f67b33731c58e48c, name=bridge, type=bridge)

$ docker run --rm -it alpine sh
/ # id
uid=0(root) gid=0(root) groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel),11(floppy),20(dialout),26(tape)
,27(video)
/ #
```

docker logs example

```
$ docker run --rm -d --name nginx -p 80:80 nginx:alpine
a18f47b57b7187e66a81931bc7f056b8b6a87680c9a14db6cc5832370bb4c84d

$ docker logs -f nginx
172.17.0.1 - - [14/Oct/2018:19:01:41 +0000] "GET / HTTP/1.1" 200 612 "-" "HTTPie/0.9.8" "-"
172.17.0.1 - - [14/Oct/2018:19:01:45 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.58.0" "-"


$ curl localhost:80
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

kubectl logs example

```
user@cloudshell:~ $ kubectl logs -f node-app-ffbc66c69-d86m4

> m-p1@1.0.0 start /app
> node server.js

POST / 200 1290 - 50.002 ms
POST / 200 1296 - 103.990 ms
POST / 200 1254 - 5.642 ms
POST / 200 1254 - 5.553 ms
```

Container integrity checks example

```
student@debian:~$ docker run --name checkintegriy  -it ubuntu:latest bash
root@8318d73a73cb:/# mkdir -p /data/output
root@8318d73a73cb:/# echo "modifed this stuff" > /.dockerenv
root@8318d73a73cb:/# exit
exit
student@debian:~$
```

```
student@debian:~$ docker diff checkintegriy
A /data
A /data/output
C /root
A /root/.bash_history
C /.dockerenv
student@debian:~$
```

System level monitoring using cAdvisor

Overview

Processes

User	PID	PPID	Start Time	CPU %	MEM %	RSS	Virtual Size	Status	Running Time	Command	Container
root	135	1	17:47	3.30	4.60	78.72 MiB	455.85 MiB	Sl	00:00:01	dockerd	/kubepods/besteffort/podfe33
root	384	369	17:47	2.20	1.30	23.06 MiB	444.79 MiB	Ssl	00:00:00	cadvisor	/kubepods/besteffort/podfe33
root	8	1	17:47	1.30	0.10	2.19 MiB	244.26 MiB	Ssl	00:00:00	rsyslogd	/kubepods/besteffort/podfe33
ninjama+	255	1	17:47	0.50	1.10	19.84 MiB	56.57 MiB	S	00:00:00	python	/kubepods/besteffort/podfe33
root	143	135	17:47	0.10	1.30	22.50 MiB	284.07 MiB	Ssl	00:00:00	docker-containe	/kubepods/besteffort/podfe33
ninjama+	262	255	17:47	0.10	0.70	11.95 MiB	541.26 MiB	Sl	00:00:00	python	/kubepods/besteffort/podfe33
ninjama+	263	255	17:47	0.10	0.90	16.78 MiB	41.58 MiB	S	00:00:00	python	/kubepods/besteffort/podfe33
root	364	135	17:47	0.10	0.30	5.36 MiB	110.47 MiB	Sl	00:00:00	docker-proxy	/kubepods/besteffort/podfe33
root	1	0	17:47	0.00	0.10	2.94 MiB	17.56 MiB	Ss	00:00:00	bash	/kubepods/besteffort/podfe33
root	57	1	17:47	0.00	0.10	3.05 MiB	68.31 MiB	Ss	00:00:00	sshd	/kubepods/besteffort/podfe33
root	256	1	17:47	0.00	0.00	1.41 MiB	24.80 MiB	S	00:00:00	logger	/kubepods/besteffort/podfe33
root	259	1	17:47	0.00	0.00	716.00 KiB	4.10 MiB	S	00:00:00	sleep	/kubepods/besteffort/podfe33
root	270	57	17:47	0.00	0.30	6.38 MiB	84.52 MiB	Ss	00:00:00	sshd	/kubepods/besteffort/podfe33
ninjama+	272	270	17:47	0.00	0.20	4.27 MiB	84.52 MiB	S	00:00:00	sshd	/kubepods/besteffort/podfe33
ninjama+	273	272	17:47	0.00	0.10	2.82 MiB	10.93 MiB	Ss+	00:00:00	start-shell.sh	/kubepods/besteffort/podfe33
ninjama+	274	272	17:47	0.00	0.00	1.66 MiB	12.40 MiB	Ss	00:00:00	sftp-server	/kubepods/besteffort/podfe33
ninjama+	279	273	17:47	0.00	0.10	2.88 MiB	18.88 MiB	S+	00:00:00	tmux:	
ninjama+	281	1	17:47	0.00	0.10	3.09 MiB	27.29 MiB	Ss	00:00:00	tmux:	
ninjama+	282	281	17:47	0.00	0.30	5.40 MiB	21.30 MiB	Ss+	00:00:00	bash	/kubepods/besteffort/podfe33
ninjama+	328	1	17:47	0.00	0.00	332.00 KiB	10.83 MiB	Ss	00:00:00	ssh-agent	/kubepods/besteffort/podfe33

What about security specific monitoring?

There are many open source tools available in the market we can use, based on our needs and use cases. Here are some of them:

- AppArmor, Seccomp, SELinux, etc.
- Auditd, go-audit
- Elastic Beats
- Sysdig Falco

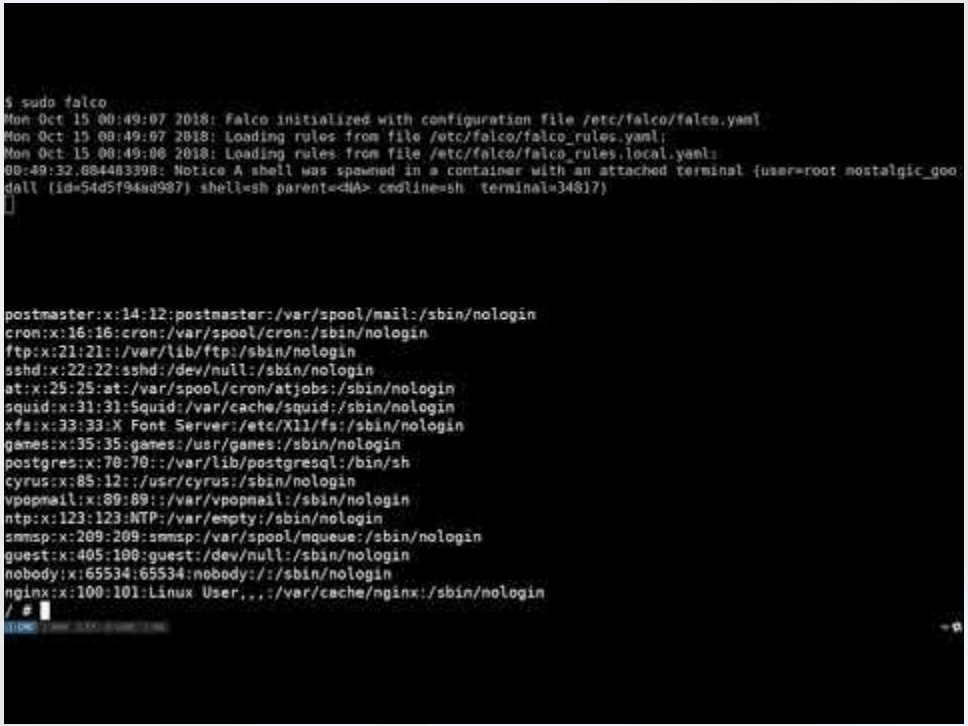
Why Sysdig Falco?

- Sysdig Falco is a behavioral activity monitor designed to detect anomalous activity in applications
- Falco lets us continuously monitor and detect anomalies in containers, applications, hosts, and network activity
- Rich ruleset and ability to filter events for taking action
- It supports Docker, Kubernetes, Mesos, etc.
- Part of CNCF Sandbox project
- https://falco.org

Why Sysdig Falco?

- Sysdig Falco is an auditing and monitoring tool where as AppArmor or Seccomp are enforcement tools
- Sysdig Falco runs in user space, using a kernel module to intercept system calls
- Installing Falco and integrating with external resources is simple compared to other tools

DEMO - Introducing Sysdig Falco

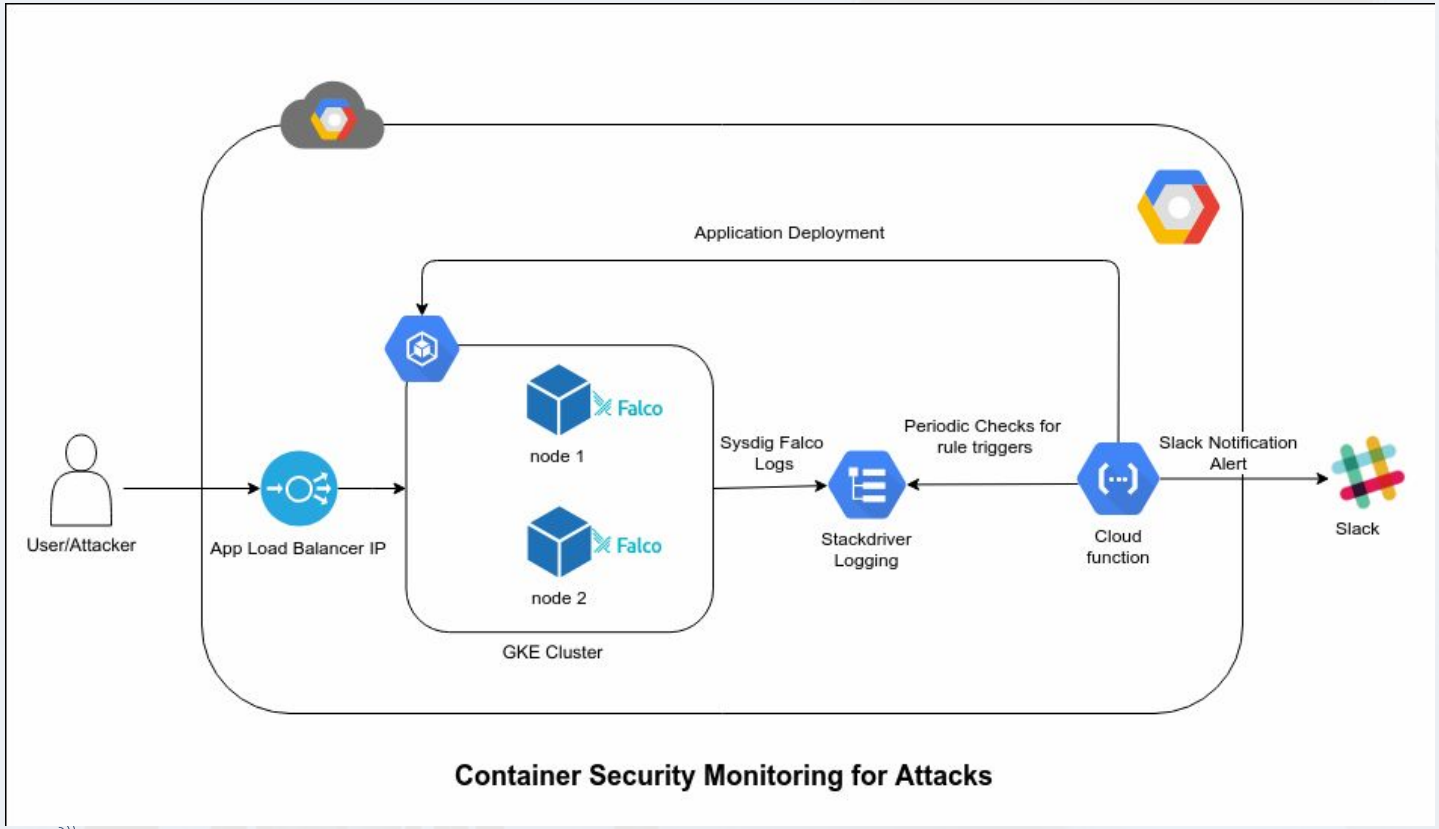


https://youtu.be/A41bAUzvym0

Automated defense for container security

In this context, automated defence means applying defence against security attacks based on an attack pattern (or) an alert

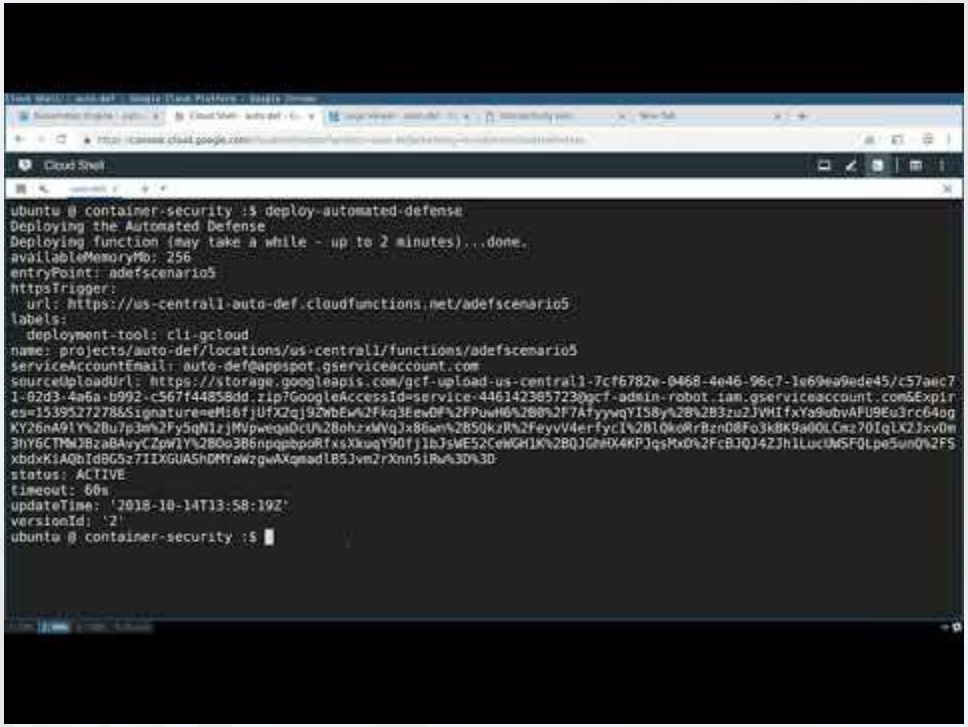
Automated defense infrastructure setup



Automated defense infrastructure setup

- We have 2 node Kubernetes cluster in the Google Cloud Platform
- Deploy the Sysdig Falco helm chart (installs as daemonset in the cluster)
- Deploy simple vulnerable web application
- Configure the Cloud Function to read GKE stackdriver logging and trigger alerts
- Slack channel for communication medium based on alerts
- Automated defence code for deploying the new releases with fixes based on attacks

DEMO - Automated defense



https://youtu.be/zd0ksjZI5Vk

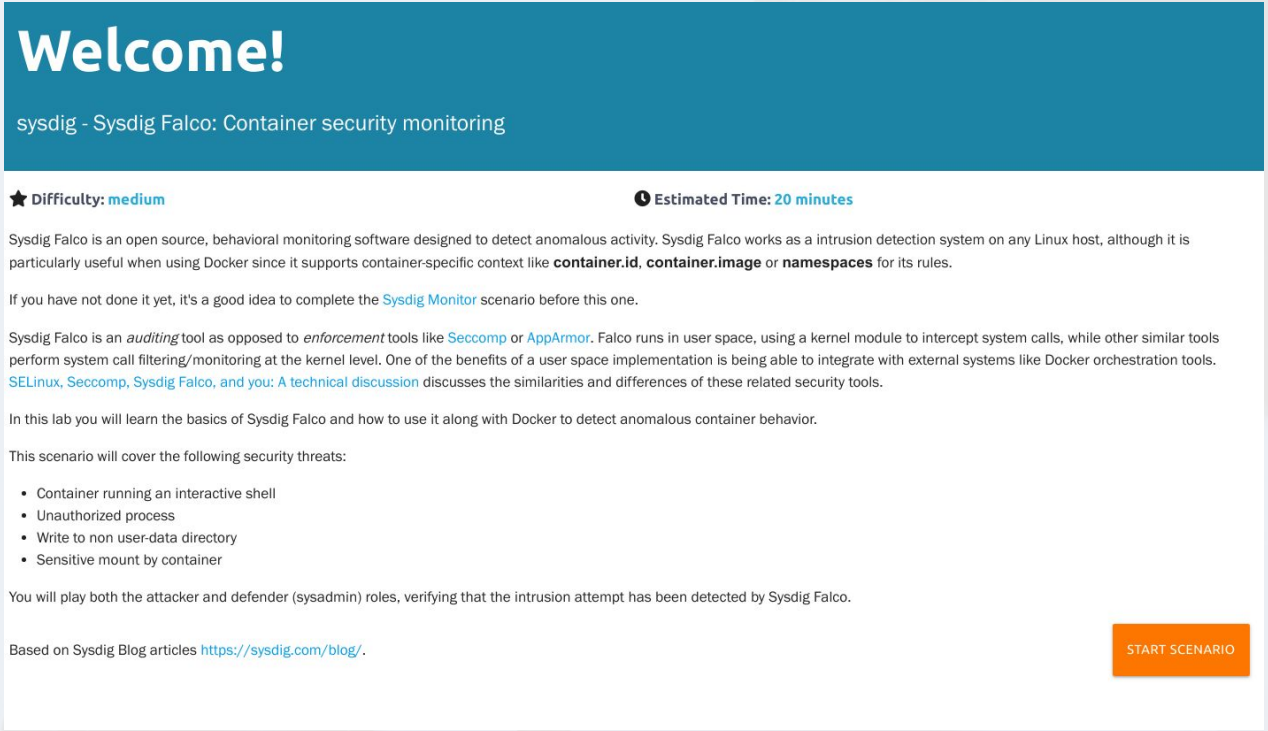
What just happened?

- We can see that an attacker performs a command injection attack to gain access to the system files
- The attack is identified and triggered as a slack alert
- After getting the alert we deploy the automated defence system
- Now the automated defense system will deploy a defence based on the alert and attack pattern
- If the attacker tries to perform the same attack, the attack fails because the fix is already in place

Use cases

- Applying continuous security monitoring checks for deployments
- Building near real-time security defence for containerised environments
- Using automated defence with servicemesh using tools like Istio, Envoy, etc.

Want to try it yourself in a browser?



Welcome!

sysdig - Sysdig Falco: Container security monitoring

Difficulty: medium **Estimated Time:** 20 minutes

Sysdig Falco is an open source, behavioral monitoring software designed to detect anomalous activity. Sysdig Falco works as a intrusion detection system on any Linux host, although it is particularly useful when using Docker since it supports container-specific context like **container.id**, **container.image** or **namespaces** for its rules.

If you have not done it yet, it's a good idea to complete the Sysdig Monitor scenario before this one.

Sysdig Falco is an *auditing* tool as opposed to *enforcement* tools like Seccomp or AppArmor. Falco runs in user space, using a kernel module to intercept system calls, while other similar tools perform system call filtering/monitoring at the kernel level. One of the benefits of a user space implementation is being able to integrate with external systems like Docker orchestration tools. SELinux, Seccomp, Sysdig Falco, and you: A technical discussion discusses the similarities and differences of these related security tools.

In this lab you will learn the basics of Sysdig Falco and how to use it along with Docker to detect anomalous container behavior.

This scenario will cover the following security threats:

- Container running an interactive shell
- Unauthorized process
- Write to non user-data directory
- Sensitive mount by container

You will play both the attacker and defender (sysadmin) roles, verifying that the intrusion attempt has been detected by Sysdig Falco.

Based on Sysdig Blog articles https://sysdig.com/blog/.

START SCENARIO

https://www.katacoda.com/sysdig/scenarios/sysdig-falco

Credits and acknowledgements

- Sysdig Falco
- Docker Community
- K8S Community
- Subash SN
- Akash Mahajan

References and resources

- https://falco.org
- https://kubernetes.io/docs
- https://docs.docker.com
- https://www.katacoda.com
- https://sysdig.com/blog/oss-container-security-runtime
- https://blog.appsecco.com/automated-defense-using-serverless-computing-84ee04b9b129

Thank You All Day DevOps Sponsors

Thank You All Day DevOps Supporters

Meet Me in the Slack Channel for Q&A

bit.ly/addo-slack

Thank You

@madhuakula | @appseccouk